Smarter. Greener. Together.

DVP-ES3 **Series Programming Manual**

DVP-0289720-01
2019/10/08

DVP-ES3 **Series**
Programming Manual

Smarter. Greener. Together.

DVP-ES3 Series Programming Manual

Revision History

Version	Revision	Date
1st	The first version was published.	2019/10/08

DVP-ES3 Series Programming Manual

Table of Contents

Chapter 6 Applied Instructions

1

Chapter 1 Product Introduction

Table of Contents

1.1 Overview

This manual introduces you to programming the DVP-ES3 Series programmable logic controllers, the basic instructions, and the applied instructions.

1.1.1 Related Manuals

The related manuals for DVP-ES3 Series programmable logic controllers are listed below.

- DVP-ES3 Series Programming Manual
  This introduces programming for the DVP-ES3 Series programmable logic controllers, basic instructions, and applied instructions.
  For DVP-ES2 Series PLC, refer to DVP-ES2-/EX2-SS2/SA2/SX2/SE & TP Operation Manual – Programming.
- ISPSoft User Manual
  This introduces the use of the ISPSoft software, programming language (Ladder, SFC, FBD, and ST), POUs, and tasks.
  DVP-ES3 Series PLC can only use ISPSoft for programming, NOT WPLSoft.
- DVP-ES3 Series Operation Manual
  This introduces electrical specifications, dimensions, CPU functions, devices, module tables, troubleshooting, and so forth.

1.1.2 Models Descriptions

Classification	Model Name	Description
DVP-ES3 Series CPU module	DVP32ES311T	24 VDC powered CPU module NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 32 I/Os (16DI+16DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP32ES300T	100-220 VAC powered CPU module NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 32 I/Os (16DI+16DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP32ES300R	100-220 VAC powered CPU module Relay output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 32 I/Os (16DI+16DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP48ES300T	100-220 VAC powered CPU module

Classification	Model Name	Description
		NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 48 I/Os (24DI+24DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP48ES300R	100-220 VAC powered CPU module Relay output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 48 I/Os (24DI+24DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP64ES300T	100-220 VAC powered CPU module NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 64 I/Os (32DI+32DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP64ES300R	100-220 VAC powered CPU module Relay output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 64 I/Os (32DI+32DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP80ES300T	100-220 VAC powered CPU module NPN output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 80 I/Os (40DI+40DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
	DVP80ES300R	100-220 VAC powered CPU module Relay output, 1x Ethernet port, 2x RS-485 ports, 1x USB port, 1x Micro SD interface, supporting 80 I/Os (40DI+40DO) and up to 256 I/Os. Program capacity: 64K steps, removable terminal blocks
Digital input/output module	DVP08XM211N	8 inputs 24VDC 5mA
	DVP08XP211R	4 inputs 24VDC

Classification	Model Name	Description
		5mA 4 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP08XP211T	4 inputs 24VDC 5mA 4 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
	DVP08XN211R	8 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP08XN211T	8 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
	DVP16XM211N	16 inputs 24VDC 5mA
	DVP16XP211R	8 inputs 24VDC 5mA 8 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP16XP211T	8 inputs 24VDC 5mA 8 NPN (sinking) outputs

Classification	Model Name	Description
		5 – 30VDC 0.5A/output, 4A/COM
	DVP16XN211R	16 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP16XN211T	16 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
	DVP24XP200R	16 inputs 24VDC 5mA 8 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP24XP200T	16 inputs 24VDC 5mA 8 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
	DVP24XN200R	24 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP24XN200T	24 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
	DVP32XP200R	16 inputs 24VDC 5mA

Classification	Model Name	Description
		8 Relay outputs 250VAC Below 30VDC 2A/output, 5A/COM
	DVP32XP200T	16 inputs 24VDC 5mA 16 NPN (sinking) outputs 5 – 30VDC 0.5A/output, 4A/COM
Analog input/output module	DVP04AD-E2	4-channel analog input Hardware resolution 14 bits: -5V ~ +5V, -10V ~ +10V, -20mA ~ +20mA Hardware resolution 13 bits: 0/4 ~ 20mA Conversion time: 400 μs/channel
	DVP02DA-E2	2-channel analog input Hardware resolution 14 bits: -10V ~ +10V, -20mA ~ +20mA Conversion time: 400 μs/channel
	DVP04DA-E2	4-channel analog input module Hardware resolution 14 bits: -10V ~ +10V, -20mA ~ +20mA Conversion time: 400 μs/channel
	DVP06XA-E2	4-channel analog input Hardware resolution 14 bits: -5V ~ +5V, -10V ~ +10V, -20mA ~ +20mA Hardware resolution 13 bits: 0/4 ~ 20mA Conversion time: 400 μs/channel 2-channel analog output Hardware resolution 14 bits: -10V ~ +10V, -20mA ~ +20mA Conversion time: 400 μs/channel
Temperature measurement module	DVP04PT-E2	4-channel, 2-wire/3-wire RTD Sensor type: Pt100 / Pt1000 / Ni100 / Ni1000 / 0-300Ω / 0-3000Ω Resolution: 0.1°C/0.1°F (16 bits)

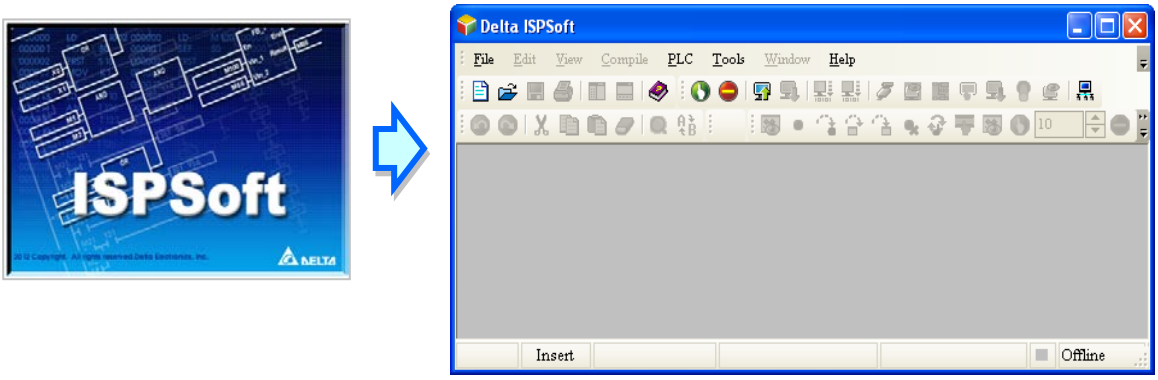
Classification	Model Name	Description
		Conversion time: 200 ms/channel PID controller
	DVP06PT-E2	6-channel, 2-wire/3-wire RTD Sensor type: Pt100 / Pt1000 / Ni100 / Ni1000 / Cu50 / Cu100 / 0-300Ω / 0-3000Ω / JPt100 / LG-Ni1000 Resolution: 0.1°C/0.1°F (16 bits) Conversion time: 200 ms/channel PID controller
	DVP04TC-E2	4-channel thermocouple Sensor type: J, K, R, S, T, E, N and -80 to +80 mV Resolution: 0.1°C/0.1°F (16 bits) Conversion time: 200 ms/channel PID controller
External terminal module	DVPAEXT01-E2	For DVP-ES2/ES3 Series PLC

1

1.2 Software

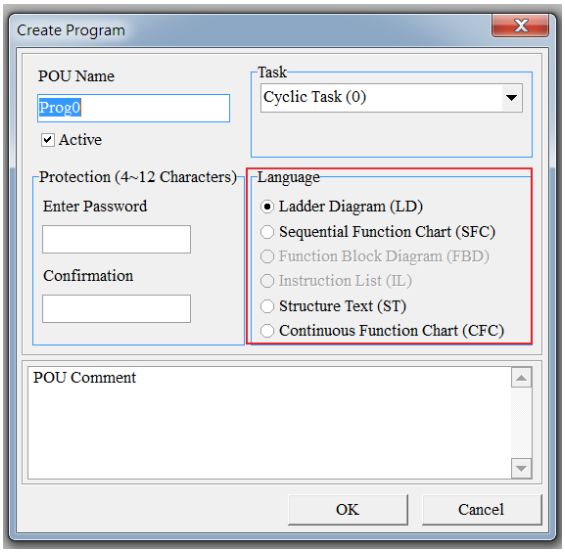
1.2.1 Program Editor

The section describes the program editor ISPSoft.

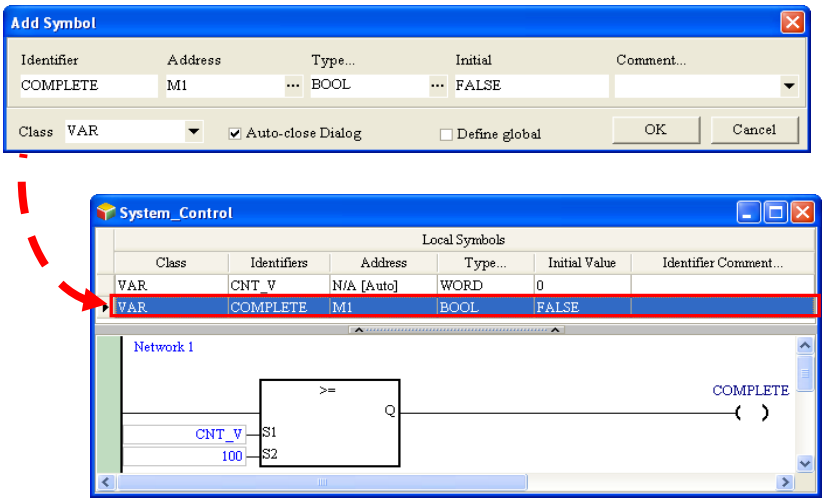


- There are four types of programming languages: structure text (ST), ladder diagram (LD), sequential function chart (SFC), and continuous function chart (CFC).

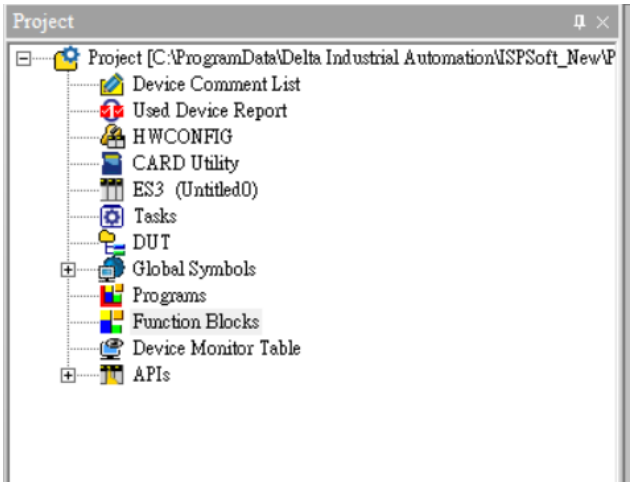
  NOTE: ISPSoft version 3.08 or higher support ES3 Series programming.



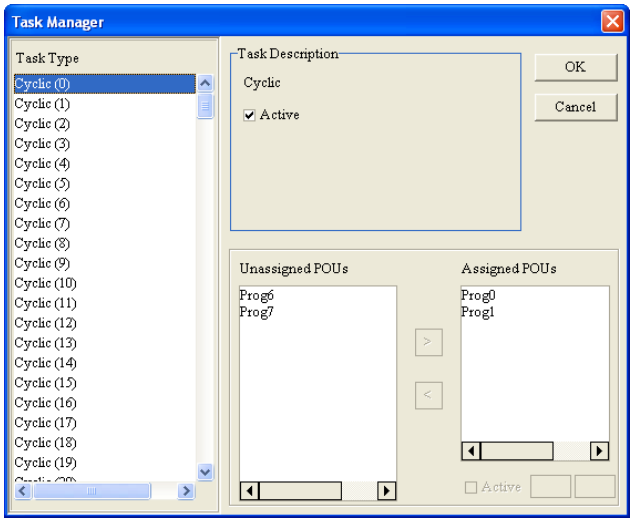
- User-defined variables allow you to define a variable to replace a PLC device name. This enhances the readability of the program, and saves time when addressing the device.



- The Program Organization Unit (POU) framework divides the main program into several program units, and also replaces the traditional subroutines with functions and function blocks. The makes the framework of the program modular and easier to manage.



- Tasks manage the execution order of the programs. Tasks help you manage large-scale program development.



1.2.2 Program Organization Units and Tasks

The Program Organization Units (POUs) are the basic elements that constitute the PLC program. Unlike the traditional PLC program, the program framework introduced by IEC 61131-3 allows you to divide a large program into several small units. These small units are called POUs. The POUs can be classified into three types.

1. Program (PROG): The program POU is the main program in the PLC. You can define the execution of this POU t to be cyclic scan or interrupt driven, and arrange the scan order in the task list for program POUs.

2. Function block (FB): The function block (FB) POU is similar to a subroutine. The instructions in the function block are executed after a program POU calls the function block with the related parameters.

3. Function (FC): The function (FC) POU in similar to a macro instruction. That is, you can write many operation instructions or functions into a function-type POU, and then use then in a program POU or a function block POU.

Tasks are functions that control the order of program execution or according to certain interrupt conditions. The task provides each program POU with a specific execution task, and specifies the execution order for the program POUs or the way to enable them.

Normally, only some of the program POUs in a project take part in the actual execution. The task controls whether to execute the program POU or not, and how to execute it. If the POU of the program type is not assigned in the task, the program POU is saved as ordinary source code with the project instead of being compiled into the execution code for the PLC. In addition, only the program POU needs to be assigned to the

task. Function block POUs or function POUs are automatically called by the program POU. There are three types of tasks.

1. Cyclic task: The program POUs assigned to cyclic tasks are scanned cyclically, and executed in order.

2. Timed interrupt task: If the interrupt time is reached, all program POUs assigned to the timed interrupt task are executed in order.

3. Conditional interrupt task: Conditional Interrupts can be divided into several types, such as external interrupts, and I/O interrupts. You must make sure that the PLC supports the interrupts before you use conditional interrupts in a project. If you assign a program POU to a conditional interrupt task, the program POU is similar to an interrupt subroutine. When the interrupt condition is satisfied (for example, the contact of the external interrupt is triggered) then all program POUs assigned to the conditional interrupt task are executed in order.

MEMO

2

Chapter 2 Devices

Table of Contents

2.1 Introduction to Devices

This section describes the values and strings processed by the PLC. It also describes the functions of devices, including input, output and auxiliary relays, as well as timers, counters, and data registers. The PLC simulates external devices in the PLC's internal memory, so the word "device" is a generic name that refers to all the internal memory locations in the PLC. A device can be a bit device or a word device. Bit devices simulate coils, contacts and flags, while word devices simulate registers.

2.1.1 Device Table

Type	Device name		Number of devices	Range
Bit device	Input relay	X	256	X0–X377
	Output relay	Y	256	Y0–Y377
	Data register	D	48,0000	D0.0–D29999.15
		W	48,0000	W0.0–W29999.15 *4
	Auxiliary relay	M	8192	M0–M8191
	Special auxiliary relay	SM	2048	SM0–SM4095
	Flag	S	2048	S0–S2047
	Timer	T	512	T0–T511
	Counter	C	512	C0–C511
	32-bit counter	HC	256	HC0–HC255
Word device	Input relay	X	64	X0–X63
	Output relay	Y	64	Y0–Y63
	Data register	D	30000	D0–D29999
		W	30000	W0–W29999 *4
	Special auxiliary relay	SR	2048	SR0–SR2047
	File register	FR	65536	FR0–FR65535
	Timer	T	512	T0–T511
	Counter	C	512	C0–C511
	32-bit counter	HC	256（512 words）	HC0–HC255
	Index register	E	10	E0–E9
			5	E10–E14 *4
Constant*1	Decimal system	K	16 bits: -32768 to 32767 32 bits: -2147483648 to 2147483647	
Constant*2	Hexadecimal system	16#	16 bits: 16#0–16#FFFF 32 bits: 16#0–16#FFFFFFFF	
	Single-precision floating-point number	F	32 bits: $\pm 1.17549435^{-38}$ to $\pm 3.40282347^{+38}$	
String*3	String	"$"	1–31 characters	

*1: Constants are indicated by K in the device lists in Chapter 5 and Chapter 6 in the DVP-ES3 Series Programming Manual. For example, when "K50" appears in the DVP-ES3 Series Programming Manual, enter only the number 50 in ISPSoft.

*2: Floating-point numbers are indicated by F/DF in the device lists in Chapter 5 and Chapter 6 in the DVP-ES3 Series

Programming Manual, but they are represented by decimal points in ISPSoft. For example, for the floating-point number F500, enter 500.0 in ISPSoft.

*3: Strings are indicated by $ in Chapter 5 and Chapter 6 in the DVP-ES3 Series Programming Manual, but they are represented by quotes (“ ”) in ISPSoft. For example, for the string of 1234, enter “1234” in ISPSoft.

*4: Used for editing in ISPSoft only.

2.1.2 Basic Structure of I/O Storage

Device	Function	Access by bits	Access by words	Modify by ISPSoft	Force the bit ON/OFF
X	Input relay	OK	OK	OK	OK
Y	Output relay	OK	OK	OK	OK
M	Auxiliary relay	OK	-	OK	-
SM	Special auxiliary relay	OK	-	OK	-
S	Flag	OK	-	OK	-
T	Timer	OK	OK	OK	-
C	Counter	OK	OK	OK	-
HC	32-bit counter	OK	OK	OK	-
D	Data register	OK	OK	OK	OK
SR	Special data register	-	OK	OK	-
FR	File register	-	OK*1	-	-
E	Index register	-	OK	OK	-

*1: Use an instruction for writing to an FR.

2.1.3 Relation Between the PLC Action and the Device Type

PLC action \ Device type		Non-latched area		Latched area	
		Device Y	**Other devices**	**File register**	**Other devices**
Power: OFF→ON		Cleared	Cleared	Retained	Retained
Restore to defaults		Cleared	Cleared	Cleared	Cleared
STOP ↓ RUN*1	**The non-latched area is cleared.**	Cleared	Cleared	Retained	Retained
	The state of the non-latched area is retained.	Retained	Retained	Retained	Retained
RUN ↓ STOP*1	**The state of device Y is cleared.**	Cleared	Retained	Retained	Retained
	The state of device Y is retained.	Retained	Retained	Retained	Retained
SM204 is ON. (All non-latched areas are cleared.)		Cleared	Cleared	Retained	Retained
SM205 is ON. (All latched areas are cleared.)		Retained	Retained	Retained	Cleared

*1: For more on setting the states, see HWCONFIG in ISPSoft. The default for PLC STOP->RUN is “clear not-latched area”. The default for PLC RUN->STOP is “clear the state of device Y”.

2.1.4 Latched Areas in the Device Range

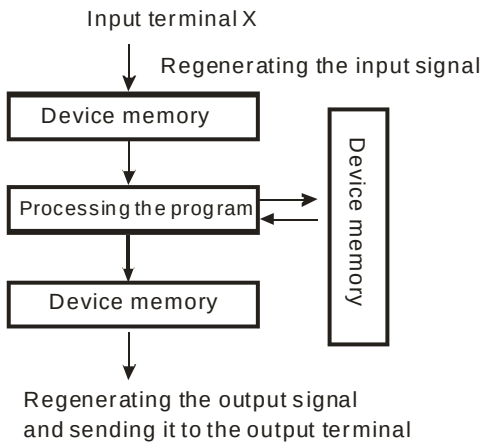
Device	Function	Device range	Latched area
X	Input relay	X0–X377	All devices are non-latched.
Y	Output relay	Y0–Y377	All devices are non-latched.
M*1	Auxiliary relay	M0–M8191	The default range is M6000–M8191.
SM	Special auxiliary relay	SM0–SM2047	Some devices are latched, and cannot be changed. Refer to the list of special auxiliary relays for more information.
S*1	Flag	S0–S2047	The default range is S512–S1023
T	Timer	T0–T511	All devices are non-latched.
C*1	Counter	C0–C511	The default range is C448–C511
HC*1	32-bit counter	HC0–HC255	The default range is HC128–HC255
D*1	Data register	D0–D29999	The default range is D20000–D29999
		W0–W29999	*2
FR	File register	FR0–FR65535	All devices are latched.
SR	Special data register	SR0–SR2047	Some are latched, and cannot be changed. Refer to the list of special data registers for more information.
E	Index register	E0–E9	All devices are non-latched.
		E10–E14	*2

*1: For more information on setting the latched area, see HWCONFIG in ISPSoft. Setting the latched area means the other areas are seen as non-latched areas. The range of latched areas cannot exceed the device range. For example, setting M600–M7000 as latched areas makes M0–M5999 and M7001–M8191 non-latched areas.

*2: Used for editing in ISPSoft only.

2.2. Device Functions

The following flow chart shows the procedure for processing a program in the PLC.

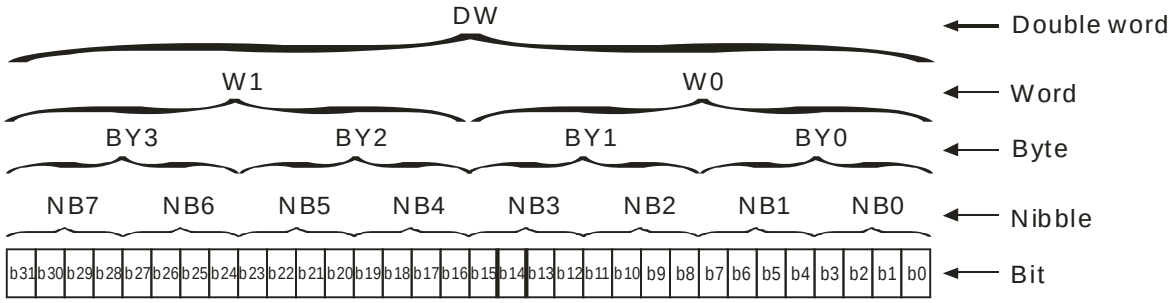


- Regenerating the input signal
  1. Before the program is executed, the state of the external input signal is read into the memory location for the input signal.
  2. When program is executed, the state in the memory location for the input signal does not change even if the input signal changes from ON to OFF or from OFF to ON. The input signal is not refreshed until the next scan begins.
- Processing the program
  After the input signal is refreshed, the instructions in the program are executed in order from the start address of the program. The results are stored in the device memory.
- Regenerating the state of the output
  After the instruction END is executed, the state in the device memory is sent to the specified output terminal.

2.2.1 Values and Constants

Name	Description
Bit	A bit is the basic unit in the binary system. Its state is either 1 or 0.
Nibble	A nibble is composed of four consecutive bits (for example b3–b0). Nibbles can represent 0–9 in the decimal system, or 0–F in the hexadecimal system.
Byte	A byte is composed of two consecutive nibbles (8 bits, b7–b0). Bytes can represent 00–FF in the hexadecimal system.
Word	A word is composed of two consecutive bytes (16 bits, b15–b0). Words can represent 0000–FFFF in the hexadecimal system.
Double word	A double word is composed of two consecutive words (i.e. 32 bits, b31–b0). Double words represent 00000000–FFFFFFFF in the hexadecimal system.

The relation among bits, nibbles, bytes, words, and double words in the binary system is shown in the picture below.



The PLC uses four types of values to execute the operation according to different control purposes.

1. Binary number (BIN)

   The PLC uses the binary system to operate on the values.

2. Octal number (OCT)
   DVP-PLC uses the octal number on the external inut and output device number. For example:

   External input device number: X0~X7, X10~X17, to X377.

   External output device number: Y0~Y7, Y10~Y17, to Y377.

3. Decimal number (DEC)

   The PLC uses decimal numbers for:

   - The setting value of a timer (T) or the setting value of a counter (C/HC); for example, TMR C0 50 (**constant K**).
   - The device number S, M, T, C, D, E; for example, M10 and T30 (device number)
   - **The constant K**, used as the operand in an applied instruction. For example, MOV 123 D0 **(constant K)**.

4. Binary-coded decimal (BCD)

   A decimal value that is represented by a nibble or four bits so that sixteen consecutive bits represent a four-digit decimal value.

5. Hexadecimal number (HEX)

   The PLC uses hexadecimal numbers for:

   - **The constant 16#**, used as the operand in an applied instruction; for example, MOV 16#1A2B D0

     (hexadecimal constant).

The following table shows the corresponding values.

Binary Number (BIN)	**Decimal Number (DEC)**	**Binary Code Decimal (BCD)**	**Hexadecimal Number (HEX)**
PLC internal execution	**Constant** K, Device number	BCD related instruction	Constant 16#, Device number
0000	0	0000	0
0001	1	0001	1
0010	2	0010	2
0011	3	0011	3
0100	4	0100	4
0101	5	0101	5
0110	6	0110	6
0111	7	0111	7
1000	8	1000	8
1001	9	1001	9
1010	10	-	A
1011	11	-	B
1100	12	-	C
1101	13	-	D
1110	14	-	E

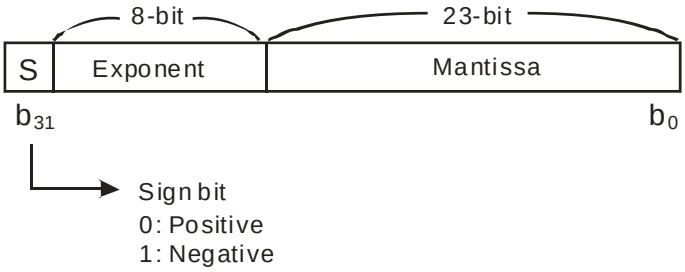
Binary Number (BIN)	Decimal Number (DEC)	Binary Code Decimal (BCD)	Hexadecimal Number (HEX)
1111	15	-	F
10000	16	0001 0000	10
10001	17	0001 0001	11

2.2.2 Floating-point Numbers

Floating-point numbers are represented by decimal points in ISPSoft. For example, the floating-point number 500 is represented as 500.0.

2.2.2.1 Single-precision Floating-point Numbers

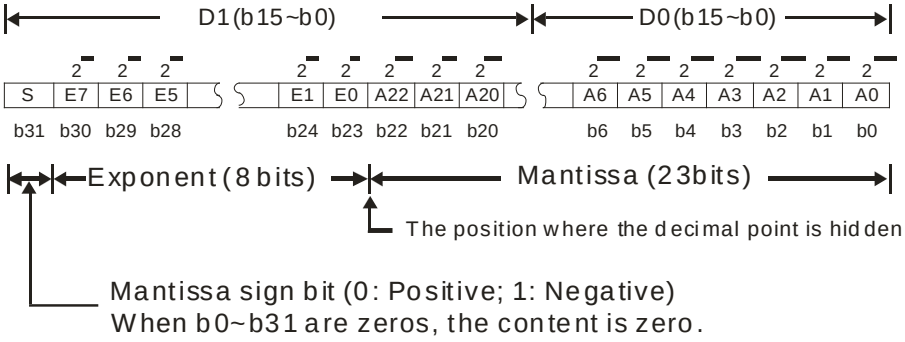
Floating-point numbers are represented by a 32-bit register. The representation adopts the IEEE754 standard, and the format shown in the following picture.



Equation: $(-1)^S \times 2^{E-B} \times 1.M; B = 127$

The single-precision floating-point numbers range between $\pm 2^{-126}$ to $\pm 2^{+128}$, and correspond to the range between $\pm 1.1755 \times 10^{-38}$ to $\pm 3.4028 \times 10^{+38}$.

The DVP-ES3 Series PLC uses two consecutive registers for a 32-bit floating-point number. Take (D1, D0) for example.



Example 1:

23 is represented by a single-precision floating-point number.

Step 1: Convert 23 into the binary number, 23.0=10111.

Step 2: Normalize the binary number, 10111=1.0111 ×24 (0111 is the mantissa, and 4 is the exponent.).

Step 3: Get the value of the exponent.

∵ E-B=4→E-127=4 ∴ E=131=100000112

Step 4: Combine the sign bit, the exponent, and the mantissa to form the floating-point number.

0 100**0001**1 011**1000**0000**0000**0000**0000**$_2$=41B80000$_{16}$

Example 2:

-23 is represented by a single-precision floating-point number.

Converting -23.0 into the floating-point number uses the same steps as converting 23.0 into the floating-point number, except that the sign bit is 1.

1 100**0001**1 011**1000**0000**0000**0000**0000**$_2$=C1B80000$_{16}$

2.2.2.2 Decimal Floating-point Numbers

- Single-precision floating-point numbers and double-precision floating-point numbers can be converted into decimal floating-point numbers so people can read them. However, internally the PLC uses single-precision floating-point numbers and double-precision floating-point numbers.
- A 32-bit decimal floating-point number is represented by two consecutive registers. The constant is stored in the first register whose number is smaller while the exponent is stored in the register whose number is bigger. Take (D1, D0) for example.

$$\text{Decimal floating-point number} = [\text{Constant D0}] * 10^{[\text{Exponent D1}]}$$

Base number D0=±1,000 to ±9,999

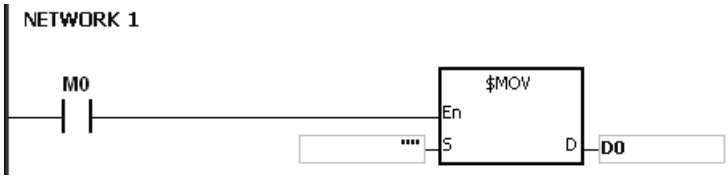
Exponent D1=-41 to +35

The base number 100 does not exist in D0 because 100 is represented by 1,000×10^{-1}. 32-bit decimal floating-point numbers range between ±1175×10^{-41} to ±402×10^{+35}.

2.2.3 Strings

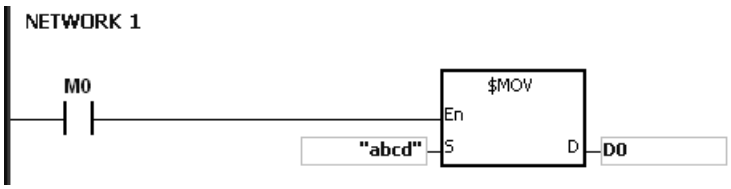
The PLC can process strings composed of ASCII codes (*1). A complete string begins with a start character, and ends with an ending character (NULL code). Strings can have maximum length of 31 characters, and ISPSoft automatically adds the ending character (16#00).

1. No string (NULL code) is moved.



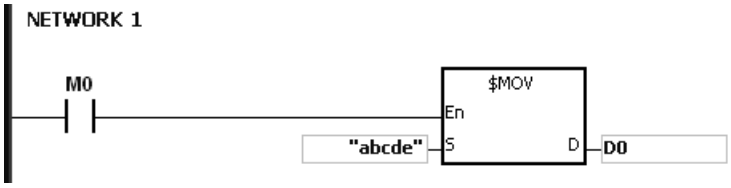
D0=0 (NULL)

2. The string has an even number of characters.



D0	16#62 (b)	16#61 (a)
D1	16#64 (d)	16#63 (b)
D2	0 (NULL)	

3. The string has an odd number of characters.



D0	16#62 (b)	16#61 (a)
D1	16#64 (d)	16#63 (b)
D2	0 (NULL)	16#65 (e)

*1: ASCII code chart

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Hex	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
ASCII	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Hex	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
ASCII	SP	!	"	#	$	%	&	'	(	)	*	+	·	-	.	/
Hex	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
ASCII	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
Hex	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
ASCII	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Hex	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
ASCII	P	Q	R	S	T	U	V	W	X	Y	Z	☒	☒	☒	☒	☒
Hex	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
ASCII	`	a	b	c	d	e	f	g	h	i	j	k	l	M	n	o
Hex	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
ASCII	p	q	r	s	t	u	v	w	x	y	z	{	|	}	–	☒

Note: ☒ represents an invisible character. Do not use it in strings.

2.2.4 Input Relays (X)

- Input function

  The input is connected to the input device (external devices such as button switches, rotary switches, and number switches), and the PLC reads the input signal. You can use input contact A or contact B several times in the program, and the ON/OFF state of the input varies with the ON/OFF state of the input device.

- Input number (the octal number)

  For the PLC, the input numbers start from X0. The number of inputs varies with the number of inputs on the digital input/output modules. The inputs are numbered according to the order in which the digital input/output modules are connected to the CPU module. The maximum number of inputs for the PLC is 256, and the input number range is between X0 to X377.

- Input type

  Inputs are classified into two types.

  1. Regenerated inputs: The PLC reads the state of a regenerated input before the program is executed; for example, LD X0.
  2. Direct input: The state of a direct input is read by the PLC during the execution of the instructions; for example, LD DX0.

2.2.5 Output Relays (Y)

- Output function

  The output sends the ON/OFF signal to drive the load connected to the output, such as an external signal lamp, a digital display, or an electromagnetic valve. There are four types of outputs. They are relays, transistors (NPN and PNP), and TRIACs (thyristors). You can use the output contact A or contact B several times in the program. Use output Y only once in the program; otherwise, according the PLC's program-scanning function, the state of the output depends on the circuit connected to the last output Y in the program.

- Output number (the octal number)

  For the PLC, the output numbers start from Y0. The number of outputs varies with the number of outputs on the digital input/output modules. The outputs are numbered according to the order in which the digital input/output modules are connected to the PLC. The maximum number of outputs on the PLC is 256, and the range is between Y0 and Y377.

  An output that is not used as an output device can be used as a general device.

- Output types

  Outputs are classified into two types.

  1. Regenerated output: The state of a regenerated output is not written until the program executes the END instruction, according to the states of the outputs; for example, OUT Y0.
  2. Direct output: The state of a direct output is written by the PLC during the execution of the instructions, according to the states of the outputs; for example, OUT DY0.

2.2.6 Auxiliary Relays (M)

The auxiliary relay has contact A and contact B. It can be used several times in the program. You can combine the control loops by using the auxiliary relay, but you cannot drive the external load using the auxiliary relay. You can use the auxiliary relays in either of the following two ways.

1. For general use: In general use, if an electric power failure occurs when the PLC is running, the auxiliary relay resets to the OFF state. When the power is restored, the auxiliary relay remains in the OFF state.
2. For latched use: In latched use, ff an electric power failure occurs when the PLC is running, the state of the auxiliary relay is retained. When the power is restored, the relay state remains the same as before the power failure.

2.2.7 Special Auxiliary Relays (SM)

Every special auxiliary relay has its specific function. Do not use the special auxiliary relays which are not defined.

The special auxiliary relays and their functions are listed as follows. As to the SM numbers marked "*", users can refer to the additional remarks on special auxiliary relays/special data registers. "R" in the attribute column indicates that the special auxiliary relay can read the data, whereas "R/W" in the attribute column indicates that it can read and write the data. In addition, the mark "–" indicates that the status of the special auxiliary relay does not make any change. The mark "#" indicates that the system will be set according to the status of the PLC, and users can read the setting value and refer to the related manual for more information.

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM0	Error: the operation or operand exceeds the allowed range	OFF	OFF	–	N	R	OFF
SM1	Error: the operation or operand exceeds the allowed range is locked.	OFF	OFF	–	N	R	OFF
SM5	Instruction inspection error	OFF	OFF	–	N	R	OFF
SM6	Data lost in the latched area	OFF	–	–	N	R/W	OFF
SM7	Insufficient power supply (24 V)	OFF	–	–	N	R	OFF
*SM8	Watchdog timer error	OFF	–	–	N	R	OFF
SM9	System error	OFF	–	–	N	R	OFF
SM10	I/O bus error	OFF	–	–	N	R	OFF
*SM22	Clearing the error log	OFF	OFF	OFF	N	R/W	OFF
SM23	Clearing the download log	OFF	OFF	OFF	N	R/W	OFF
SM24	Clearing the state-changing log of the PLC	OFF	OFF	OFF	N	R/W	OFF
SM25	The online-editing processing flag is on when the online-editing mode starts.	OFF	–	–	N	R	OFF
SM26	The debugging mode processing flag is on when the debugging mode starts.	OFF	–	–	N	R	OFF
SM28	Error: the output point is the same as the output that the high speed instruction used	OFF	OFF	OFF	N	R/W	OFF
SM30	Error in the remote module	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM34	Incorrect password	OFF	–	–	N	R/W	OFF
*SM36	Enable saving data to the memory card. When ON, the PLC runs according to the value in SR36.	OFF	–	–	N	R/W	OFF
SM94	Change in the LED lighting control in COM1	–	–	–	H	R/W	OFF
SM95	Change in the LED lighting control in COM2	–	–	–	H	R/W	OFF
*SM96	Data is sent through COM1.	OFF	OFF	–	N	R/W	OFF
*SM97	Data is sent through COM2.	OFF	OFF	–	N	R/W	OFF
*SM98	Waiting to receive the reply through COM1	OFF	OFF	–	N	R	OFF
*SM99	Waiting to receive the reply through COM2	OFF	OFF	–	N	R	OFF
*SM100	Reception through COM1 is complete.	OFF	OFF	–	N	R/W	OFF
*SM101	Reception through COM2 is complete.	OFF	OFF	–	N	R/W	OFF
*SM102	Error during receiving data through COM1 when using the MODRW instruction or the RS instruction.	OFF	OFF	–	N	R/W	OFF
*SM103	Error during receiving data through COM2 when using the MODRW instruction or the RS instruction.	OFF	OFF	–	N	R/W	OFF
*SM104	No data received through COM1 after a specified period of time.	OFF	OFF	–	N	R/W	OFF
*SM105	No data received through COM2 after a specified period of time.	OFF	OFF	–	N	R/W	OFF
*SM106	Choice made by COM1 between the 8-bit processing mode and the 16-bit processing mode ON: 8-bit processing mode OFF: 16-bit processing mode	OFF	–	–	N	R/W	OFF
*SM107	Choice made by COM2 between the 8-bit processing mode and the 16-bit processing mode ON: 8-bit processing mode OFF: 16-bit processing mode	OFF	–	–	N	R/W	OFF
*SM204	All non-latched areas are cleared.	OFF	OFF	OFF	N	R/W	OFF
*SM205	All latched areas are cleared.	OFF	OFF	OFF	N	R/W	OFF
SM206	All output is inhibited	OFF	–	–	N	R/W	OFF
*SM209	The communication protocol of COM1 changes	OFF	OFF	OFF	N	R/W	OFF
*SM210	Choice made by COM1 between the ASCII mode and the RTU mode ON: RTU mode OFF: ASCII mode	–	–	–	H	R/W	OFF
*SM211	The communication protocol of COM1 changes	OFF	OFF	OFF	N	R/W	OFF
*SM212	Choice made by COM2 between the ASCII mode and the RTU mode ON: RTU mode OFF: ASCII mode	–	–	–	H	R/W	OFF
SM215	Running state of the PLC	OFF	ON	OFF	N	R/W	OFF
SM218	Error: real-time clock malfunction	–	–	–	N	R	OFF
SM219	Error: battery power for the real-time clock is low	–	–	–	N	R	OFF
*SM220	Calibrating the real-time clock within ±30 seconds	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM221	Enable daylight saving time (D.S.T.)	–	–	–	H	R	OFF
SM222	Calibrate system clock using NTP (ON: successful calibration ; OFF: unsuccessful calibration or no calibration)	OFF	–	–	N	R	OFF
*SM227	Once the CPU is supplied with power, it checks the number of the connected modules. (ON: enabled ; OFF: disabled)	–	–	–	Y	R/W	OFF
*SM228	Disable the data exchange among the CPU and its connected modules	OFF	–	–	N	R/W	OFF
SM270	Reversing the input direction for MPG 1 flag (X0/X1)	OFF	OFF	–	N	R/W	OFF
SM271	Reversing the input direction for MPG 2 flag (X2/X3)	OFF	OFF	–	N	R/W	OFF
SM272	Reversing the input direction for MPG 3 flag (X4/X5)	OFF	OFF	–	N	R/W	OFF
SM273	Reversing the input direction for MPG 4 flag (X6/X7)	OFF	OFF	–	N	R/W	OFF
SM274	Reversing the input direction for MPG 5 flag (X10/X11)	OFF	OFF	–	N	R/W	OFF
SM275	Reversing the input direction for MPG 6 flag (X12/X13)	OFF	OFF	–	N	R/W	OFF
SM281	Reversing the input direction for high-speed counter 1 flag	OFF	OFF	–	N	R/W	OFF
SM282	Reversing the input direction for high-speed counter 2 flag	OFF	OFF	–	N	R/W	OFF
SM283	Reversing the input direction for high-speed counter 3 flag	OFF	OFF	–	N	R/W	OFF
SM284	Reversing the input direction for high-speed counter 4 flag	OFF	OFF	–	N	R/W	OFF
SM285	Reversing the input direction for high-speed counter 5 flag	OFF	OFF	–	N	R/W	OFF
SM286	Reversing the input direction for high-speed counter 6 flag	OFF	OFF	–	N	R/W	OFF
SM287	Reversing the input direction for high-speed counter 7 flag	OFF	OFF	–	N	R/W	OFF
SM288	Reversing the input direction for high-speed counter 8 flag	OFF	OFF	–	N	R/W	OFF
SM291	Clearing the input point for high-speed counter 1 flag	OFF	OFF	–	N	R/W	OFF
SM292	Clearing the input point for high-speed counter 2 flag	OFF	OFF	–	N	R/W	OFF
SM293	Clearing the input point for high-speed counter 3 flag	OFF	OFF	–	N	R/W	OFF
SM294	Clearing the input point for high-speed counter 4 flag	OFF	OFF	–	N	R/W	OFF
SM295	Clearing the input point for high-speed counter 5 flag	OFF	OFF	–	N	R/W	OFF
SM296	Clearing the input point for high-speed counter 6 flag	OFF	OFF	–	N	R/W	OFF
SM300	Setting counting mode for HC200. HC200 counts down when SM300 is ON.	OFF	OFF	–	N	R/W	OFF
SM301	Setting counting mode for HC201. HC201 counts down when SM301 is ON.	OFF	–	–	N	R	OFF
SM302	Setting counting mode for HC202. HC202 counts down when SM302 is ON.	OFF	–	–	N	R	OFF
SM303	Setting counting mode for HC203.	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	HC203 counts down when SM303 is ON.						
SM304	Setting counting mode for HC204. HC204 counts down when SM304 is ON.	OFF	OFF	–	N	R/W	OFF
SM305	Setting counting mode for HC205. HC205 counts down when SM305 is ON.	OFF	–	–	N	R	OFF
SM306	Setting counting mode for HC206. HC206 counts down when SM306 is ON.	OFF	–	–	N	R	OFF
SM307	Setting counting mode for HC207. HC207 counts down when SM307 is ON.	OFF	–	–	N	R	OFF
SM308	Setting counting mode for HC208. HC208 counts down when SM308 is ON.	OFF	OFF	–	N	R/W	OFF
SM309	Setting counting mode for HC209. HC209 counts down when SM309 is ON.	OFF	–	–	N	R	OFF
SM310	Setting counting mode for HC210. HC210 counts down when SM310 is ON.	OFF	–	–	N	R	OFF
SM311	Setting counting mode for HC211. HC211 counts down when SM311 is ON.	OFF	–	–	N	R	OFF
SM312	Setting counting mode for HC212. HC212 counts down when SM312 is ON.	OFF	OFF	–	N	R/W	OFF
SM313	Setting counting mode for HC213. HC213 counts down when SM313 is ON.	OFF	–	–	N	R	OFF
SM314	Setting counting mode for HC214. HC214 counts down when SM314 is ON.	OFF	–	–	N	R	OFF
SM315	Setting counting mode for HC215. HC215 counts down when SM315 is ON.	OFF	–	–	N	R	OFF
SM316	Setting counting mode for HC216. HC216 counts down when SM316 is ON.	OFF	OFF	–	N	R/W	OFF
SM317	Setting counting mode for HC217. HC217 counts down when SM317 is ON.	OFF	–	–	N	R	OFF
SM318	Setting counting mode for HC218. HC218 counts down when SM318 is ON.	OFF	–	–	N	R	OFF
SM319	Setting counting mode for HC219. HC219 counts down when SM319 is ON.	OFF	–	–	N	R	OFF
SM320	Setting counting mode for HC220. HC220 counts down when SM320 is ON.	OFF	OFF	–	N	R/W	OFF
SM321	Setting counting mode for HC221. HC221 counts down when SM321 is ON.	OFF	–	–	N	R	OFF
SM322	Setting counting mode for HC222. HC222 counts down when SM322 is ON.	OFF	–	–	N	R	OFF
SM323	Setting counting mode for HC223. HC223 counts down when SM323 is ON.	OFF	–	–	N	R	OFF
SM332	Setting counting mode for HC232. HC232 counts down when SM332 is ON.	OFF	OFF	–	N	R/W	OFF
SM333	Setting counting mode for HC233.	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	HC233 counts down when SM333 is ON.						
SM334	Setting counting mode for HC234. HC234 counts down when SM334 is ON.	OFF	–	–	N	R	OFF
SM335	Setting counting mode for HC235. HC235 counts down when SM335 is ON.	OFF	–	–	N	R	OFF
SM336	Setting counting mode for HC236. HC236 counts down when SM336 is ON.	OFF	OFF	–	N	R/W	OFF
SM337	Setting counting mode for HC237. HC237 counts down when SM337 is ON.	OFF	–	–	N	R	OFF
SM338	Setting counting mode for HC238. HC238 counts down when SM338 is ON.	OFF	–	–	N	R	OFF
SM339	Setting counting mode for HC239. HC239 counts down when SM339 is ON.	OFF	–	–	N	R	OFF
SM340	Setting counting mode for HC240. HC240 counts down when SM340 is ON.	OFF	OFF	–	N	R/W	OFF
SM341	Setting counting mode for HC241. HC241 counts down when SM341 is ON.	OFF	–	–	N	R	OFF
SM342	Setting counting mode for HC242. HC242 counts down when SM342 is ON.	OFF	OFF	–	N	R/W	OFF
SM343	Setting counting mode for HC243. HC243 counts down when SM343 is ON.	OFF	–	–	N	R	OFF
SM344	Setting counting mode for HC244. HC244 counts down when SM344 is ON.	OFF	OFF	–	N	R/W	OFF
SM345	Setting counting mode for HC245. HC245 counts down when SM345 is ON.	OFF	–	–	N	R	OFF
SM346	Setting counting mode for HC246. HC246 counts down when SM346 is ON.	OFF	OFF	–	N	R/W	OFF
SM347	Setting counting mode for HC247. HC247 counts down when SM347 is ON.	OFF	–	–	N	R	OFF
SM348	Setting counting mode for HC248. HC248 counts down when SM348 is ON.	OFF	OFF	–	N	R/W	OFF
SM349	Setting counting mode for HC249. HC249 counts down when SM349 is ON.	OFF	–	–	N	R	OFF
SM350	Setting counting mode for HC250. HC250 counts down when SM350 is ON.	OFF	OFF	–	N	R/W	OFF
SM351	Setting counting mode for HC251. HC251 counts down when SM351 is ON.	OFF	–	–	N	R	OFF
SM352	Setting counting mode for HC252. HC252 counts down when SM352 is ON.	OFF	OFF	–	N	R/W	OFF
SM353	Setting counting mode for HC253. HC253 counts down when SM353 is ON.	OFF	OFF	–	N	R/W	OFF
*SM400	The flag is always ON when the CPU runs.	OFF	ON	OFF	N	R	OFF
*SM401	The flag is always OFF when the CPU runs.	OFF	OFF	ON	N	R	OFF
*SM402	The flag is ON only at the first scan.	OFF	ON	OFF	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM403	The flag is OFF only at the first scan.	OFF	OFF	ON	N	R	OFF
*SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and then OFF for 5 milliseconds	OFF	–	–	N	R	OFF
*SM405	100 millisecond clock pulse during which the pulse is ON for 50 milliseconds and then OFF for 50 milliseconds	OFF	–	–	N	R	OFF
*SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and then OFF for 100 milliseconds	OFF	–	–	N	R	OFF
*SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and then OFF for 500 milliseconds	OFF	–	–	N	R	OFF
*SM450	Memory card is present ON: the memory card is present. OFF: the memory card is not present.	–	–	–	N	R	OFF
*SM452	The data in the memory card is being accessed. ON: the data in the memory card is being accessed. OFF: the data in the memory card is not accessed.	OFF	–	–	N	R	OFF
*SM453	Error during the operation of the memory card. ON: an error occurs. OFF: no error.	OFF	–	–	N	R	OFF
SM454	Enabling/disabling the data logger. ON: enable OFF: disable	OFF	–	–	N	R/W	OFF
SM455	The data logger is currently taking samples. ON: number of records has reached its sampling limit	OFF	–	–	N	R	OFF
*SM456	Execution of data logger and the memory card. ON: execution by the values in SR902.	OFF	–	–	N	R/W	OFF
SM457	State of the sample parameters in data logger ON: the sample parameter is set.	–	–	–	N	R	OFF
SM460	Outputting Y0/axis 1 (Y0/Y1).	OFF	OFF	–	N	R	OFF
SM461	Y0/axis 1 (Y0/Y1) output is complete.	OFF	OFF	–	N	R/W	OFF
SM462	Reversing the output direction of axis 1 (Y1)	OFF	OFF	–	N	R/W	OFF
*SM463	Stopping the output of Y0/axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM464	Enabling the positive maximum value for axis 1 (Y0/Y1)	–	–	–	Y	R/W	OFF
SM465	Alarm: positive limit for axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM466	Enabling the negative maximum value for axis 1 (Y0/Y1)	–	–	–	Y	R/W	OFF
SM467	Alarm: negative limit for axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
*SM468	Enabling the S curve ramp-up/down for axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM469	Enabling fixed slope ramp-up/down for axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM470	Auto-reset for Y0/axis 1 (Y0/Y1) output is complete.	OFF	OFF	–	N	R/W	OFF
SM471	Executing an interrupt I500 when pulse output ends for axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM472	Outputting Y1.	OFF	OFF	–	N	R	OFF
SM473	Y1 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM474	Stopping the output of Y1.	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM475	Auto-reset for Y1 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM476	Output immediately stops when the instruction is disabled or stops for Y0/axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
*SM477	Output immediately stops when the instruction is disabled or stops for Y1	OFF	OFF	–	N	R/W	OFF
SM478	Change the target positon while outputting Y0/axis 1 (Y0/Y1)	OFF	OFF	–	N	R/W	OFF
SM479	Change the target positon while outputting Y1	OFF	OFF	–	N	R/W	OFF
SM480	Outputting Y2/axis 2 (Y2/Y3).	OFF	OFF	–	N	R	OFF
SM481	Y2/axis 2 (Y2/Y3) output is complete.	OFF	OFF	–	N	R/W	OFF
SM482	Reversing the output direction of axis 2 (Y3)	OFF	OFF	–	N	R/W	OFF
*SM483	Stopping the output of Y2/axis 2 (Y2/Y3)	OFF	OFF	–	N	R/W	OFF
SM484	Enabling the positive maximum value for axis 2 (Y2/Y3).	–	–	–	Y	R/W	OFF
SM485	Alarm: positive limit for axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
SM486	Enabling the negative maximum value for axis 2 (Y2/Y3) .	–	–	–	Y	R/W	OFF
SM487	Alarm: negative limit for axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
*SM488	Enabling the S curve ramp-up/down for axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
SM489	Enabling fixed slope ramp-up/down for axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
SM490	Auto-reset for Y2/axis 2 (Y2/Y3) output is complete.	OFF	OFF	–	N	R/W	OFF
SM491	Executing an interrupt I501 when pulse output ends for axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
SM492	Outputting Y3.	OFF	OFF	–	N	R	OFF
SM493	Y3 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM494	Stopping the output of Y3.	OFF	OFF	–	N	R/W	OFF
SM495	Auto-reset for Y3 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM496	The output immediately stops when the instruction is disabled or stops for Y2/axis 2 (Y2/Y3).	OFF	OFF	–	N	R/W	OFF
*SM497	The output immediately stops when the instruction is disabled or stops for Y3.	OFF	OFF	–	N	R/W	OFF
SM498	Change the target positon while outputting Y2/axis 2 (Y2/Y3)	OFF	OFF	–	N	R/W	OFF
SM499	Change the target positon while outputting Y3	OFF	OFF	–	N	R/W	OFF
SM500	Outputting Y4/axis 3 (Y4/Y5).	OFF	OFF	–	N	R	OFF
SM501	Y4/axis 3 (Y4/Y5) output is complete.	OFF	OFF	–	N	R/W	OFF
SM502	Reversing the output direction of axis 3 (Y5).	OFF	OFF	–	N	R/W	OFF
*SM503	Stopping the output of Y4/axis 3 (Y4/Y5).	OFF	OFF	–	N	R/W	OFF
SM504	Enabling the positive maximum value for axis 3 (Y4/Y5).	–	–	–	Y	R/W	OFF
SM505	Alarm: positive limit for axis 3 (Y4/Y5)	OFF	OFF	–	N	R/W	OFF
SM506	Enabling the negative maximum value for axis 3 (Y4/Y5)	–	–	–	Y	R/W	OFF
SM507	Alarm: negative limit for axis 3 (Y4/Y5)	OFF	OFF	–	N	R/W	OFF
*SM508	Enabling the S curve ramp-up/down for axis 3 (Y4/Y5)	OFF	OFF	–	N	R/W	OFF
SM509	Enabling fixed slope ramp-up/down for axis 3 (Y4/Y5)	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM510	Auto-reset for Y4/axis 3 (Y4/Y5) output is complete.	OFF	OFF	–	N	R/W	OFF
SM511	Executing an interrupt I502 when pulse output ends for axis 3 (Y4/Y5).	OFF	OFF	–	N	R/W	OFF
SM512	Outputting Y5.	OFF	OFF	–	N	R	OFF
SM513	Y5 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM514	Stopping the output of Y5.	OFF	OFF	–	N	R/W	OFF
SM515	Auto-reset for Y5 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM516	Output immediately stops when the instruction is disabled or stops for Y4/axis 3 (Y4/Y5).	OFF	OFF	–	N	R/W	OFF
*SM517	Output immediately stops when the instruction is disabled or stops for Y5.	OFF	OFF	–	N	R/W	OFF
SM518	Change the target positon while outputting Y4/axis 3 (Y4/Y5)	OFF	OFF	–	N	R/W	OFF
SM519	Change the target positon while outputting Y5	OFF	OFF	–	N	R/W	OFF
SM520	Outputting Y6/axis 4 (Y6/Y7).	OFF	OFF	–	N	R	OFF
SM521	Y6/axis 4 (Y6/Y7) output is complete.	OFF	OFF	–	N	R/W	OFF
SM522	Reversing the output direction of axis 4 (Y7) .	OFF	OFF	–	N	R/W	OFF
*SM523	Stopping the output of Y6/axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
SM524	Enabling the positive maximum value for axis 4 (Y6/Y7).	–	–	–	Y	R/W	OFF
SM525	Alarm: positive limit for axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
SM526	Enabling the negative maximum value for axis 4 (Y6/Y7).	–	–	–	Y	R/W	OFF
SM527	Alarm: negative limit for axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
*SM528	Enabling the S curve ramp-up/down for axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
SM529	Enabling fixed slope ramp-up/down for axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
SM530	Auto-reset for Y6/axis 4 (Y6/Y7) is complete.	OFF	OFF	–	N	R/W	OFF
SM531	Executing an interrupt I503 when pulse output ends for axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
SM532	Outputting Y7.	OFF	OFF	–	N	R	OFF
SM533	Y7 output is complete.	OFF	OFF	–	N	R/W	OFF
*SM534	Stopping the output of Y7.	OFF	OFF	–	N	R/W	OFF
SM535	Auto-reset for Y7 output complete.	OFF	OFF	–	N	R/W	OFF
*SM536	Output immediately stops when the instruction is disabled or stops for Y6/axis 4 (Y6/Y7).	OFF	OFF	–	N	R/W	OFF
*SM537	Output immediately stops when the instruction is disabled or stops for Y7.	OFF	OFF	–	N	R/W	OFF
SM538	Change the target positon while outputting Y6/axis 4 (Y6/Y7)	OFF	OFF	–	N	R/W	OFF
SM539	Change the target positon while outputting Y7	OFF	OFF	–	N	R/W	OFF
SM580	All outputs immediately stop when the instruction is disabled or stops.	OFF	OFF	–	N	R/W	OFF
SM585	When the single-axis parameter is modified in the position planning table, its acceleration/deceleration time in SR should also be synchronized.	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM586	Used to calculate the interval frequency of acceleration and deceleration (for axis 1-4)	OFF	OFF	–	N	R/W	OFF
*SM587	When TPO instruction uses position planning table to control the output, if the output is stopped during execution, the axis should refer to its relevant flags (e.g. SM476, SM496--SM576) for the action to stop.	OFF	OFF	–	N	R/W	OFF
SM600	Zero flag	OFF	–	–	N	R	OFF
SM601	Borrow flag	OFF	–	–	N	R	OFF
SM602	Carry flag	OFF	–	–	N	R	OFF
SM604	Setting the working mode of the SORT instruction. ON: descending order OFF: ascending order	OFF	–	–	N	R/W	OFF
SM605	Designating the working mode of the SMOV instruction.	OFF	–	–	N	R/W	OFF
SM606	8-bit or 16-bit working mode. ON: 8-bit OFF: 16-bit	OFF	–	–	N	R/W	OFF
SM607	Matrix comparison flag. ON: Comparing equivalent values. OFF: Comparing different values.	OFF	–	–	N	R/W	OFF
SM608	Matrix comparison complete. When the last bits are compared, SM608 is ON.	OFF	–	–	N	R	OFF
SM609	When SM609 is ON, the comparison starts from bit 0.	OFF	–	–	N	R	OFF
SM610	Matrix bit search flag. When the matrix search finds the matching bits or completes the search, the search stops immediately and SM610 is ON.	OFF	–	–	N	R	OFF
SM611	Matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.	OFF	–	–	N	R	OFF
SM612	Matrix pointer increasing flag. The current value of the pointer increases by one.	OFF	–	–	N	R/W	OFF
SM613	Matrix pointer clearing flag. The current value of the pointer is cleared to zero.	OFF	–	–	N	R/W	OFF
SM614	Carry flag for the matrix rotation/shift/output.	OFF	–	–	N	R	OFF
SM615	Borrow flag for the matrix shift/output.	OFF	–	–	N	R/W	OFF
SM616	Direction flag for the matrix rotation/shift. OFF: the bits are shifted leftward. ON: The bits are shifted rightward.	OFF	–	–	N	R/W	OFF
SM617	The bits with the value 0 or 1 are counted.	OFF	–	–	N	R/W	OFF
SM618	ON when the matrix counting result is 0.	OFF	–	–	N	R/W	OFF
SM619	ON when the instruction EI is executed.	OFF	OFF	–	N	R	OFF
SM620	When the results from the comparison using the instruction CMPT# are that all devices are ON, SM620 is ON.	OFF	–	–	N	R	OFF
SM621	Setting counting mode for HC0. HC0 counts down when SM621 is ON.	OFF	–	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM622	Setting counting mode for HC. HC1 counts down when SM622 is ON.	OFF	–	–	N	R/W	OFF
SM623	Setting counting mode for HC2. HC2 counts down when SM623 is ON.	OFF	–	–	N	R/W	OFF
SM624	Setting counting mode for HC3. HC3 counts down when SM624 is ON.	OFF	–	–	N	R/W	OFF
SM625	Setting counting mode for HC4. HC4 counts down when SM625 is ON.	OFF	–	–	N	R/W	OFF
SM626	Setting counting mode for HC5. HC5 counts down when SM626 is ON.	OFF	–	–	N	R/W	OFF
SM627	Setting counting mode for HC6. HC6 counts down when SM627 is ON.	OFF	–	–	N	R/W	OFF
SM628	Setting counting mode for HC7. HC7 counts down when SM628 is ON.	OFF	–	–	N	R/W	OFF
SM629	Setting counting mode for HC8. HC8 counts down when SM629 is ON.	OFF	–	–	N	R/W	OFF
SM630	Setting counting mode for HC9. HC9 counts down when SM630 is ON.	OFF	–	–	N	R/W	OFF
SM631	Setting counting mode for HC10. HC10 counts down when SM631 is ON.	OFF	–	–	N	R/W	OFF
SM632	Setting counting mode for HC11. HC11 counts down when SM632 is ON.	OFF	–	–	N	R/W	OFF
SM633	Setting counting mode for HC12. HC12 counts down when SM633 is ON.	OFF	–	–	N	R/W	OFF
SM634	Setting counting mode for HC13. HC13 counts down when SM634 is ON.	OFF	–	–	N	R/W	OFF
SM635	Setting counting mode for HC14. HC14 counts down when SM635 is ON.	OFF	–	–	N	R/W	OFF
SM636	Setting counting mode for HC15. HC15 counts down when SM636 is ON.	OFF	–	–	N	R/W	OFF
SM637	Setting counting mode for HC16. HC16 counts down when SM637 is ON.	OFF	–	–	N	R/W	OFF
SM638	Setting counting mode for HC17. HC17 counts down when SM638 is ON.	OFF	–	–	N	R/W	OFF
SM639	Setting counting mode for HC18. HC18 counts down when SM639 is ON.	OFF	–	–	N	R/W	OFF
SM640	Setting counting mode for HC19. HC19 counts down when SM640 is ON.	OFF	–	–	N	R/W	OFF
SM641	Setting counting mode for HC20. HC20 counts down when SM641 is ON.	OFF	–	–	N	R/W	OFF
SM642	Setting counting mode for HC21. HC21 counts down when SM642 is ON.	OFF	–	–	N	R/W	OFF
SM643	Setting counting mode for HC22. HC22 counts down when SM643 is ON.	OFF	–	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM644	Setting counting mode for HC23. HC23 counts down when SM644 is ON.	OFF	–	–	N	R/W	OFF
SM645	Setting counting mode for HC24. HC24 counts down when SM645 is ON.	OFF	–	–	N	R/W	OFF
SM646	Setting counting mode for HC25. HC25 counts down when SM646 is ON.	OFF	–	–	N	R/W	OFF
SM647	Setting counting mode for HC26. HC26 counts down when SM647 is ON.	OFF	–	–	N	R/W	OFF
SM648	Setting counting mode for HC27. HC27 counts down when SM648 is ON.	OFF	–	–	N	R/W	OFF
SM649	Setting counting mode for HC28. HC28 counts down when SM649 is ON.	OFF	–	–	N	R/W	OFF
SM650	Setting counting mode for HC29. HC29 counts down when SM650 is ON.	OFF	–	–	N	R/W	OFF
SM651	Setting counting mode for HC30. HC30 counts down when SM651 is ON.	OFF	–	–	N	R/W	OFF
SM652	Setting counting mode for HC31. HC31 counts down when SM652 is ON.	OFF	–	–	N	R/W	OFF
SM653	Setting counting mode for HC32. HC32 counts down when SM653 is ON.	OFF	–	–	N	R/W	OFF
SM654	Setting counting mode for HC33. HC33 counts down when SM653 is ON.	OFF	–	–	N	R/W	OFF
SM655	Setting counting mode for HC34. HC34 counts down when SM655 is ON.	OFF	–	–	N	R/W	OFF
SM656	Setting counting mode for HC35. HC35 counts down when SM656 is ON.	OFF	–	–	N	R/W	OFF
SM657	Setting counting mode for HC36. HC36 counts down when SM657 is ON.	OFF	–	–	N	R/W	OFF
SM658	Setting counting mode for HC37. HC37 counts down when SM658 is ON.	OFF	–	–	N	R/W	OFF
SM659	Setting counting mode for HC38. HC38 counts down when SM659 is ON.	OFF	–	–	N	R/W	OFF
SM660	Setting counting mode for HC39. HC39 counts down when SM660 is ON.	OFF	–	–	N	R/W	OFF
SM661	Setting counting mode for HC40. HC40 counts down when SM661 is ON.	OFF	–	–	N	R/W	OFF
SM662	Setting counting mode for HC41. HC41 counts down when SM662 is ON.	OFF	–	–	N	R/W	OFF
SM663	Setting counting mode for HC42. HC42 counts down when SM663 is ON.	OFF	–	–	N	R/W	OFF
SM664	Setting counting mode for HC43. HC43 counts down when SM664 is ON.	OFF	–	–	N	R/W	OFF
SM665	Setting counting mode for HC44. HC44 counts down when SM665 is ON.	OFF	–	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM666	Setting counting mode for HC45. HC45 counts down when SM666 is ON.	OFF	–	–	N	R/W	OFF
SM667	Setting counting mode for HC46. HC46 counts down when SM667 is ON.	OFF	–	–	N	R/W	OFF
SM668	Setting counting mode for HC47. HC47 counts down when SM668 is ON.	OFF	–	–	N	R/W	OFF
SM669	Setting counting mode for HC48. HC48 counts down when SM669 is ON.	OFF	–	–	N	R/W	OFF
SM670	Setting counting mode for HC49. HC49 counts down when SM670 is ON.	OFF	–	–	N	R/W	OFF
SM671	Setting counting mode for HC50. HC50 counts down when SM671 is ON.	OFF	–	–	N	R/W	OFF
SM672	Setting counting mode for HC51. HC51 counts down when SM672 is ON.	OFF	–	–	N	R/W	OFF
SM673	Setting counting mode for HC52. HC52 counts down when SM673 is ON.	OFF	–	–	N	R/W	OFF
SM674	Setting counting mode for HC53. HC53 counts down when SM674 is ON.	OFF	–	–	N	R/W	OFF
SM675	Setting counting mode for HC54. HC54 counts down when SM675 is ON.	OFF	–	–	N	R/W	OFF
SM676	Setting counting mode for HC55. HC55 counts down when SM676 is ON.	OFF	–	–	N	R/W	OFF
SM677	Setting counting mode for HC56. HC56 counts down when SM677 is ON.	OFF	–	–	N	R/W	OFF
SM678	Setting counting mode for HC57. HC57 counts down when SM678 is ON.	OFF	–	–	N	R/W	OFF
SM679	Setting counting mode for HC58. HC58 counts down when SM679 is ON.	OFF	–	–	N	R/W	OFF
SM680	Setting counting mode for HC59. HC59 counts down when SM680 is ON.	OFF	–	–	N	R/W	OFF
SM681	Setting counting mode for HC60. HC60 counts down when SM681 is ON.	OFF	–	–	N	R/W	OFF
SM682	Setting counting mode for HC61. HC61 counts down when SM682 is ON.	OFF	–	–	N	R/W	OFF
SM683	Setting counting mode for HC62. HC62 counts down when SM683 is ON.	OFF	–	–	N	R/W	OFF
SM684	Setting counting mode for HC63. HC63 counts down when SM684 is ON.	OFF	–	–	N	R/W	OFF
SM685	The DSCLP instruction uses floating-point operations.	OFF	–	–	N	R/W	OFF
SM686	RAMP instruction mode	OFF	–	–	N	R/W	OFF
SM687	RAMP instruction execution is complete.	OFF	–	–	N	R/W	OFF
SM688	INCD instruction execution is complete.	OFF	–	–	N	R/W	OFF
SM690	String control mode.	OFF	–	–	N	R/W	OFF
SM691	HKY instruction input mode is 16-bit.	OFF	–	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	ON: the input is the hexadecimal input OFF: A–F are function keys.						
SM692	After the execution of the HKY instruction is complete, SM692 is ON for a scan cycle.	OFF	–	–	N	R/W	OFF
SM693	After the execution of the SEGL instruction is complete, SM693 is ON for a scan cycle.	OFF	–	–	N	R/W	OFF
SM694	After the execution of the DSW instruction is complete, SM694 is ON for a scan cycle.	OFF	–	–	N	R/W	OFF
SM695	Radian/degree flag. ON: degrees OFF: Radians	OFF	–	–	N	R/W	OFF
SM749	Error occurs in the initialization of the data exchange through COM1.	OFF	–	–	N	R/W	OFF
*SM750	Data exchange through COM1 enabled by ISPSoft.	OFF	–	–	H	R/W	OFF
*SM752	Connection 1 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM753	Connection 2 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM754	Connection 3 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM755	Connection 4 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM756	Connection 5 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM757	Connection 6 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM758	Connection 7 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM759	Connection 8 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM760	Connection 9 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM761	Connection 10 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM762	Connection 11 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM763	Connection 12 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM764	Connection 13 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM765	Connection 14 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM766	Connection 15 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM767	Connection 16 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM768	Connection 17 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM769	Connection 18 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM770	Connection 19 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM771	Connection 20 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM772	Connection 21 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM773	Connection 22 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM774	Connection 23 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM775	Connection 24 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM776	Connection 25 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM777	Connection 26 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM778	Connection 27 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM779	Connection 28 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM780	Connection 29 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM781	Connection 30 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM782	Connection 31 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM783	Connection 32 for data exchange through COM1 started	OFF	–	–	H	R/W	OFF
*SM784	Successful data exchange connection 1 through COM1	OFF	–	–	N	R	OFF
*SM785	Successful data exchange connection 2 through COM1	OFF	–	–	N	R	OFF
*SM786	Successful data exchange connection 3 through COM1	OFF	–	–	N	R	OFF
*SM787	Successful data exchange connection 4 through COM1	OFF	–	–	N	R	OFF
*SM788	Successful data exchange connection 5 through COM1	OFF	–	–	N	R	OFF
*SM789	Successful data exchange connection 6 through COM1	OFF	–	–	N	R	OFF
*SM790	Successful data exchange connection 7 through COM1	OFF	–	–	N	R	OFF
*SM791	Successful data exchange connection 8 through COM1	OFF	–	–	N	R	OFF
*SM792	Successful data exchange connection 9 through COM1	OFF	–	–	N	R	OFF
*SM793	Successful data exchange connection 10 through COM1	OFF	–	–	N	R	OFF
*SM794	Successful data exchange connection 11 through COM1	OFF	–	–	N	R	OFF
*SM795	Successful data exchange connection 12 through COM1	OFF	–	–	N	R	OFF
*SM796	Successful data exchange connection 13 through COM1	OFF	–	–	N	R	OFF
*SM797	Successful data exchange connection 14 through COM1	OFF	–	–	N	R	OFF
*SM798	Successful data exchange connection 15 through COM1	OFF	–	–	N	R	OFF
*SM799	Successful data exchange connection 16 through COM1	OFF	–	–	N	R	OFF
*SM800	Successful data exchange connection 17 through COM1	OFF	–	–	N	R	OFF
*SM801	Successful data exchange connection 18 through COM1	OFF	–	–	N	R	OFF
*SM802	Successful data exchange connection 19 through COM1	OFF	–	–	N	R	OFF
*SM803	Successful data exchange connection 20 through COM1	OFF	–	–	N	R	OFF
*SM804	Successful data exchange connection 21 through COM1	OFF	–	–	N	R	OFF
*SM805	Successful data exchange connection 22 through COM1	OFF	–	–	N	R	OFF
*SM806	Successful data exchange connection 23 through COM1	OFF	–	–	N	R	OFF
*SM807	Successful data exchange connection 24 through COM1	OFF	–	–	N	R	OFF
*SM808	Successful data exchange connection 25 through COM1	OFF	–	–	N	R	OFF
*SM809	Successful data exchange connection 26 through COM1	OFF	–	–	N	R	OFF
*SM810	Successful data exchange connection 27 through COM1	OFF	–	–	N	R	OFF
*SM811	Successful data exchange connection 28 through COM1	OFF	–	–	N	R	OFF
*SM812	Successful data exchange connection 29 through COM1	OFF	–	–	N	R	OFF
*SM813	Successful data exchange connection 30 through COM1	OFF	–	–	N	R	OFF
*SM814	Successful data exchange connection 31 through COM1	OFF	–	–	N	R	OFF
*SM815	Successful data exchange connection 32 through COM1	OFF	–	–	N	R	OFF
*SM816	Error in data exchange connection 1 through COM1	OFF	–	–	N	R	OFF
*SM817	Error in data exchange connection 2 through COM1	OFF	–	–	N	R	OFF
*SM818	Error in data exchange connection 3 through COM1	OFF	–	–	N	R	OFF
*SM819	Error in data exchange connection 4 through COM1	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM820	Error in data exchange connection 5 through COM1	OFF	–	–	N	R	OFF
*SM821	Error in data exchange connection 6 through COM1	OFF	–	–	N	R	OFF
*SM822	Error in data exchange connection 7 through COM1	OFF	–	–	N	R	OFF
*SM823	Error in data exchange connection 8 through COM1	OFF	–	–	N	R	OFF
*SM824	Error in data exchange connection 9 through COM1	OFF	–	–	N	R	OFF
*SM825	Error in data exchange connection 10 through COM1	OFF	–	–	N	R	OFF
*SM826	Error in data exchange connection 11 through COM1	OFF	–	–	N	R	OFF
*SM827	Error in data exchange connection 12 through COM1	OFF	–	–	N	R	OFF
*SM828	Error in data exchange connection 13 through COM1	OFF	–	–	N	R	OFF
*SM829	Error in data exchange connection 14 through COM1	OFF	–	–	N	R	OFF
*SM830	Error in data exchange connection 15 through COM1	OFF	–	–	N	R	OFF
*SM831	Error in data exchange connection 16 through COM1	OFF	–	–	N	R	OFF
*SM832	Error in data exchange connection 17 through COM1	OFF	–	–	N	R	OFF
*SM833	Error in data exchange connection 18 through COM1	OFF	–	–	N	R	OFF
*SM834	Error in data exchange connection 19 through COM1	OFF	–	–	N	R	OFF
*SM835	Error in data exchange connection 20 through COM1	OFF	–	–	N	R	OFF
*SM836	Error in data exchange connection 21 through COM1	OFF	–	–	N	R	OFF
*SM837	Error in data exchange connection 22 through COM1	OFF	–	–	N	R	OFF
*SM838	Error in data exchange connection 23 through COM1	OFF	–	–	N	R	OFF
*SM839	Error in data exchange connection 24 through COM1	OFF	–	–	N	R	OFF
*SM840	Error in data exchange connection 25 through COM1	OFF	–	–	N	R	OFF
*SM841	Error in data exchange connection 26 through COM1	OFF	–	–	N	R	OFF
*SM842	Error in data exchange connection 27 through COM1	OFF	–	–	N	R	OFF
*SM843	Error in data exchange connection 28 through COM1	OFF	–	–	N	R	OFF
*SM844	Error in data exchange connection 29 through COM1	OFF	–	–	N	R	OFF
*SM845	Error in data exchange connection 30 through COM1	OFF	–	–	N	R	OFF
*SM846	Error in data exchange connection 31 through COM1	OFF	–	–	N	R	OFF
*SM847	Error in data exchange connection 32 through COM1	OFF	–	–	N	R	OFF
SM861	Error in the initialization of the data exchange through COM2.	OFF	–	–	N	R/W	OFF
*SM862	Data exchange through COM2 enabled by ISPSoft.	OFF	–	OFF	H	R/W	OFF
*SM864	Connection 1 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM865	Connection 2 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM866	Connection 3 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM867	Connection 4 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM868	Connection 5 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM869	Connection 6 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM870	Connection 7 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM871	Connection 8 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM872	Connection 9 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM873	Connection 10 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM874	Connection 11 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM875	Connection 12 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM876	Connection 13 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM877	Connection 14 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM878	Connection 15 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM879	Connection 16 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM880	Connection 17 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM881	Connection 18 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM882	Connection 19 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM883	Connection 20 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM884	Connection 21 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM885	Connection 22 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM886	Connection 23 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM887	Connection 24 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM888	Connection 25 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM889	Connection 26 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM890	Connection 27 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM891	Connection 28 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM892	Connection 29 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM893	Connection 30 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM894	Connection 31 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM895	Connection 32 for data exchange through COM2 started	OFF	–	–	H	R/W	OFF
*SM896	Successful data exchange connection 1 through COM2	OFF	–	–	N	R	OFF
*SM897	Successful data exchange connection 2 through COM2	OFF	–	–	N	R	OFF
*SM898	Successful data exchange connection 3 through COM2	OFF	–	–	N	R	OFF
*SM899	Successful data exchange connection 4 through COM2	OFF	–	–	N	R	OFF
*SM900	Successful data exchange connection 5 through COM2	OFF	–	–	N	R	OFF
*SM901	Successful data exchange connection 6 through COM2	OFF	–	–	N	R	OFF
*SM902	Successful data exchange connection 7 through COM2	OFF	–	–	N	R	OFF
*SM903	Successful data exchange connection 8 through COM2	OFF	–	–	N	R	OFF
*SM904	Successful data exchange connection 9 through COM2	OFF	–	–	N	R	OFF
*SM905	Successful data exchange connection 10 through COM2	OFF	–	–	N	R	OFF
*SM906	Successful data exchange connection 11 through COM2	OFF	–	–	N	R	OFF
*SM907	Successful data exchange connection 12 through COM2	OFF	–	–	N	R	OFF
*SM908	Successful data exchange connection 13 through COM2	OFF	–	–	N	R	OFF
*SM909	Successful data exchange connection 14 through COM2	OFF	–	–	N	R	OFF
*SM910	Successful data exchange connection 15 through COM2	OFF	–	–	N	R	OFF
*SM911	Successful data exchange connection 16 through COM2	OFF	–	–	N	R	OFF
*SM912	Successful data exchange connection 17 through COM2	OFF	–	–	N	R	OFF
*SM913	Successful data exchange connection 18 through COM2	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM914	Successful data exchange connection 19 through COM2	OFF	–	–	N	R	OFF
*SM915	Successful data exchange connection 20 through COM2	OFF	–	–	N	R	OFF
*SM916	Successful data exchange connection 21 through COM2	OFF	–	–	N	R	OFF
*SM917	Successful data exchange connection 22 through COM2	OFF	–	–	N	R	OFF
*SM918	Successful data exchange connection 23 through COM2	OFF	–	–	N	R	OFF
*SM919	Successful data exchange connection 24 through COM2	OFF	–	–	N	R	OFF
*SM920	Successful data exchange connection 25 through COM2	OFF	–	–	N	R	OFF
*SM921	Successful data exchange connection 26 through COM2	OFF	–	–	N	R	OFF
*SM922	Successful data exchange connection 27 through COM2	OFF	–	–	N	R	OFF
*SM923	Successful data exchange connection 28 through COM2	OFF	–	–	N	R	OFF
*SM924	Successful data exchange connection 29 through COM2	OFF	–	–	N	R	OFF
*SM925	Successful data exchange connection 30 through COM2	OFF	–	–	N	R	OFF
*SM926	Successful data exchange connection 31 through COM2	OFF	–	–	N	R	OFF
*SM927	Successful data exchange connection 32 through COM2	OFF	–	–	N	R	OFF
*SM928	Error in data exchange connection 1 through COM2	OFF	–	–	N	R	OFF
*SM929	Error in data exchange connection 2 through COM2	OFF	–	–	N	R	OFF
*SM930	Error in data exchange connection 3 through COM2	OFF	–	–	N	R	OFF
*SM931	Error in data exchange connection 4 through COM2	OFF	–	–	N	R	OFF
*SM932	Error in data exchange connection 5 through COM2	OFF	–	–	N	R	OFF
*SM933	Error in data exchange connection 6 through COM2	OFF	–	–	N	R	OFF
*SM934	Error in data exchange connection 7 through COM2	OFF	–	–	N	R	OFF
*SM935	Error in data exchange connection 8 through COM2	OFF	–	–	N	R	OFF
*SM936	Error in data exchange connection 9 through COM2	OFF	–	–	N	R	OFF
*SM937	Error in data exchange connection 10 through COM2	OFF	–	–	N	R	OFF
*SM938	Error in data exchange connection 11 through COM2	OFF	–	–	N	R	OFF
*SM939	Error in data exchange connection 12 through COM2	OFF	–	–	N	R	OFF
*SM940	Error in data exchange connection 13 through COM2	OFF	–	–	N	R	OFF
*SM941	Error in data exchange connection 14 through COM2	OFF	–	–	N	R	OFF
*SM942	Error in data exchange connection 15 through COM2	OFF	–	–	N	R	OFF
*SM943	Error in data exchange connection 16 through COM2	OFF	–	–	N	R	OFF
*SM944	Error in data exchange connection 17 through COM2	OFF	–	–	N	R	OFF
*SM945	Error in data exchange connection 18 through COM2	OFF	–	–	N	R	OFF
*SM946	Error in data exchange connection 19 through COM2	OFF	–	–	N	R	OFF
*SM947	Error in data exchange connection 20 through COM2	OFF	–	–	N	R	OFF
*SM948	Error in data exchange connection 21 through COM2	OFF	–	–	N	R	OFF
*SM949	Error in data exchange connection 22 through COM2	OFF	–	–	N	R	OFF
*SM950	Error in data exchange connection 23 through COM2	OFF	–	–	N	R	OFF
*SM951	Error in data exchange connection 24 through COM2	OFF	–	–	N	R	OFF
*SM952	Error in data exchange connection 25 through COM2	OFF	–	–	N	R	OFF
*SM953	Error in data exchange connection 26 through COM2	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM954	Error in data exchange connection 27 through COM2	OFF	–	–	N	R	OFF
*SM955	Error in data exchange connection 28 through COM2	OFF	–	–	N	R	OFF
*SM956	Error in data exchange connection 29 through COM2	OFF	–	–	N	R	OFF
*SM957	Error in data exchange connection 30 through COM2	OFF	–	–	N	R	OFF
*SM958	Error in data exchange connection 31 through COM2	OFF	–	–	N	R	OFF
*SM959	Error in data exchange connection 32 through COM2	OFF	–	–	N	R	OFF
SM1000	Ethernet setting flag; ON: the data in SR1000–SR1006 is written into the flash memory.	OFF	–	–	N	R/W	OFF
SM1001	State of the Ethernet connectivity	OFF	–	–	N	R	OFF
*SM1090	TCP connection busy.	OFF	–	–	N	R	OFF
*SM1091	UDP connection busy.	OFF	–	–	N	R	OFF
SM1100	Ethernet not connected	OFF	–	–	N	R	OFF
*SM1106	Basic Ethernet management: connection error	OFF	–	–	N	R	OFF
*SM1107	Basic Ethernet management: Basic setting error	OFF	–	–	N	R	OFF
*SM1109	Basic TCP/UDP socket management: the local port is already used.	OFF	–	–	N	R	OFF
SM1111	EtherNet/IP data exchange flag	OFF	–	–	N	R	OFF
*SM1113	Email service error	OFF	–	–	N	R	OFF
*SM1116	Trigger 1 switch for the email.	OFF	–	–	N	R	OFF
*SM1117	Trigger 1 for the email	OFF	–	–	N	R	OFF
*SM1119	ON: Trigger 1 is triggered and the email has been sent successfully.	OFF	–	–	N	R	OFF
*SM1120	ON: Trigger 1 is triggered but the email cannot be sent due to email content error.	OFF	–	–	N	R	OFF
*SM1122	ON: Trigger 1 is triggered and there is an SMTP server response timeout.	OFF	–	–	N	R	OFF
*SM1123	ON: Trigger 1 is triggered and there is an SMTP server response error.	OFF	–	–	N	R	OFF
*SM1124	ON: Trigger 1 is triggered and the size of the attachment exceeds the limit.	OFF	–	–	N	R	OFF
*SM1125	Trigger 1 is triggered and the attachment is not found.	OFF	–	–	N	R	OFF
*SM1126	Trigger 2 switch for the email.	OFF	–	–	N	R	OFF
*SM1127	Trigger 2 for the email	OFF	–	–	N	R	OFF
*SM1129	ON: Trigger 2 is triggered and the email has been sent successfully.	OFF	–	–	N	R	OFF
*SM1130	ON: Trigger 2 is triggered but the email cannot be sent due to email content error.	OFF	–	–	N	R	OFF
*SM1132	ON: Trigger 2 is triggered and there is an SMTP server response timeout.	OFF	–	–	N	R	OFF
*SM1133	ON: Trigger 2 is triggered and there is an SMTP server response error.	OFF	–	–	N	R	OFF
*SM1134	ON: Trigger 2 is triggered and the size of the attachment exceeds the limit.	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1135	ON: Trigger 2 is triggered and the attachment is not found.	OFF	–	–	N	R	OFF
*SM1136	Trigger 3 switch for the email.	OFF	–	–	N	R	OFF
*SM1137	Trigger 3 for the email	OFF	–	–	N	R	OFF
*SM1139	ON: Trigger 3 is triggered and the email has been sent successfully.	OFF	–	–	N	R	OFF
*SM1140	ON: Trigger 3 is triggered but the email cannot be sent due to email content error.	OFF	–	–	N	R	OFF
*SM1142	ON: Trigger 3 is triggered and there is an SMTP server response timeout.	OFF	–	–	N	R	OFF
*SM1143	ON: Trigger 3 is triggered and there is an SMTP server response error.	OFF	–	–	N	R	OFF
*SM1144	ON: Trigger 3 is triggered and the size of the attachment exceeds the limit.	OFF	–	–	N	R	OFF
*SM1145	ON: Trigger 3 is triggered and the attachment is not found.	OFF	–	–	N	R	OFF
*SM1146	Trigger 4 switch for the email.	OFF	–	–	N	R	OFF
*SM1147	Trigger 4 for the email	OFF	–	–	N	R	OFF
*SM1149	ON: Trigger 4 is triggered and the email has been sent successfully.	OFF	–	–	N	R	OFF
*SM1150	ON: Trigger 4 is triggered but the email cannot be sent due to email content error.	OFF	–	–	N	R	OFF
*SM1152	ON: Trigger 4 is triggered and there is an SMTP server response timeout.	OFF	–	–	N	R	OFF
*SM1153	ON: Trigger 4 is triggered and there is an SMTP server response error.	OFF	–	–	N	R	OFF
*SM1154	ON: Trigger 4 is triggered and the size of the attachment exceeds the limit.	OFF	–	–	N	R	OFF
*SM1155	ON: Trigger 4 is triggered and the attachment is not found.	OFF	–	–	N	R	OFF
*SM1166	Error in data exchange through Ethernet	–	–	–	N	R	OFF
*SM1167	Data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1168	Connection 1 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1169	Connection 2 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1170	Connection 3 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1171	Connection 4 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1172	Connection 5 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1173	Connection 6 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1174	Connection 7 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1175	Connection 8 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1176	Connection 9 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1177	Connection 10 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1178	Connection 11 for data exchange through Ethernet	OFF	–	–	H	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	started						
*SM1179	Connection 12 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1180	Connection 13 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1181	Connection 14 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1182	Connection 15 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1183	Connection 16 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1184	Connection 17 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1185	Connection 18 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1186	Connection 19 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1187	Connection 20 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1188	Connection 21 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1189	Connection 22 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1190	Connection 23 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1191	Connection 24 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1192	Connection 25 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1193	Connection 26 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1194	Connection 27 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1195	Connection 28 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1196	Connection 29 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1197	Connection 30 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1198	Connection 31 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1199	Connection 32 for data exchange through Ethernet started	OFF	–	–	H	R/W	OFF
*SM1200	Successful data exchange connection 1 through Ethernet	OFF	–	–	N	R	OFF
*SM1201	Successful data exchange connection 2 through Ethernet	OFF	–	–	N	R	OFF
*SM1202	Successful data exchange connection 3 through Ethernet	OFF	–	–	N	R	OFF
*SM1203	Successful data exchange connection 4 through Ethernet	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1204	Successful data exchange connection 5 through Ethernet	OFF	–	–	N	R	OFF
*SM1205	Successful data exchange connection 6 through Ethernet	OFF	–	–	N	R	OFF
*SM1206	Successful data exchange connection 7 through Ethernet	OFF	–	–	N	R	OFF
*SM1207	Successful data exchange connection 8 through Ethernet	OFF	–	–	N	R	OFF
*SM1208	Successful data exchange connection 9 through Ethernet	OFF	–	–	N	R	OFF
*SM1209	Successful data exchange connection 10 through Ethernet	OFF	–	–	N	R	OFF
*SM1210	Successful data exchange connection 11 through Ethernet	OFF	–	–	N	R	OFF
*SM1211	Successful data exchange connection 12 through Ethernet	OFF	–	–	N	R	OFF
*SM1212	Successful data exchange connection 13 through Ethernet	OFF	–	–	N	R	OFF
*SM1213	Successful data exchange connection 14 through Ethernet	OFF	–	–	N	R	OFF
*SM1214	Successful data exchange connection 15 through Ethernet	OFF	–	–	N	R	OFF
*SM1215	Successful data exchange connection 16 through Ethernet	OFF	–	–	N	R	OFF
*SM1216	Successful data exchange connection 17 through Ethernet	OFF	–	–	N	R	OFF
*SM1217	Successful data exchange connection 18 through Ethernet	OFF	–	–	N	R	OFF
*SM1218	Successful data exchange connection 19 through Ethernet	OFF	–	–	N	R	OFF
*SM1219	Successful data exchange connection 20 through Ethernet	OFF	–	–	N	R	OFF
*SM1220	Successful data exchange connection 21 through Ethernet	OFF	–	–	N	R	OFF
*SM1221	Successful data exchange connection 22 through Ethernet	OFF	–	–	N	R	OFF
*SM1222	Successful data exchange connection 23 through Ethernet	OFF	–	–	N	R	OFF
*SM1223	Successful data exchange connection 24 through Ethernet	OFF	–	–	N	R	OFF
*SM1224	Successful data exchange connection 25 through Ethernet	OFF	–	–	N	R	OFF
*SM1225	Successful data exchange connection 26 through Ethernet	OFF	–	–	N	R	OFF
*SM1226	Successful data exchange connection 27 through Ethernet	OFF	–	–	N	R	OFF
*SM1227	Successful data exchange connection 28 through Ethernet	OFF	–	–	N	R	OFF
*SM1228	Successful data exchange connection 29 through Ethernet	OFF	–	–	N	R	OFF
*SM1229	Successful data exchange connection 30 through Ethernet	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SM1230	Successful data exchange connection 31 through Ethernet	OFF	–	–	N	R	OFF
*SM1231	Successful data exchange connection 32 through Ethernet	OFF	–	–	N	R	OFF
*SM1232	Error in data exchange connection 1 through Ethernet	OFF	–	–	N	R	OFF
*SM1233	Error in data exchange connection 2 through Ethernet	OFF	–	–	N	R	OFF
*SM1234	Error in data exchange connection 3 through Ethernet	OFF	–	–	N	R	OFF
*SM1235	Error in data exchange connection 4 through Ethernet	OFF	–	–	N	R	OFF
*SM1236	Error in data exchange connection 5 through Ethernet	OFF	–	–	N	R	OFF
*SM1237	Error in data exchange connection 6 through Ethernet	OFF	–	–	N	R	OFF
*SM1238	Error in data exchange connection 7 through Ethernet	OFF	–	–	N	R	OFF
*SM1239	Error in data exchange connection 8 through Ethernet	OFF	–	–	N	R	OFF
*SM1240	Error in data exchange connection 9 through Ethernet	OFF	–	–	N	R	OFF
*SM1241	Error in data exchange connection 10 through Ethernet	OFF	–	–	N	R	OFF
*SM1242	Error in data exchange connection 11 through Ethernet	OFF	–	–	N	R	OFF
*SM1243	Error in data exchange connection 12 through Ethernet	OFF	–	–	N	R	OFF
*SM1244	Error in data exchange connection 13 through Ethernet	OFF	–	–	N	R	OFF
*SM1245	Error in data exchange connection 14 through Ethernet	OFF	–	–	N	R	OFF
*SM1246	Error in data exchange connection 15 through Ethernet	OFF	–	–	N	R	OFF
*SM1247	Error in data exchange connection 16 through Ethernet	OFF	–	–	N	R	OFF
*SM1248	Error in data exchange connection 17 through Ethernet	OFF	–	–	N	R	OFF
*SM1249	Error in data exchange connection 18 through Ethernet	OFF	–	–	N	R	OFF
*SM1250	Error in data exchange connection 19 through Ethernet	OFF	–	–	N	R	OFF
*SM1251	Error in data exchange connection 20 through Ethernet	OFF	–	–	N	R	OFF
*SM1252	Error in data exchange connection 21 through Ethernet	OFF	–	–	N	R	OFF
*SM1253	Error in data exchange connection 22 through Ethernet	OFF	–	–	N	R	OFF
*SM1254	Error in data exchange connection 23 through Ethernet	OFF	–	–	N	R	OFF
*SM1255	Error in data exchange connection 24 through Ethernet	OFF	–	–	N	R	OFF
*SM1256	Error in data exchange connection 25 through Ethernet	OFF	–	–	N	R	OFF
*SM1257	Error in data exchange connection 26 through Ethernet	OFF	–	–	N	R	OFF
*SM1258	Error in data exchange connection 27 through Ethernet	OFF	–	–	N	R	OFF
*SM1259	Error in data exchange connection 28 through Ethernet	OFF	–	–	N	R	OFF
*SM1260	Error in data exchange connection 29 through Ethernet	OFF	–	–	N	R	OFF
*SM1261	Error in data exchange connection 30 through Ethernet	OFF	–	–	N	R	OFF
*SM1262	Error in data exchange connection 31 through Ethernet	OFF	–	–	N	R	OFF
*SM1263	Error in data exchange connection 32 through Ethernet	OFF	–	–	N	R	OFF
SM1269	Socket configuration error	OFF	–	–	N	R/W	OFF
SM1270	Successful TCP socket 1 connection	OFF	–	–	N	R	OFF
SM1271	TCP socket 1 data received	OFF	–	–	N	R	OFF
SM1272	TCP socket 1 data sent	OFF	–	–	N	R	OFF
SM1273	TCP socket 1 connection starting	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1274	TCP socket 1 connection closing	ON	–	–	Y	R	ON
SM1275	TCP socket 1 data is being sent	OFF	–	–	N	R	OFF
SM1277	TCP socket 1─Error flag	OFF	–	–	N	R	OFF
SM1278	Successful TCP socket 2 connection	OFF	–	–	N	R	OFF
SM1279	TCP socket 2 data received	OFF	–	–	N	R	OFF
SM1280	TCP socket 2 data sent	OFF	–	–	N	R	OFF
SM1281	TCP socket 2 connection starting	OFF	–	–	N	R	OFF
SM1282	TCP socket 2 connection closing	ON	–	–	Y	R	ON
SM1283	TCP socket 2 data is being sent	OFF	–	–	N	R	OFF
SM1285	TCP socket 2─Error flag	OFF	–	–	N	R	OFF
SM1286	Successful TCP socket 3 connection	OFF	–	–	N	R	OFF
SM1287	TCP socket 3 data received	OFF	–	–	N	R	OFF
SM1288	TCP socket 3 data sent	OFF	–	–	N	R	OFF
SM1289	TCP socket 3 connection starting	OFF	–	–	N	R	OFF
SM1290	TCP socket 3 connection closing	ON	–	–	Y	R	ON
SM1291	TCP socket 3 data is being sent.	OFF	–	–	N	R	OFF
SM1293	TCP socket 3─Error flag	OFF	–	–	N	R	OFF
SM1294	Successful TCP socket 4 connection	OFF	–	–	N	R	OFF
SM1295	TCP socket 4 data received	OFF	–	–	N	R	OFF
SM1296	TCP socket 4 data sent	OFF	–	–	N	R	OFF
SM1297	TCP socket 4 connection starting	OFF	–	–	N	R	OFF
SM1298	TCP socket 4 connection closing	ON	–	–	Y	R	ON
SM1299	TCP socket 4 data is being sent.	OFF	–	–	N	R	OFF
SM1301	TCP socket 4─Error flag	OFF	–	–	N	R	OFF
SM1312	RTU-EN01 connection 1 status (ON: connected; OFF: disconnected or not connected)	OFF	–	–	N	R	OFF
SM1313	RTU-EN01 connection 2 status	OFF	–	–	N	R	OFF
SM1314	RTU-EN01 connection 3 status	OFF	–	–	N	R	OFF
SM1315	RTU-EN01 connection 4 status	OFF	–	–	N	R	OFF
SM1334	UDP socket 1 connection started	OFF	–	–	N	R	OFF
SM1335	UDP socket 1 data receiving	OFF	–	–	N	R	OFF
SM1336	UDP socket 1 data sending	OFF	–	–	N	R	OFF
SM1337	UDP socket 1 connection closed	ON	–	–	Y	R	ON
SM1338	UDP socket 1─Error flag	OFF	–	–	N	R	OFF
SM1339	UDP socket 2 connection started	OFF	–	–	N	R	OFF
SM1340	UDP socket 2 data receiving	OFF	–	–	N	R	OFF
SM1341	UDP socket 2 data sending	OFF	–	–	N	R	OFF
SM1342	UDP socket 2 connection closed	ON	–	–	Y	R	ON
SM1343	UDP socket 2─Error flag	OFF	–	–	N	R	OFF
SM1344	UDP socket 3 connection started	OFF	–	–	N	R	OFF
SM1345	UDP socket 3 data receiving	OFF	–	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1346	UDP socket 3 data sending	OFF	–	–	N	R	OFF
SM1347	UDP socket 3 connection closed	ON	–	–	Y	R	ON
SM1348	UDP socket 3─Error flag	OFF	–	–	N	R	OFF
SM1349	UDP socket 4 connection started	OFF	–	–	N	R	OFF
SM1350	UDP socket 4 data receiving	OFF	–	–	N	R	OFF
SM1351	UDP socket 4 data sending	OFF	–	–	N	R	OFF
SM1352	UDP socket 4 connection closed	ON	–	–	Y	R	ON
SM1353	UDP socket 4─Error flag	OFF	–	–	N	R	OFF
SM1440	Error in I/O connection 1 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1441	Error in I/O connection 2 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1442	Error in I/O connection 3 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1443	Error in I/O connection 4 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1444	Error in I/O connection 5 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1445	Error in I/O connection 6 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1446	Error in I/O connection 7 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1447	Error in I/O connection 8 through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1448	I/O connection 1 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1449	I/O connection 2 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1450	I/O connection 3 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1451	I/O connection 4 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1452	I/O connection 5 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1453	I/O connection 6 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1454	I/O connection 7 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
*SM1455	I/O connection 8 is established through EtherNet/IP (Adapter)	OFF	-	-	N	R	OFF
SM1581	Refresh and then release the control over the connection of Delta servo CAN axis 1	OFF	OFF	–	N	R/W	OFF
SM1582	Refresh and then release the control over the connection of Delta servo CAN axis 2	OFF	OFF	–	N	R/W	OFF
SM1583	Refresh and then release the control over the connection of Delta servo CAN axis 3	OFF	OFF	–	N	R/W	OFF
SM1584	Refresh and then release the control over the connection of Delta servo CAN axis 4	OFF	OFF	–	N	R/W	OFF
SM1585	Refresh and then release the control over the connection of Delta servo CAN axis 5	OFF	OFF	–	N	R/W	OFF
SM1586	Refresh and then release the control over the connection of Delta servo CAN axis 6	OFF	OFF	–	N	R/W	OFF
SM1587	Refresh and then release the control over the connection of Delta servo CAN axis 7	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1588	Refresh and then release the control over the connection of Delta servo CAN axis 8	OFF	OFF	–	N	R/W	OFF
SM1601	Refresh and then release the control over the connection of Delta motor CAN axis 21	OFF	OFF	–	N	R/W	OFF
SM1602	Refresh and then release the control over the connection of Delta motor CAN axis 22	OFF	OFF	–	N	R/W	OFF
SM1603	Refresh and then release the control over the connection of Delta motor CAN axis 23	OFF	OFF	–	N	R/W	OFF
SM1604	Refresh and then release the control over the connection of Delta motor CAN axis 24	OFF	OFF	–	N	R/W	OFF
SM1605	Refresh and then release the control over the connection of Delta motor CAN axis 25	OFF	OFF	–	N	R/W	OFF
SM1606	Refresh and then release the control over the connection of Delta motor CAN axis 26	OFF	OFF	–	N	R/W	OFF
SM1607	Refresh and then release the control over the connection of Delta motor CAN axis 27	OFF	OFF	–	N	R/W	OFF
SM1608	Refresh and then release the control over the connection of Delta motor CAN axis 28	OFF	OFF	–	N	R/W	OFF
SM1611	Heartbeat error of Delta motor CAN communication ID 21	OFF	OFF	–	N	R	OFF
SM1612	Heartbeat error of Delta motor CAN communication ID 22	OFF	OFF	–	N	R	OFF
SM1613	Heartbeat error of Delta motor CAN communication ID 23	OFF	OFF	–	N	R	OFF
SM1614	Heartbeat error of Delta motor CAN communication ID 24	OFF	OFF	–	N	R	OFF
SM1615	Heartbeat error of Delta motor CAN communication ID 25	OFF	OFF	–	N	R	OFF
SM1616	Heartbeat error of Delta motor CAN communication ID 26	OFF	OFF	–	N	R	OFF
SM1617	Heartbeat error of Delta motor CAN communication ID 27	OFF	OFF	–	N	R	OFF
SM1618	Heartbeat error of Delta motor CAN communication ID 28	OFF	OFF	–	N	R	OFF
SM1621	Delta motor CAN communication ID 21 is starting	OFF	OFF	–	N	R	OFF
SM1622	Delta motor CAN communication ID 22 is starting	OFF	OFF	–	N	R	OFF
SM1623	Delta motor CAN communication ID 23 is starting	OFF	OFF	–	N	R	OFF
SM1624	Delta motor CAN communication ID 24 is starting	OFF	OFF	–	N	R	OFF
SM1625	Delta motor CAN communication ID 25 is starting	OFF	OFF	–	N	R	OFF
SM1626	Delta motor CAN communication ID 26 is starting	OFF	OFF	–	N	R	OFF
SM1627	Delta motor CAN communication ID 27 is starting	OFF	OFF	–	N	R	OFF
SM1628	Delta motor CAN communication ID 28 is starting	OFF	OFF	–	N	R	OFF
SM1631	Positioning completed for Delta servo CAN axis 1	OFF	OFF	–	N	R/W	OFF
SM1632	Positioning completed for Delta servo CAN axis 2	OFF	OFF	–	N	R/W	OFF
SM1633	Positioning completed for Delta servo CAN axis 3	OFF	OFF	–	N	R/W	OFF
SM1634	Positioning completed for Delta servo CAN axis 4	OFF	OFF	–	N	R/W	OFF
SM1635	Positioning completed for Delta servo CAN axis 5	OFF	OFF	–	N	R/W	OFF
SM1636	Positioning completed for Delta servo CAN axis 6	OFF	OFF	–	N	R/W	OFF
SM1637	Positioning completed for Delta servo CAN axis 7	OFF	OFF	–	N	R/W	OFF
SM1638	Positioning completed for Delta servo CAN axis 8	OFF	OFF	–	N	R/W	OFF
SM1641	Communication stops for Delta servo CAN axis 1	OFF	OFF	–	N	R/W	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1642	Communication stops for Delta servo CAN axis 2	OFF	OFF	–	N	R/W	OFF
SM1643	Communication stops for Delta servo CAN axis 3	OFF	OFF	–	N	R/W	OFF
SM1644	Communication stops for Delta servo CAN axis 4	OFF	OFF	–	N	R/W	OFF
SM1645	Communication stops for Delta servo CAN axis 5	OFF	OFF	–	N	R/W	OFF
SM1646	Communication stops for Delta servo CAN axis 6	OFF	OFF	–	N	R/W	OFF
SM1647	Communication stops for Delta servo CAN axis 7	OFF	OFF	–	N	R/W	OFF
SM1648	Communication stops for Delta servo CAN axis 8	OFF	OFF	–	N	R/W	OFF
SM1651	Servo is ON for Delta servo CAN axis 1	OFF	OFF	–	N	R	OFF
SM1652	Servo is ON for Delta servo CAN axis 2	OFF	OFF	–	N	R	OFF
SM1653	Servo is ON for Delta servo CAN axis 3	OFF	OFF	–	N	R	OFF
SM1654	Servo is ON for Delta servo CAN axis 4	OFF	OFF	–	N	R	OFF
SM1655	Servo is ON for Delta servo CAN axis 5	OFF	OFF	–	N	R	OFF
SM1656	Servo is ON for Delta servo CAN axis 6	OFF	OFF	–	N	R	OFF
SM1657	Servo is ON for Delta servo CAN axis 7	OFF	OFF	–	N	R	OFF
SM1658	Servo is ON for Delta servo CAN axis 8	OFF	OFF	–	N	R	OFF
SM1661	The function of going back and forth is enabled for Delta servo CAN axis 1. See the DDRVAC instruction (API 2804).	OFF	OFF	–	N	R/W	OFF
SM1662	The function of going back and forth is enabled for Delta servo CAN axis 2. See the DDRVAC instruction (API 2804).	OFF	OFF	–	N	R/W	OFF
SM1663	The function of going back and forth is enabled for Delta servo CAN axis 3. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1664	The function of going back and forth is enabled for Delta servo CAN axis 4. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1665	The function of going back and forth is enabled for Delta servo CAN axis 5. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1666	The function of going back and forth is enabled for Delta servo CAN axis 6. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1667	The function of going back and forth is enabled for Delta servo CAN axis 7. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1668	The function of going back and forth is enabled for Delta servo CAN axis 8. See the DDRVAC instruction (API 2804)	OFF	OFF	–	N	R/W	OFF
SM1671	The go-back/go-forth direction indication flag for Delta servo CAN axis 1	OFF	OFF	–	N	R	OFF
SM1672	The go-back/go-forth direction indication flag for Delta servo CAN axis 2	OFF	OFF	–	N	R	OFF
SM1673	The go-back/go-forth direction indication flag for Delta servo CAN axis 3	OFF	OFF	–	N	R	OFF

SM	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SM1674	The go-back/go-forth direction indication flag for Delta servo CAN axis 4	OFF	OFF	–	N	R	OFF
SM1675	The go-back/go-forth direction indication flag for Delta servo CAN axis 5	OFF	OFF	–	N	R	OFF
SM1676	The go-back/go-forth direction indication flag for Delta servo CAN axis 6	OFF	OFF	–	N	R	OFF
SM1677	The go-back/go-forth direction indication flag for Delta servo CAN axis 7	OFF	OFF	–	N	R	OFF
SM1678	The go-back/go-forth direction indication flag for Delta servo CAN axis 8	OFF	OFF	–	N	R	OFF
SM1681	Initialization and CAN communication (INITC and CASD) completed for Delta servo	OFF	OFF	–	N	R	OFF
SM1682	The CAN communication error flag for Delta servo	OFF	OFF	–	N	R	OFF
SM1683	Initialization and CAN communication completed for Delta motor	OFF	OFF	–	N	R	OFF
SM1684	To set the Delta special CAN communication ON/OFF when a connection is lost OFF: stops all the communications ON: stops only the lost-connection one	OFF	OFF	–	N	R/W	OFF
SM1685	To read the self-defined Delta servo parameters P0-12	OFF	OFF	–	N	R/W	OFF
SM1691	Heartbeat error of Delta special CAN axis 1	OFF	OFF	–	N	R	OFF
SM1692	Heartbeat error of Delta special CAN axis 2	OFF	OFF	–	N	R	OFF
SM1693	Heartbeat error of Delta special CAN axis 3	OFF	OFF	–	N	R	OFF
SM1694	Heartbeat error of Delta special CAN axis 4	OFF	OFF	–	N	R	OFF
SM1695	Heartbeat error of Delta special CAN axis 5	OFF	OFF	–	N	R	OFF
SM1696	Heartbeat error of Delta special CAN axis 6	OFF	OFF	–	N	R	OFF
SM1697	Heartbeat error of Delta special CAN axis 7	OFF	OFF	–	N	R	OFF
SM1698	Heartbeat error of Delta special CAN axis 8	OFF	OFF	–	N	R	OFF

*1: For items with a * mark, Refer to Section 2.2.1.6 for Additional Remarks on Special Auxiliary Relays and Special Data Registers for details.

*2: The system executes according to the parameters set in HWCONFIG. When the SM power changes from OFF to ON, the state is -, and the latched area is marked as N.

*3: The communication card here means AS-F232, AS-F422 and AS-F485.

2.2.8 Refresh Time for Special Auxiliary Relays

Special auxiliary relay	Refresh time
SM0–SM1	The system automatically sets the flag to ON and resets it to OFF. ON: operation error.
SM5	The system automatically sets the flag to ON and resets it to OFF. ON: an error occurs when the program is written in the PLC.
SM6	During power-on, the system checks whether the data in the latched area has been lost. ON: data in the latched area has been lost. You reset it to OFF.

Special auxiliary relay	Refresh time
SM7	ON: power supply (24 V) is not sufficient. You reset it to OFF.
SM8	The system automatically sets SM8 to ON and resets it to OFF. ON: there is a watchdog timer error.
SM9	The system automatically sets SM9 to ON and resets it to OFF. ON: there is a system error.
SM10	The system automatically sets SM10 to ON and resets it to OFF. ON: there is an I/O bus error.
SM22–SM24	You set the flag to ON, and the system automatically resets it to OFF. ON: the log is cleared.
SM25–SM26	ON: users are editing with ISPSoft. OFF: users have logged out of ISPSoft.
SM28	The system checks for anything wrong. ON: something is wrong. You reset it to OFF.
SM30	ON: an error occurs in the remote module. The system resets it to OFF.
SM34	ON: the wrong password is entered. The system resets it to OFF.
SM36	ON: the system saves the data to the memory card. After saving is complete, the system resets it to OFF automatically. You set it to ON to enable saving.
SM94–SM95	After power-on, the flag is ON or OFF according to the settings in HWCONFIG. You can change this setting.
SM96–SM97	You set the flag to ON. After the data is sent, the system automatically resets the flag to OFF.
SM98–SM99	ON: communication is in process. After the communication is complete, the system resets it to OFF automatically.
SM100–SM101	The system automatically sets the flag to ON, and you reset it to OFF. ON: the command is received.
SM102–SM103	The system automatically sets the flag to ON, and you reset it to OFF. ON: the command received is wrong.
SM104–SM105	The system automatically sets the flag to ON, and you reset it to OFF. ON: there is a receive timeout.
SM106–SM107	You set the flag to ON and reset it to OFF. ON: 8-bit mode OFF: 16-bit mode
SM204–SM205	You set the flag to ON, and the system automatically resets it to OFF. ON: clear the non-latched/latched areas.
SM206	You set SM206 to ON and reset it to OFF. ON: inhibit all output.
SM209	You set SM209 to ON, and the system automatically resets it to OFF. ON: the communication protocol of COM1 changes.
SM210	You set SM210 to ON and reset it to OFF for COM1. ON: RTU mode OFF: ASCII mode
SM211	You set SM211 to ON, and the system automatically resets it to OFF. ON: the communication protocol of COM2 changes.
SM212	You set SM210 to ON and reset it to OFF for COM2. ON: RTU mode OFF: ASCII mode
SM215	You set SM215 to ON and reset it to OFF. ON: the PLC runs.

Special auxiliary relay	Refresh time
	OFF: the PLC stops.
SM218	The system checks the real-time clock at power-on. ON: real-time clock error You reset it to OFF.
SM219	The system monitors the battery power of the real-time clock. ON: real-time clock power is low The system resets it to OFF.
SM220	You set SM220 to ON and reset it to OFF. ON: calibrating the real-time clock within ±30 seconds
SM221	The flag is refreshed according to the settings in HWCONFIG or when the DST instruction (API 1607) is executed. ON: the DST instruction is executed.
SM222	The flag is refreshed during calibration
SM227~SM229	The flag is refreshed when the CPU is supplied with power
SM270–SM275	The flag is ON when the CSFO instruction is executed. ON: enable reversing the input direction OFF: disable reversing the input direction
SM281–SM288	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM291–SM296	You set the flag to ON and reset it to OFF. ON: enable clearing the input points OFF: disable clearing the input points
SM300	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM301–SM303	The system sets the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM304	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM305–SM307	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM308	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM309–SM311	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM312	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM313–SM315	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM316	You set the flag to ON and reset it to OFF.

Special auxiliary relay	Refresh time
	ON: counting down OFF: counting up
SM317–SM319	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM320	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM321–SM323	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM332	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM333–SM335	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM336	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM337–SM339	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM340	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM341	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM342	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM343	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM344	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM345	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM346	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM347	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM348	You set the flag to ON and reset it to OFF.

Special auxiliary relay	Refresh time
	ON: counting down OFF: counting up
SM349	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM350	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM351	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM352–SM353	You set the flag to ON and reset it to OFF. ON: counting down OFF: counting up
SM400–SM403	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed every scan cycle.
SM404	The system automatically sets the flag to ON and resets it to OFF. SM404 is refreshed every 5 milliseconds.
SM405	The system automatically sets SM405 to ON and resets it to OFF. SM405 is refreshed every 50 milliseconds.
SM406	The system automatically sets SM406 to ON and resets it to OFF. SM406 is refreshed every 100 milliseconds.
SM407	The system automatically sets SM407 to ON and resets it to OFF. SM407 is refreshed every 0.5 seconds.
SM450	The system automatically sets SM450 to ON and resets it to OFF. ON: memory card is inserted into the PLC. OFF: memory card is removed out of the PLC.
SM452–SM453	The system sets the flag to ON or OFF.
SM454	You set the flag to ON or OFF.
SM455	The system sets the flag to ON or OFF.
SM456	You set the flag to ON to save. After the saving is complete, the system resets it to OFF.
SM457	The system sets the flag to ON or OFF.
SM460	The system sets the flag to ON or OFF.
SM461	The system sets the flag to ON and you reset it to OFF.
SM462–SM464	You set the flag to ON or OFF.
SM465	The system sets the flag to ON and you reset it to OFF.
SM466	You set the flag to ON or OFF.
SM467	The system sets the flag to ON and you reset it to OFF.
SM468–SM471	You set the flag to ON or OFF.
SM472	The system sets the flag to ON or OFF.
SM473	The system sets the flag to ON and you reset it to OFF.
SM474–SM477	You set the flag to ON or OFF.
SM478–SM479	You set the flag to ON or OFF and the system sets the flag to OFF.
SM480	The system sets the flag to ON or OFF.
SM481	The system sets the flag to ON and you reset it to OFF.
SM482–SM484	You set the flag to ON or OFF.
SM485	The system sets the flag to ON and you reset it to OFF.

Special auxiliary relay	Refresh time
SM486	You set the flag to ON or OFF.
SM487	The system sets the flag to ON and you reset it to OFF.
SM488–SM491	You set the flag to ON or OFF.
SM492	The system sets the flag to ON or OFF.
SM493	The system sets the flag to ON and you reset it to OFF.
SM494–SM497	You set the flag to ON or OFF.
SM498–SM499	You set the flag to ON or OFF and the system sets the flag to OFF.
SM500	The system sets the flag to ON or OFF.
SM501	The system sets the flag to ON and you reset it to OFF.
SM502–SM504	You set the flag to ON or OFF.
SM505	The system sets the flag to ON and you reset it to OFF.
SM506	You set the flag to ON or OFF.
SM507	The system sets the flag to ON and you reset it to OFF.
SM508–SM511	You set the flag to ON or OFF.
SM512	The system sets the flag to ON or OFF.
SM513	The system sets the flag to ON and you reset it to OFF.
SM514–SM517	You set the flag to ON or OFF.
SM518–SM519	You set the flag to ON or OFF and the system sets the flag to OFF.
SM520	The system sets the flag to ON or OFF.
SM521	The system sets the flag to ON and you reset it to OFF.
SM522–SM524	You set the flag to ON or OFF.
SM525	The system sets the flag to ON and you reset it to OFF.
SM526	You set the flag to ON or OFF.
SM527	The system sets the flag to ON and you reset it to OFF.
SM528–SM531	You set the flag to ON or OFF.
SM532	The system sets the flag to ON or OFF.
SM533	The system sets the flag to ON and you reset it to OFF.
SM534–SM537	You set the flag to ON or OFF.
SM538–SM539	You set the flag to ON or OFF and the system sets the flag to OFF.
SM580	You set the flag to ON and the system resets it to OFF. ON: disable high-speed output
SM585	You set the flag to ON or OFF and the system sets the flag to OFF.
SM586--SM587	You set the flag to ON or OFF.
SM600–SM602	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed when the instruction is executed.
SM604	You set SM604 to ON and reset it to OFF. ON: sort in descending order OFF: sort in ascending order
SM605	You set SM605 to ON and reset it to OFF.
SM606	You set SM606 to ON and reset it to OFF. ON: 8-bit mode OFF: 16-bit mode
SM607	You set the flag to ON or OFF.
SM608	The flag is refreshed when the instruction is executed.
SM609	You set the flag to ON or OFF.
SM610–SM611	The flag is refreshed when the instruction is executed.

Special auxiliary relay	Refresh time
SM612–SM613	You set the flag to ON or OFF.
SM614	The flag is refreshed when the instruction is executed.
SM615–SM617	You set the flag to ON or OFF.
SM618	The flag is refreshed when the instruction is executed.
SM619	The flag is refreshed when the EI or DI instruction is executed.
SM620	The flag is refreshed when the CMPT instruction is executed.
SM621–SM686	You set the flag to ON or OFF.
SM687	The flag is refreshed when the RAMP instruction is executed.
SM688	The flag is refreshed when the INCD instruction is executed.
SM690–SM691	You set the flag to ON or OFF.
SM692	The flag is refreshed when the HKY instruction is executed.
SM693	The flag is refreshed when the SEGL instruction is executed.
SM694	The flag is refreshed when the DSW instruction is executed.
SM695	You set the flag to ON or OFF.
SM749	The system is refreshed at power-on after the data exchange parameters are downloaded.
SM750–SM783	After the data exchange parameters are downloaded, you set the flag to ON or OFF.
SM784–SM847	The flag is ON when the system is refreshed.
SM861	The system is refreshed at power-on after the data exchange parameters are downloaded.
SM862–SM895	After the parameters of data exchange are downloaded, you set the flag to ON or OFF.
SM896–SM959	The flag is ON, when the system is refreshed automatically.
SM976-SM990	The flag is ON, when the system is refreshed automatically.
SM1000	You set the flag to ON and after saving, the system sets the flag to OFF.
SM1006	You set the flag to ON or OFF.
SM1008–SM1015	You set the flag to ON or OFF.
SM1016–SM1031	The flag is ON, when the system is refreshed automatically.
SM1001	ON: the Ethernet connection is active. OFF: the Ethernet connection is not active.
SM1090	ON: the TCP connection is busy.
SM1091	ON: the UDP connection is busy.
SM1100	The flag is refreshed when API 2200-API 2210 is executed or the network cable is reconnected.
SM1106	ON: the PHY initialization fails.
SM1107	ON: the IP address, the netmask address, and the gateway address are set incorrectly.
SM1109	ON: the socket function is enabled and the same port is used.
SM1110	ON: the IP filter of AS-FEN02 is set incorrectly.
SM1111	You set the flag to ON or OFF.
SM1113	ON: there is a server error.
SM1116	ON: the trigger of the PLC parameter is enabled.
SM1117	ON: the trigger of the PLC parameter is triggered.
SM1119	ON: the trigger is enabled and the last mail has been sent successfully.
SM1120	ON: the trigger is enabled and the last mail has been sent with an error.
SM1122–SM1123	ON: the trigger is enabled and there is an SMTP server response timeout.
SM1124	ON: the trigger is enabled and the size of the attachment exceeds the limit.
SM1125	ON: the trigger is enabled and the attachment is not found.
SM1126–SM1127	ON: the trigger of the PLC parameter is enabled.
SM1129	ON: the trigger is enabled and the last mail has been sent successfully.

Special auxiliary relay	Refresh time
SM1130	ON: the trigger is enabled and the last mail has been sent with an error.
SM1132	ON: the trigger is enabled and there is an SMTP server response timeout.
SM1133	ON: the trigger is enabled and there is an SMTP server response error.
SM1134	ON: the trigger is enabled and the size of the attachment exceeds the limit.
SM1135	ON: the trigger is enabled and the attachment is not found.
SM1136	ON: the trigger for the PLC parameter is enabled.
SM1137	ON: the trigger for the PLC parameter is triggered.
SM1139	ON: the trigger is enabled and the last mail has been sent successfully.
SM1140	ON: the trigger is enabled and the last mail has been sent with an error.
SM1142	ON: the trigger is enabled and there is an SMTP server response timeout.
SM1143	ON: the trigger is enabled and there is an SMTP server response error.
SM1144	ON: the trigger is enabled and the size of the attachment exceeds the limit.
SM1145	ON: the trigger is enabled and the attachment is not found.
SM1146	ON: the trigger for the PLC parameter is enabled.
SM1147	ON: the trigger for the PLC parameter is triggered.
SM1149	ON: the trigger is enabled and the last mail has been sent successfully.
SM1150	ON: the trigger is enabled and the last mail has been sent with an error.
SM1152	ON: the trigger is enabled and there is an SMTP server response timeout.
SM1153	ON: the trigger is enabled and there is an SMTP server response error.
SM1154	ON: the trigger is enabled and the size of the attachment exceeds the limit.
SM1155	ON: the trigger is enabled and the attachment is not found.
SM1166	After the data exchange parameters are downloaded, the system is refreshed.
SM1167–SM1183	After the data exchange parameters are downloaded, you set the flag to ON or OFF.
SM1200–SM1215 SM1232–SM1247	ON: when the system is refreshed.
SM1269	ON: there is a socket configuration error.
SM1270–SM1353	The flag is refreshed when the socket function is executed.
SM1440–SM1447	ON: a timeout occurred in the adapter of the I/O connection through EtherNet/IP.
SM1448~SM1455	ON: I/O connection is established through EtherNet/IP (Adapter). OFF: I/O connection is NOT established through EtherNet/IP (Adapter) or the communication is disconnected.
SM1581–SM1588	You set the flag to ON or OFF.
SM1601–SM1608	You set the flag to ON or OFF.
SM1611-SM1618	The system sets the flag to ON or OFF.
SM1621-SM1628	The system sets the flag to ON or OFF.
SM1631–SM1638	The system sets the flag to ON and you set it to OFF.
SM1641–SM1648	You set the flag to ON or OFF.
SM1651–SM1658	The system sets the flag to ON or OFF.
SM1661–SM1668	You set the flag to ON or OFF.
SM1671–SM1682	The system sets the flag to ON or OFF.
SM1683	The system sets the flag to ON or OFF.
SM1684—SM1685	You set the flag to ON or OFF.
SM1691-SM1698	The system sets the flag to ON or OFF.

2.2.9 Stepping Relays (S)

You can easily use the stepping relay in industrial automation to set a procedure. It is the most basic device in sequential function chart (SFC) programming. Refer to the ISPSoft User Manual for more information on sequential function chart programming.

There are 2048 stepping relays, (S0–S2047). Every stepping relay is like an output relay in that it has an output coil, contact A, and contact B. You can use a stepping relay several times in a program, but the relay cannot directly drive the external load. In addition, you can use the stepping relay as a general auxiliary relay when it is not used in a sequential function chart.

2.2.10 Timers (T)

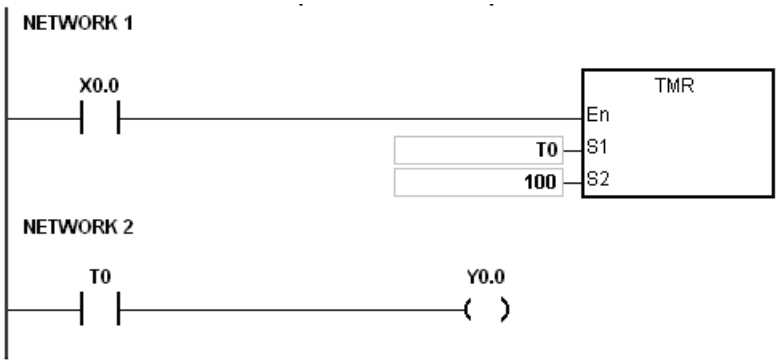
This topic describes the timers available in ISPSoft. Refer to the ISPSoft User Manual for more information on timers.

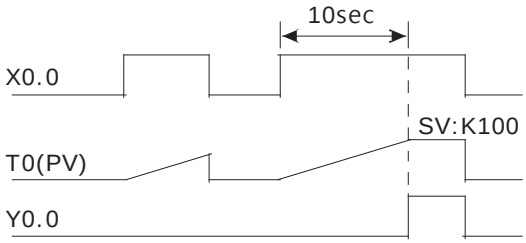
- 100 millisecond timer: The timer specified by the TMR instruction takes 100 milliseconds as the timing unit.
- 1 millisecond timer: The timer specified by the TMRH instruction takes 1 millisecond as the timing unit.
- The accumulative timers are ST0–ST511. If you want to use the device-monitoring function, these timers can monitor T0–T511.
- If you use the same timer repeatedly in a program, including in different TMR and TMRH instructions, the timer setting value is the one that the timer matches first.
- If you use the same timer repeatedly in a program, the timer is OFF when one of the conditional contacts is OFF.
- If you use the same timer in a program as the timer for a subroutine's exclusive use and an accumulative timer in the program, it is OFF when one of the conditional contacts is OFF.
- When the timer switches from ON to OFF and the conditional contact is ON, the timer is reset and counts again.
- When the TMR instruction is executed, the specified timer coil is ON and the timer begins to count. When the value of the timer matches the timer setting value (value of the timer ≧ setting value), the state of the contact is ON.

A. General-purpose timers

When the TMR instruction is executed, the general-purpose timer begins to count. When the value of the timer matches the timer setting value, the output coil is ON.

- When X0=ON and the timer takes 100 ms as the timing unit, the output coil T0 is ON when the value of the timer = timer setting value100.
- When X0=OFF or the power is off, the value of the timer is 0 and the output coil T0 is OFF.

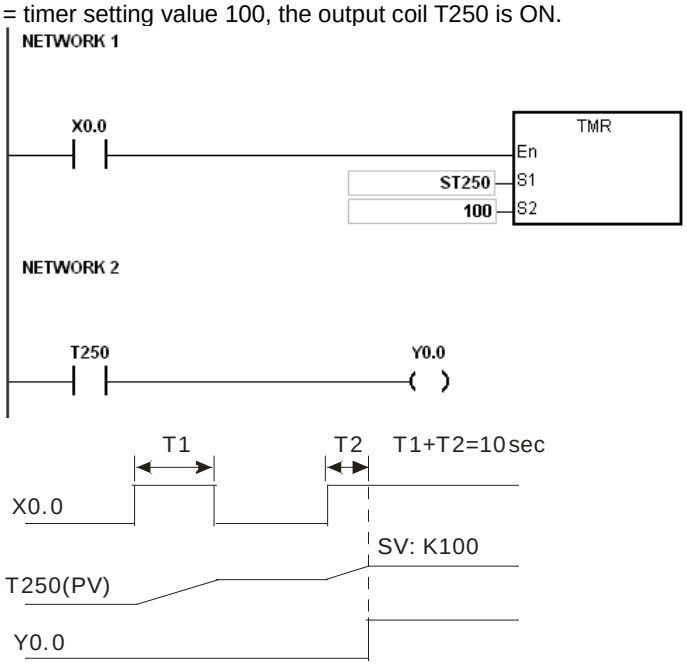




B. Accumulative timers

When the TMR instruction is executed, the accumulative timer begins to count. When the value of the timer matches the timer setting value, the output coil is ON. As long as you add the letter S in front of the letter T, the timer becomes an accumulative timer. When the conditional contact is OFF, the value of the accumulative timer is not reset. When the conditional contact is ON, the accumulative timer counts from the current value.

- When X0=ON and the timer T250 takes 100 ms as the timing unit, the output coil T250 is ON when the value of the timer = timer setting value 100.
- When X0=OFF or the power is off, the accumulative timer ST250 stops counting, and the value of the timer stays the same. When X0=ON, the value of the timer is the accumulating value. When the accumulated value = timer setting value 100, the output coil T250 is ON.



C. Timers used in function blocks

T412–T511 are the timers that you can use in the function block or in interrupts.

When the TMR or END instruction is executed, the timer in the functional block begins to count. When the value of the timer matches the timer setting value, the output coil is ON.

If you use a general-purpose timer in a function block or an interrupt, and the function or interrupt is not executed, the timer cannot count correctly.

2.2.11 Counters

- Characteristics of the 16-bit counter

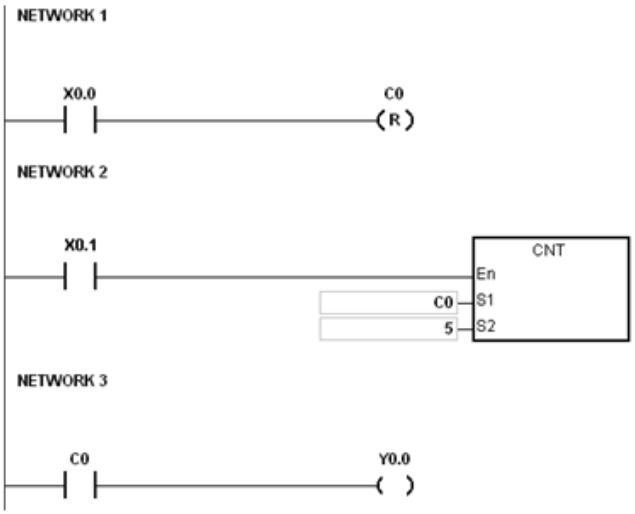
Item	16-bit counter
Type	General type
Number	C0–C511
Direction	Counting up
Setting value	0–32,767
Specifying the counter setting value	The setting value can be either the constant or the value in the data register.
Change of the current value	The counter stops counting when the value of the counter matches the counter setting value.
Output contact	The contact is ON when the value of the counter matches the counter setting value.
Reset	When the instruction RST is executed, the current value is cleared to zero, and the contact is reset of OFF.
Action of the contact	After the scan is complete, the contact acts.

- Function of the counter

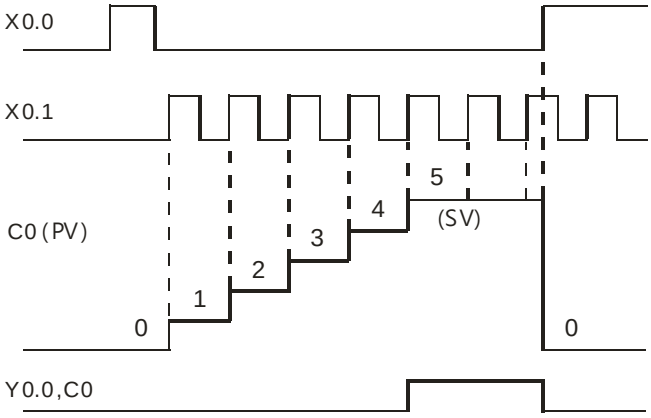
  Each time the input switches from OFF to ON, the value of the counter is the same as the output coil. You can use either the decimal constant or the value in the data register as the counter setting value.

16-bit counter:

1. Setting range: 0–32,767. The setting values 0 and 1 mean the same thing in that the output contact is ON when the counter counts for the first time.
2. For the general-purpose counter, the current value of the counter is cleared when power is lost. If the counter is latching, the current value of the counter and the state of the contact before power was lost power are retained. The latched counter counts from the current value when the power supply is restored.
3. If you use the MOV instruction or ISPSoft to transmit a value larger than the counter setting value to the current value register C0, the contact of the counter C0 is ON and the current value becomes the same as the counter setting value the next time X1 switches from OFF to ON.
4. You can use either the constant or the value in the data register as the counter setting value.
5. The counter setting value can be positive or negative. If the counter counts up from 32,767, the next value is 0.



1. When X0=ON, the RST instruction is executed, the current value of C0 is reset to zero, and the output contact of the counter C0 is FF.
2. When X1 changes from OFF to ON, the value of the counter increments by one.
3. When the value of the counter C0 reaches the counter setting value of 5, the contact of the counter C0 is ON (the current value of C0 = the counter setting value = 5). After that the trigger from X1 is not accepted by C0 and the current value of C0 stays at the value 5.



2.2.12 32-bit Counters (HC)

- Characteristics of the 32-bit counter

Item	32-bit counter		
Type	Up/down counter	Up counter	High-speed counter
Number	HC0–HC63	HC64–HC199	HC200–HC255
Direction	Counts up/down	Counts up	Counts up/down
Setting value	-2,147,483,648 to +2,147,483,647		
Specification of the counter setting value	The counter setting value can be either the constant or the value occupying two data registers (32-bit).		
Change of the current value	The counter keeps counting even after the value of the counter matches the counter setting value.		
Output contact	The contact is ON when the value of the addition counter matches the counter setting value. The contact is reset to OFF when the value of the subtraction counter matches the counter setting value.		
Reset	When the RST instruction is executed, the current value is cleared to zero, and the contact is reset to OFF.		
Action of the contact	After the DCNT instruction scan is complete, the contact acts.		

- 32-bit general-purpose addition/subtraction counter

1. The difference between the 32-bit general-purpose addition counters and the 32-bit general-purpose subtraction

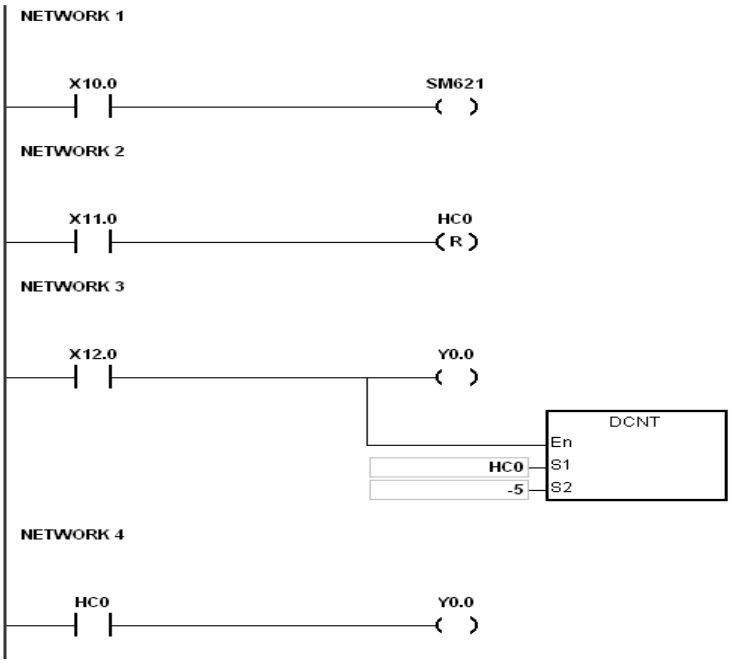
counters depends on the states of the special auxiliary relays SM621–SM684. For example, the counter HC0 is an addition counter when SM621 is OFF, whereas HC0 is a subtraction counter when SM621 is ON.

2. You can use either the constant or the value in the data registers as the counter setting value, and this setting value can be positive or negative. If you use the value in the data registers as the counter setting value, this setting value occupies two consecutive registers.
3. For the general-purpose counter, the current value of the counter is cleared when power is lost. If the counter is latching, the current value of the counter and the state of the contact before loss of power is retained. The latched counter counts from the current value when power is restored.
4. If the counter counts up from 2,147,483,647, the next incremental value is -2,147,483,648. If the counter counts down from -2,147,483,648, the next incremental value is 2,147,483,647.

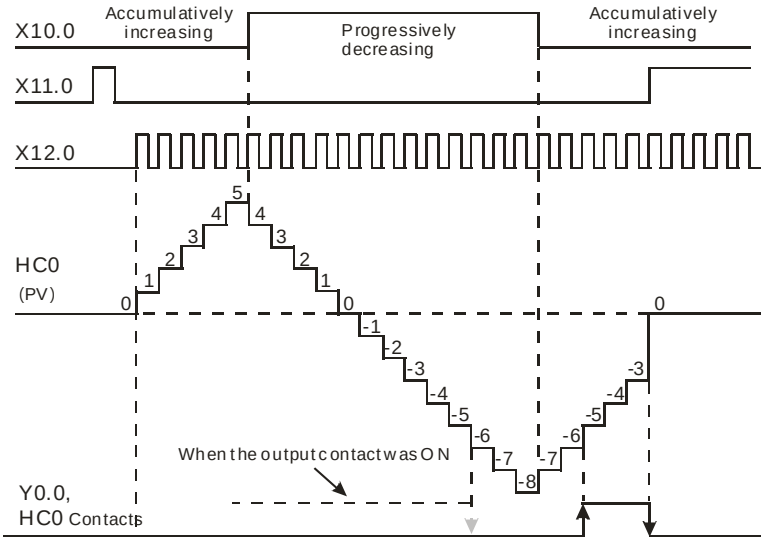
- 32-bit high speed addition/subtraction counter

Refer to the instruction description of API1004 DCNT in DVP-ES3 Series Programming Manual for more details.

Example:



1. X10.0 drives SM621 to determine the counting direction (up/down) for HC0.
2. When X11.0 changes from OFF to ON, the RST instruction is executed and the PV in HC0 is cleared to 0 and its contact is OFF.
3. When X12.0 changes from OFF to ON, PV for HC0 will count up (plus 1) or count down (minus 1).
4. When PV in HC0 changes from -6 to -5, the contact HC0 changes from OFF to ON. When PV in HC0 changes from -5 to -6, the contact HC0 changes from ON to OFF.



2.2.13 Data Registers (D)

The data register stores 16-bit data. The highest bit represents either a positive sign or a negative sign, and the values that the data registers can store range between -32,768 to +32,767.

Two 16-bit registers can be combined into a 32-bit register; for example, (D+1, D) in which the lower number register represents the low 16 bits. The highest bit represents either a positive sign or a negative sign, and the values that the data registers can store range between -2,147,483,648 to +2,147,483,647.

- Four 16-bit registers can be combined into a 64-bit register; for example, (D+3, D+2, D+1, D) in which the lower number register represents the lower 16 bits. The highest bit represents either a positive sign or a negative sign, and the values that the data registers can store range between -9,223,372,036,854,776 to +9,223,372,036,854,775,807.
- You can also use the data registers to refresh the values in the control registers in the modules other than digital I/O modules. Refer to the ISPSoft User Manual for more information on refreshing the values in the control registers.

Types of registers.

- General-purpose registers: When the PLC changes to RUN, or is disconnected, the value in the register is cleared to zero. If you want to retain the data when the PLC changes to RUN, Refer to the ISPSoft User Manual for more information. Note that the value is still cleared to zero when the PLC is disconnected.
- Latched register: If the PLC is disconnected, the data in the latched register is not cleared. In other words, the value before the disconnection is retained. If you want to clear the data in the latched area, you can use the RST or ZRST instructions.
- Area to store special extension modules data: If the PLC is connected with a special extension module, the PLC uses registers in D28000~D28079 and the registers in this area correspond to CRs to update data. If you need to use this area, you need to pay attention not to use the same area repetively.

2.2.14 Special Data Registers (SR)

Every special data register has its own definition and specific function. System status and the error messages are stored in the special data registers. You can also use special data registers to monitor the system statuses. The special data registers and their functions are listed in the table below.

- For SR numbers marked “*”, Refer to the additional remarks in Section 2.2.1.6 on special auxiliary relays/special data registers.
- The “R” in the attribute column indicates that the special data register can read the data; “R/W” in the attribute

column indicates that it can read and write the data.

- The "–" indicates that the status of the special data register does not make any change.
- The "#" indicates that the system is set according to the status of the PLC, and you can read the setting. Refer to the related manual for more information.
- The "Y" in the column latched means it is latched, the "N" means it is non-latched; and "H" means it follows the settings in HWCONFIG.

During execution, you can edit programs in the PLC, but the settings in the HWCONFIG do not change.

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR0	PLC operation/operand error	0	0	-	N	R	0
SR1	The address of the operation error (32-bit)	0	0	-	N	R	0
SR2							
SR4	Grammar check error	0	0	-	N	R	0
SR5	The address of the grammar check error (32-bit)	0	0	-	N	R	0
SR6							
*SR8	Step address at which the watchdog timer is ON (32-bit)	0	-	-	N	R	0
SR9							
SR23	Number of times the MAC address is made	-	-	-	N	R	-
SR28	Last output number when the high speed output instruction is used repeatedly	-1	-1	-1	N	R	-1
SR32	The last instruction address that exceeded the allowed range	-1	-1	-	N	R	-1
SR33							
*SR36	System saves data to the memory card. This function works with SM36	0	-	-	N	R/W	0
*SR40	Number of error logs	-	-	-	Y	R	0
*SR41	Error log pointer	-	-	-	Y	R	0
*SR43	Error log 1: module ID	-	-	-	Y	R	0
*SR44	Error log 1: error code	-	-	-	Y	R	0
*SR45	Error log 1: year and the month	-	-	-	Y	R	0
*SR46	Error log 1: day and the hour	-	-	-	Y	R	0
*SR47	Error log 1: minute and the second	-	-	-	Y	R	0
*SR49	Error log 2: module ID	-	-	-	Y	R	0
*SR50	Error log 2: error code	-	-	-	Y	R	0
*SR51	Error log 2: year and the month	-	-	-	Y	R	0
*SR52	Error log 2: day and the hour	-	-	-	Y	R	0
*SR53	Error log 2: minute and the second	-	-	-	Y	R	0
*SR55	Error log 3: module ID	-	-	-	Y	R	0
*SR56	Error log 3: error code	-	-	-	Y	R	0
*SR57	Error log 3: year and the month	-	-	-	Y	R	0
*SR58	Error log 3: day and the hour	-	-	-	Y	R	0
*SR59	Error log 3: minute and the second	-	-	-	Y	R	0
*SR61	Error log 4: module ID	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR62	Error log 4: error code	-	-	-	Y	R	0
*SR63	Error log 4: year and the month	-	-	-	Y	R	0
*SR64	Error log 4: day and the hour	-	-	-	Y	R	0
*SR65	Error log 4: minute and the second	-	-	-	Y	R	0
*SR67	Error log 5: module ID	-	-	-	Y	R	0
*SR68	Error log 5: error code	-	-	-	Y	R	0
*SR69	Error log 5: year and the month	-	-	-	Y	R	0
*SR70	Error log 5: day and the hour	-	-	-	Y	R	0
*SR71	Error log 5: minute and the second	-	-	-	Y	R	0
*SR73	Error log 6: module ID	-	-	-	Y	R	0
*SR74	Error log 6: error code	-	-	-	Y	R	0
*SR75	Error log 6: year and the month	-	-	-	Y	R	0
*SR76	Error log 6: day and the hour	-	-	-	Y	R	0
*SR77	Error log 6: minute and the second	-	-	-	Y	R	0
*SR79	Error log 7: module ID	-	-	-	Y	R	0
*SR80	Error log 7: error code	-	-	-	Y	R	0
*SR81	Error log 7: year and the month	-	-	-	Y	R	0
*SR82	Error log 7: day and the hour	-	-	-	Y	R	0
*SR83	Error log 7: minute and the second	-	-	-	Y	R	0
*SR85	Error log 8: module ID	-	-	-	Y	R	0
*SR86	Error log 8: error code	-	-	-	Y	R	0
*SR87	Error log 8: year and the month	-	-	-	Y	R	0
*SR88	Error log 8: day and the hour	-	-	-	Y	R	0
*SR89	Error log 8: minute and the second	-	-	-	Y	R	0
*SR91	Error log 9: module ID	-	-	-	Y	R	0
*SR92	Error log 9: error code	-	-	-	Y	R	0
*SR93	Error log 9: year and the month	-	-	-	Y	R	0
*SR94	Error log 9: day and the hour	-	-	-	Y	R	0
*SR95	Error log 9: minute and the second	-	-	-	Y	R	0
*SR97	Error log 10: module ID	-	-	-	Y	R	0
*SR98	Error log 10: error code	-	-	-	Y	R	0
*SR99	Error log 10: year and the month	-	-	-	Y	R	0
*SR100	Error log 10: day and the hour	-	-	-	Y	R	0
*SR101	Error log 10: minute and the second	-	-	-	Y	R	0
*SR103	Error log 11: module ID	-	-	-	Y	R	0
*SR104	Error log 11: error code	-	-	-	Y	R	0
*SR105	Error log 11: year and the month	-	-	-	Y	R	0
*SR106	Error log 11: day and the hour	-	-	-	Y	R	0
*SR107	Error log 11: minute and the second	-	-	-	Y	R	0
*SR109	Error log 12: module ID	-	-	-	Y	R	0
*SR110	Error log 12: error code	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR111	Error log 12: year and the month	-	-	-	Y	R	0
*SR112	Error log 12: day and the hour	-	-	-	Y	R	0
*SR113	Error log 12: minute and the second	-	-	-	Y	R	0
*SR115	Error log 13: module ID	-	-	-	Y	R	0
*SR116	Error log 13: error code	-	-	-	Y	R	0
*SR117	Error log 13: year and the month	-	-	-	Y	R	0
*SR118	Error log 13: day and the hour	-	-	-	Y	R	0
*SR119	Error log 13: minute and the second	-	-	-	Y	R	0
*SR121	Error log 14: module ID	-	-	-	Y	R	0
*SR122	Error log 14: error code	-	-	-	Y	R	0
*SR123	Error log 14: year and the month	-	-	-	Y	R	0
*SR124	Error log 14: day and the hour	-	-	-	Y	R	0
*SR125	Error log 14: minute and the second	-	-	-	Y	R	0
*SR127	Error log 15: module ID	-	-	-	Y	R	0
*SR128	Error log 15: error code	-	-	-	Y	R	0
*SR129	Error log 15: year and the month	-	-	-	Y	R	0
*SR130	Error log 15: day and the hour	-	-	-	Y	R	0
*SR131	Error log 15: minute and the second	-	-	-	Y	R	0
*SR133	Error log 16: module ID	-	-	-	Y	R	0
*SR134	Error log 16: error code	-	-	-	Y	R	0
*SR135	Error log 16: year and the month	-	-	-	Y	R	0
*SR136	Error log 16: day and the hour	-	-	-	Y	R	0
*SR137	Error log 16: minute and the second	-	-	-	Y	R	0
*SR139	Error log 17: module ID	-	-	-	Y	R	0
*SR140	Error log 17: error code	-	-	-	Y	R	0
*SR141	Error log 17: year and the month	-	-	-	Y	R	0
SR142	Error log 17: day and the hour	-	-	-	Y	R	0
*SR143	Error log 17: minute and the second	-	-	-	Y	R	0
*SR145	Error log 18: module ID	-	-	-	Y	R	0
*SR146	Error log 18: error code	-	-	-	Y	R	0
*SR147	Error log 18: year and the month	-	-	-	Y	R	0
*SR148	Error log 18: day and the hour	-	-	-	Y	R	0
*SR149	Error log 18: minute and the second	-	-	-	Y	R	0
*SR151	Error log 19: module ID	-	-	-	Y	R	0
*SR152	Error log 19: error code	-	-	-	Y	R	0
*SR153	Error log 19: year and the month	-	-	-	Y	R	0
*SR154	Error log 19: day and the hour	-	-	-	Y	R	0
*SR155	Error log 19: minute and the second	-	-	-	Y	R	0
*SR157	Error log 20: module ID	-	-	-	Y	R	0
*SR158	Error log 20: error code	-	-	-	Y	R	0
*SR159	Error log 20: year and the month	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR160	Error log 20: day and the hour	-	-	-	Y	R	0
*SR161	Error log 20: minute and the second	-	-	-	Y	R	0
SR162	Length of time that the PLC is powered on (unit: minutes) (32-bit)	-	-	-	Y	R	-
SR163							
SR180	The last warning error code	0	-	-	N	R	0
SR190	Frequency multiplication of the high speed counter group 1 (default: 1-time frequency)	1	-	-	N	R/W	1
SR191	Frequency multiplication of the high speed counter group 2 (default: 1-time frequency)	1	-	-	N	R/W	1
SR192	Frequency multiplication of the high speed counter group 3 (default: 1-time frequency)	1	-	-	N	R/W	1
SR193	Frequency multiplication of the high speed counter group 4 (default: 1-time frequency)	1	-	-	N	R/W	1
SR194	Frequency multiplication of the high speed counter group 5 (default: 1-time frequency)	1	-	-	N	R/W	1
SR195	Frequency multiplication of the high speed counter group 6 (default: 1-time frequency)	1	-	-	N	R/W	1
SR196	Frequency multiplication of the high speed counter group 7 (default: 1-time frequency)	1	-	-	N	R/W	1
SR197	Frequency multiplication of the high speed counter group 8 (default: 1-time frequency)	1	-	-	N	R/W	1
SR198	Pi (π), floating-point number (32-bit)	16# 0FDB	16# 0FDB	16# 0FDB	N	R	16# 0FDB
SR199		16# 4049	16# 4049	16# 4049	N		16# 4049
*SR201	COM1 communication address	-	-	-	H	R/W	1
*SR202	COM2 communication address	-	-	-	H	R/W	1
*SR209	COM1 communication protocol	-	-	-	H	R/W	16# 0024
*SR210	COM1 communication timeout (unit: milliesecond) 0: no timeout	-	-	-	H	R/W	0
*SR212	COM2 communication protocol	-	-	-	H	R/W	16# 0024
*SR213	COM2 communication timeout (unit: milliesecond) 0: no timeout	-	-	-	H	R/W	0
SR217	COM1 baudrate value (unit:100 bps)	96	-	-	H	R/W	96
SR218	COM2 baudrate value (unit:100 bps)	96	-	-	H	R/W	96
*SR220	Real-time clock (RTC) year value: 00–99 (A.D.)	-	-	-	Y	R	0
*SR221	Real-time clock (RTC) month value : 01–12	-	-	-	Y	R	1
*SR222	Real-time clock (RTC) day value : 1–31	-	-	-	Y	R	1
*SR223	Real-time clock (RTC) hour value: 00–23	-	-	-	Y	R	0
*SR224	Real-time clock (RTC) minute value: 00–59	-	-	-	Y	R	0
*SR225	Real-time clock (RTC) second value: 00–59	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR226	Real-time clock (RTC) week value: 1–7	-	-	-	Y	R	1
*SR227	Number of download logs (maximum is 20)	-	-	-	Y	R	0
*SR228	Download log pointer	-	-	-	Y	R	0
*SR229	Download log 1: action number	-	-	-	Y	R	0
*SR230	Download log 1: year and the month	-	-	-	Y	R	0
*SR231	Download log 1: day and the hour	-	-	-	Y	R	0
*SR232	Download log 1: minute and the second	-	-	-	Y	R	0
*SR233	Download log 2: action number	-	-	-	Y	R	0
*SR234	Download log 2: year and the month	-	-	-	Y	R	0
*SR235	Download log 2: day and the hour	-	-	-	Y	R	0
*SR236	Download log 2: minute and the second	-	-	-	Y	R	0
*SR237	Download log 3: action number	-	-	-	Y	R	0
*SR238	Download log 3: year and the month	-	-	-	Y	R	0
*SR239	Download log 3: day and the hour	-	-	-	Y	R	0
*SR240	Download log 3: minute and the second	-	-	-	Y	R	0
*SR241	Download log 4: action number	-	-	-	Y	R	0
*SR242	Download log 4: year and the month	-	-	-	Y	R	0
*SR243	Download log 4: day and the hour	-	-	-	Y	R	0
*SR244	Download log 4: minute and the second	-	-	-	Y	R	0
*SR245	Download log 5: action number	-	-	-	Y	R	0
*SR246	Download log 5: year and the month	-	-	-	Y	R	0
*SR247	Download log 5: day and the hour	-	-	-	Y	R	0
*SR248	Download log 5: minute and the second	-	-	-	Y	R	0
*SR249	Download log 6: action number	-	-	-	Y	R	0
*SR250	Download log 6: year and the month	-	-	-	Y	R	0
*SR251	Download log 6: day and the hour	-	-	-	Y	R	0
*SR252	Download log 6: minute and the second	-	-	-	Y	R	0
*SR253	Download log 7: action number	-	-	-	Y	R	0
*SR254	Download log 7: year and the month	-	-	-	Y	R	0
*SR255	Download log 7: day and the hour	-	-	-	Y	R	0
*SR256	Download log 7: minute and the second	-	-	-	Y	R	0
*SR257	Download log 8: action number	-	-	-	Y	R	0
*SR258	Download log 8: year and the month	-	-	-	Y	R	0
*SR259	Download log 8: day and the hour	-	-	-	Y	R	0
*SR260	Download log 8: minute and the second	-	-	-	Y	R	0
*SR261	Download log 9: action number	-	-	-	Y	R	0
*SR262	Download log 9: year and the month	-	-	-	Y	R	0
*SR263	Download log 9: day and the hour	-	-	-	Y	R	0
*SR264	Download log 9: minute and the second	-	-	-	Y	R	0
*SR265	Download log 10: action number	-	-	-	Y	R	0
*SR266	Download log 10: year and the month	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR267	Download log 10: day and the hour	-	-	-	Y	R	0
*SR268	Download log 10: minute and the second	-	-	-	Y	R	0
*SR269	Download log 11: action number	-	-	-	Y	R	0
*SR270	Download log 11: year and the month	-	-	-	Y	R	0
*SR271	Download log 11: day and the hour	-	-	-	Y	R	0
*SR272	Download log 11: minute and the second	-	-	-	Y	R	0
*SR273	Download log 12: action number	-	-	-	Y	R	0
*SR274	Download log 12: year and the month	-	-	-	Y	R	0
*SR275	Download log 12: day and the hour	-	-	-	Y	R	0
*SR276	Download log 12: minute and the second	-	-	-	Y	R	0
*SR277	Download log 13: action number	-	-	-	Y	R	0
*SR278	Download log 13: year and the month	-	-	-	Y	R	0
*SR279	Download log 13: day and the hour	-	-	-	Y	R	0
*SR280	Download log 13: minute and the second	-	-	-	Y	R	0
*SR281	Download log 14: action number	-	-	-	Y	R	0
*SR282	Download log 14: year and the month	-	-	-	Y	R	0
*SR283	Download log 14: day and the hour	-	-	-	Y	R	0
*SR284	Download log 14: minute and the second	-	-	-	Y	R	0
*SR285	Download log 15: action number	-	-	-	Y	R	0
*SR286	Download log 15: year and the month	-	-	-	Y	R	0
*SR287	Download log 15: day and the hour	-	-	-	Y	R	0
*SR288	Download log 15: minute and the second	-	-	-	Y	R	0
*SR289	Download log 16: action number	-	-	-	Y	R	0
*SR290	Download log 16: year and the month	-	-	-	Y	R	0
*SR291	Download log 16: day and the hour	-	-	-	Y	R	0
*SR292	Download log 16: minute and the second	-	-	-	Y	R	0
*SR293	Download log 17: action number	-	-	-	Y	R	0
*SR294	Download log 17: year and the month	-	-	-	Y	R	0
*SR295	Download log 17: day and the hour	-	-	-	Y	R	0
*SR296	Download log 17: minute and the second	-	-	-	Y	R	0
*SR297	Download log 18: action number	-	-	-	Y	R	0
*SR298	Download log 18: year and the month	-	-	-	Y	R	0
*SR299	Download log 18: day and the hour	-	-	-	Y	R	0
*SR300	Download log 18: minute and the second	-	-	-	Y	R	0
*SR301	Download log 19: action number	-	-	-	Y	R	0
*SR302	Download log 19: year and the month	-	-	-	Y	R	0
*SR303	Download log 19: day and the hour	-	-	-	Y	R	0
*SR304	Download log 19: minute and the second	-	-	-	Y	R	0
*SR305	Download log 20: action number	-	-	-	Y	R	0
*SR306	Download log 20: year and the month	-	-	-	Y	R	0
*SR307	Download log 20: day and the hour	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR308	Download log 20: minute and the second	-	-	-	Y	R	0
*SR309	Number of PLC status change logs (maximum is 20)	-	-	-	Y	R	0
*SR310	PLC status change log pointer	-	-	-	Y	R	0
*SR311	PLC status change log 1: action number	-	-	-	Y	R	0
*SR312	PLC status change log 1: year and the month	-	-	-	Y	R	0
*SR313	PLC status change log 1: day and the hour	-	-	-	Y	R	0
*SR314	PLC status change log 1: minute and the second	-	-	-	Y	R	0
*SR315	PLC status change log 2: action number	-	-	-	Y	R	0
*SR316	PLC status change log 2: year and the month	-	-	-	Y	R	0
*SR317	PLC status change log 2: day and the hour	-	-	-	Y	R	0
*SR318	PLC status change log 2: minute and the second	-	-	-	Y	R	0
*SR319	PLC status change log 3: action number	-	-	-	Y	R	0
*SR320	PLC status change log 3: year and the month	-	-	-	Y	R	0
*SR321	PLC status change log 3: day and the hour	-	-	-	Y	R	0
*SR322	PLC status change log 3: minute and the second	-	-	-	Y	R	0
*SR323	PLC status change log 4: action number	-	-	-	Y	R	0
*SR324	PLC status change log 4: year and the month	-	-	-	Y	R	0
*SR325	PLC status change log 4: day and the hour	-	-	-	Y	R	0
*SR326	PLC status change log 4: minute and the second	-	-	-	Y	R	0
*SR327	PLC status change log 5: action number	-	-	-	Y	R	0
*SR328	PLC status change log 5: year and the month	-	-	-	Y	R	0
*SR329	PLC status change log 5: day and the hour	-	-	-	Y	R	0
*SR330	PLC status change log 5: minute and the second	-	-	-	Y	R	0
*SR331	PLC status change log 6: action number	-	-	-	Y	R	0
*SR332	PLC status change log 6: year and the month	-	-	-	Y	R	0
*SR333	PLC status change log 6: day and the hour	-	-	-	Y	R	0
*SR334	PLC status change log 6: minute and the second	-	-	-	Y	R	0
*SR335	PLC status change log 7: action number	-	-	-	Y	R	0
*SR336	PLC status change log 7: year and the month	-	-	-	Y	R	0
*SR337	PLC status change log 7: day and the hour	-	-	-	Y	R	0
*SR338	PLC status change log 7: minute and the second	-	-	-	Y	R	0
*SR339	PLC status change log 8: action number	-	-	-	Y	R	0
*SR340	PLC status change log 8: year and the month	-	-	-	Y	R	0
*SR341	PLC status change log 8: day and the hour	-	-	-	Y	R	0
*SR342	PLC status change log 8: minute and the second	-	-	-	Y	R	0
*SR343	PLC status change log 9: action number	-	-	-	Y	R	0
*SR344	PLC status change log 9: year and the month	-	-	-	Y	R	0
*SR345	PLC status change log 9: day and the hour	-	-	-	Y	R	0
*SR346	PLC status change log 9: minute and the second	-	-	-	Y	R	0
*SR347	PLC status change log 10: action number	-	-	-	Y	R	0
*SR348	PLC status change log 10: year and the month	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR349	PLC status change log 10: day and the hour	-	-	-	Y	R	0
*SR350	PLC status change log 10: minute and the second	-	-	-	Y	R	0
*SR351	PLC status change log 11: action number	-	-	-	Y	R	0
*SR352	PLC status change log 11: year and the month	-	-	-	Y	R	0
*SR353	PLC status change log 11: day and the hour	-	-	-	Y	R	0
*SR354	PLC status change log 11: minute and the second	-	-	-	Y	R	0
*SR355	PLC status change log 12: action number	-	-	-	Y	R	0
*SR356	PLC status change log 12: year and the month	-	-	-	Y	R	0
*SR357	PLC status change log 12: day and the hour	-	-	-	Y	R	0
*SR358	PLC status change log 12: minute and the second	-	-	-	Y	R	0
*SR359	PLC status change log 13: action number	-	-	-	Y	R	0
*SR360	PLC status change log 13: year and the month	-	-	-	Y	R	0
*SR361	PLC status change log 13: day and the hour	-	-	-	Y	R	0
*SR362	PLC status change log 13: minute and the second	-	-	-	Y	R	0
*SR363	PLC status change log 14: action number	-	-	-	Y	R	0
*SR364	PLC status change log 14: year and the month	-	-	-	Y	R	0
*SR365	PLC status change log 14: day and the hour	-	-	-	Y	R	0
*SR366	PLC status change log 14: minute and the second	-	-	-	Y	R	0
*SR367	PLC status change log 15: action number	-	-	-	Y	R	0
*SR368	PLC status change log 15: year and the month	-	-	-	Y	R	0
*SR369	PLC status change log 15: day and the hour	-	-	-	Y	R	0
*SR370	PLC status change log 15: minute and the second	-	-	-	Y	R	0
*SR371	PLC status change log 16: action number	-	-	-	Y	R	0
*SR372	PLC status change log 16: year and the month	-	-	-	Y	R	0
*SR373	PLC status change log 16: day and the hour	-	-	-	Y	R	0
*SR374	PLC status change log 16: minute and the second	-	-	-	Y	R	0
*SR375	PLC status change log 17: action number	-	-	-	Y	R	0
*SR376	PLC status change log 17: year and the month	-	-	-	Y	R	0
*SR377	PLC status change log 17: day and the hour	-	-	-	Y	R	0
*SR378	PLC status change log 17: minute and the second	-	-	-	Y	R	0
*SR379	PLC status change log 18: action number	-	-	-	Y	R	0
*SR380	PLC status change log 18: year and the month	-	-	-	Y	R	0
*SR381	PLC status change log 18: day and the hour	-	-	-	Y	R	0
*SR382	PLC status change log 18: minute and the second	-	-	-	Y	R	0
*SR383	PLC status change log 19: action number	-	-	-	Y	R	0
*SR384	PLC status change log 19: year and the month	-	-	-	Y	R	0
*SR385	PLC status change log 19: day and the hour	-	-	-	Y	R	0
*SR386	PLC status change log 19: minute and the second	-	-	-	Y	R	0
*SR387	PLC status change log 20: action number	-	-	-	Y	R	0
*SR388	PLC status change log 20: year and the month	-	-	-	Y	R	0
*SR389	PLC status change log 20: day and the hour	-	-	-	Y	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR390	PLC status change log 20: minute and the second	-	-	-	Y	R	0
*SR391	Real-time clock (RTC) year value: 00–99 (A.D.)	-	-	-	Y	R	0
*SR392	Real-time clock (RTC) month value: 01–12	-	-	-	Y	R	1
*SR393	Real-time clock (RTC) day value: 1–31	-	-	-	Y	R	1
*SR394	Real-time clock (RTC) hour value: 00–23	-	-	-	Y	R	0
*SR395	Real-time clock (RTC) minute value: 00–59	-	-	-	Y	R	0
*SR396	Real-time clock (RTC) second value: 00–59	-	-	-	Y	R	0
*SR397	Real-time clock (RTC) week value: 1–7	-	-	-	Y	R	1
SR407	When the PLC runs, the value in SR407 increases by one every second. SR407 counts from 0 to 32767, and then from -32768 to 0.	0	0	-	N	R/W	0
SR408	When the PLC runs, the value in SR408 increases by one every scan cycle. SR408 counts from 0 to 32767, and then from -32768 to 0.	0	0	-	N	R/W	0
SR411	The current scan time is stored in SR411 and SR412 (unit of measurement is 100 microseconds). Milliseconds are stored in SR411 (range is 0–65535). Microseconds are stored in SR412 (the range is 0–900). For example, if SR411=12 and SR412=300, then the current scan time is 12.3 milliseconds.	0	-	-	N	R	0
SR412		0	-	-	N	R	0
SR413	The maximum scan time is stored in SR413 and SR414 (unit of measurement is 100 microseconds). The value of the millisecond is stored in SR413.	0	-	-	N	R	0
SR414							
SR415	The minimum scan time is stored in SR415 and SR416 (unit of measurement is 100 microseconds). The value of the millisecond is stored in SR415.	0	-	-	N	R	0
SR416							
SR421	Duration of the timer interrupt I601 (unit: milliesecond). The default is 0, meaning the system uses the settings in HWCONFIG.	0	0	-	N	R/W	0
SR422	Duration of the timer interrupt I602 (unit: milliesecond). The default is 0, meaning the system uses the settings in HWCONFIG.	0	0	-	N	R/W	0
SR423	Duration of the timer interrupt I603 (unit: milliesecond). The default is 0, meaning the system uses the settings in HWCONFIG.	0	0	-	N	R/W	0
SR424	Duration of the timer interrupt I604 (unit: 0.1 milliesecond). The default is 0, meaning the system uses the settings in HWCONFIG.	0	0	-	N	R/W	0
SR440	MAC address (Example: 12:34:56:78:9A:BC => SR440=16#1234, SR441=16#5678, SR442=16#9ABC)	-	-	-	Y	R	-
SR441		-	-	-	Y	R	-
SR442		-	-	-	Y	R	-
SR443	PLC series	-	-	-	Y	R	-
SR444		-	-	-	Y	R	-

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR445	EX：AS324MTAW15500012	-	-	-	Y	R	-
SR446	AS➔ SR443 = 16#5341	-	-	-	Y	R	-
SR447	32 ➔ SR444 = 16#3233	-	-	-	Y	R	-
SR448	4M ➔ SR445 = 16#4D34	-	-	-	Y	R	-
SR449	TA ➔ SR446 = 16#4154	-	-	-	Y	R	-
SR450	W1 ➔ SR447 = 16#3157	-	-	-	Y	R	-
SR451	55 ➔ SR448 = 16#3535 00 ➔ SR449 = 16#3030 01 ➔ SR450 = 16#3130 2 ➔ SR451 = 16#0032	-	-	-	Y	R	-
*SR453	If an error occurs during the operation of the memory card, the error code is recorded.	-	-	-	Y	R	0
SR460	Y0/axis 1 (Y0/Y1) position (unit: number of pulse)	-	-	-	Y	R/W	0
SR461							
SR462	Axis 1 (Y0/Y1) output mode	-	-	-	Y	R/W	0
SR463	Axis 1 (Y0/Y1) starting/ending frequency	-	-	-	Y	R/W	200
SR464	Axis 1 (Y0/Y1) accelerating time	-	-	-	Y	R/W	200
SR465	Axis 1 (Y0/Y1) decelerating time	-	-	-	Y	R/W	200
SR466	Axis 1 (Y0/Y1) JOG frequency	-	-	-	Y	R/W	200
SR467	Axis 1 (Y0/Y1) number in the current position planning table	0	0	-	N	R	0
SR468	Axis 1 (Y0/Y1) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR469	Axis 1 (Y0/Y1) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR470	Axis 1 (Y0/Y1) position of the machine unit (single-precision floating-point values)	-	-	-	Y	R	0
SR471							
SR472	Axis 1 (Y0/Y1) target frequency for fixed slope	-	-	-	Y	R	0
SR473							
SR474	Y1 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR475							
SR476	Y1 starting/ending frequency	-	-	-	Y	R/W	200
SR477	Y1 accelerating/decelerating time	-	-	-	Y	R/W	200
SR478	Axis 1 (Y0) backlash compensation pulse	-	-	-	Y	R/W	0
SR479	Y1 backlash compensation pulse	-	-	-	Y	R/W	0
SR480	Y2/axis 2 (Y2/Y3) position (unit: number of pulse)	-	-	-	Y	R/W	0
SR481							
SR482	Y2/axis 2 (Y2/Y3) output mode	-	-	-	Y	R/W	0
SR483	Y2/axis 2 (Y2/Y3) starting/ending frequency	-	-	-	Y	R/W	200
SR484	Y2/axis 2 (Y2/Y3) accelerating time	-	-	-	Y	R/W	200
SR485	Y2/axis 2 (Y2/Y3) decelerating time	-	-	-	Y	R/W	200
SR486	Y2/axis 2 (Y2/Y3) JOG frequency	-	-	-	Y	R/W	200
SR487	Y2/axis 2 (Y2/Y3) number in the current position planning table	0	0	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR488	Y2/axis 2 (Y2/Y3) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR489	Y2/axis 2 (Y2/Y3) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR490	Y2/axis 2 (Y2/Y3) position of the Machine unit (single-precision floating-point values)	-	-	-	Y	R	0
SR491							
SR492	Y2/axis 2 (Y2/Y3) target frequency for fixed slope	-	-	-	Y	R	0
SR493							
SR494	Y3 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR495							
SR496	Y3 starting/ending frequency	-	-	-	Y	R/W	200
SR497	Y3 accelerating/decelerating time	-	-	-	Y	R/W	200
SR498	Axis 2 (Y2) backlash compensation pulse	-	-	-	Y	R/W	0
SR499	Y3 backlash compensation pulse	-	-	-	Y	R/W	0
SR500	Y4/axis 3 (Y4/Y5) position (unit: number of pulse)	-	-	-	Y	R/W	0
SR501							
SR502	Y4/axis 3 (Y4/Y5) output mode	-	-	-	Y	R/W	0
SR503	Y4/axis 3 (Y4/Y5) starting/ending frequency	-	-	-	Y	R/W	200
SR504	Y4/axis 3 (Y4/Y5) accelerating time	-	-	-	Y	R/W	200
SR505	Y4/axis 3 (Y4/Y5) decelerating time	-	-	-	Y	R/W	200
SR506	Y4/axis 3 (Y4/Y5) JOG frequency	-	-	-	Y	R/W	200
SR507	Y4/axis 3 (Y4/Y5) number in the current position planning table	0	0	-	N	R	0
SR508	Y4/axis 3 (Y4/Y5) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR509	Y4/axis 3 (Y4/Y5) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR510	Y4/axis 3 (Y4/Y5) position of the Machine unit (single-precision floating-point values)	-	-	-	Y	R	0
SR511							
SR512	Y4/axis 3 (Y4/Y5) target frequency for fixed slope	-	-	-	Y	R	0
SR513							
SR514	Y5 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR515							
SR516	Y5 starting/ending frequency	-	-	-	Y	R/W	200
SR517	Y5 accelerating/decelerating time	-	-	-	Y	R/W	200
SR518	Axis 3 (Y4) backlash compensation pulse	-	-	-	Y	R/W	0
SR519	Y5 backlash compensation pulse	-	-	-	Y	R/W	0
SR520	Y6/axis 4 (Y6/Y7) position (unit: number of pulse)	-	-	-	Y	R/W	0
SR521							
SR522	Y6/axis 4 (Y6/Y7) output mode	-	-	-	Y	R/W	0
SR523	Y6/axis 4 (Y6/Y7) starting/ending frequency	-	-	-	Y	R/W	200
SR524	Y6/axis 4 (Y6/Y7) accelerating time	-	-	-	Y	R/W	200

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR525	Y6/axis 4 (Y6/Y7) decelerating time	-	-	-	Y	R/W	200
SR526	Y6/axis 4 (Y6/Y7) JOG frequency	-	-	-	Y	R/W	200
SR527	Y6/axis 4 (Y6/Y7) number in the current position planning table	0	0	-	N	R	0
SR528	Y6/axis 4 (Y6/Y7) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR529	Y6/axis 4 (Y6/Y7) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR530	Y6/axis 4 (Y6/Y7) position of the Machine unit(single-precision floating-point values)	-	-	-	Y	R	0
SR531							
SR532	Y6/axis 4 (Y6/Y7) target frequency for fixed slope	-	-	-	Y	R	0
SR533							
SR534	Y7 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR535							
SR536	Y7 starting/ending frequency	-	-	-	Y	R/W	200
SR537	Y7 accelerating/decelerating time	-	-	-	Y	R/W	200
SR538	Axis 4 (Y6) backlash compensation pulse	-	-	-	Y	R/W	0
SR539	Y7 backlash compensation pulse	-	-	-	Y	R/W	0
SR540	Y8/axis 5 (Y8/Y9) position where (unit: number of pulse)	-	-	-	Y	R/W	0
SR541							
SR542	The axis 5 (Y8/Y9) output mode	-	-	-	Y	R/W	0
SR543	Y8/axis 5 (Y8/Y9) starting/ending frequency	-	-	-	Y	R/W	200
SR544	Y8/axis 5 (Y8/Y9) accelerating time	-	-	-	Y	R/W	200
SR545	Y8/axis 5 (Y8/Y9) decelerating time	-	-	-	Y	R/W	200
SR546	Y8/axis 5 (Y8/Y9) JOG frequency	-	-	-	Y	R/W	200
SR547	Y8/axis 5 (Y8/Y9)number in the current position planning table	0	0	-	N	R	0
SR548	Y8/axis 5 (Y8/Y9) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR549	Y8/axis 5 (Y8/Y9) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR550	Y8/axis 5 (Y8/Y9) position of the Machine unit (single-precision floating-point values)	-	-	-	Y	R	0
SR551							
SR552	Y8/axis 5 (Y8/Y9) target frequency for fixed slope	-	-	-	Y	R	0
SR553							
SR554	Y9 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR555							
SR556	Y9 starting/ending frequency	-	-	-	Y	R/W	200
SR557	Y9 accelerating/decelerating time	-	-	-	Y	R/W	200
SR558	Axis 5 (Y8) backlash compensation pulse	-	-	-	Y	R/W	0
SR559	Y9 backlash compensation pulse	-	-	-	Y	R/W	0
SR560	Y10/axis 6 (Y10/Y11) position (unit: number of pulse)	-	-	-	Y	R/W	0
SR561							

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR562	Y10/axis 6 (Y10/Y11) output mode	-	-	-	Y	R/W	0
SR563	Y10/axis 6 (Y10/Y11) starting/ending frequency	-	-	-	Y	R/W	200
SR564	Y10/axis 6 (Y10/Y11) accelerating time	-	-	-	Y	R/W	200
SR565	Y10/axis 6 (Y10/Y11) decelerating time	-	-	-	Y	R/W	200
SR566	Y10/axis 6 (Y10/Y11) JOG frequency	-	-	-	Y	R/W	200
SR567	Y10/axis 6 (Y10/Y11) number in the current position planning table	0	0	-	N	R	0
SR568	Y10/axis 6 (Y10/Y11) numerator value transferred from the machine unit	-	-	-	H	R/W	0
SR569	Y10/axis 6 (Y10/Y11) denominator value transferred from the machine unit	-	-	-	H	R/W	0
SR570	Y10/axis 6 (Y10/Y11) position of the Machine unit (single-precision floating-point values)	-	-	-	Y	R	0
SR571							
SR572	Y10/axis 6 (Y10/Y11) target frequency for fixed slope	-	-	-	Y	R	0
SR573							
SR574	Y11 position (unit: number of pulse)	-	-	-	Y	R/W	0
SR575							
SR576	Y11 starting/ending frequency	-	-	-	Y	R/W	200
SR577	Y11 accelerating/decelerating time	-	-	-	Y	R/W	200
SR578	Axis 6 (Y10) backlash compensation pulse	-	-	-	Y	R/W	0
SR579	Y11 backlash compensation pulse	-	-	-	Y	R/W	0
*SR580	Axis 1 (Y0/Y1) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR581							
*SR582	Axis 1 (Y0/Y1) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR583							
*SR584	Axis 2 (Y2/Y3) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR585							
*SR586	Axis 2 (Y2/Y3) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR587							
*SR588	Axis 3(Y4/Y5) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR589							
*SR590	Axis 3(Y4/Y5) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR591							
*SR592	Axis 4(Y6/Y7) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR593							
*SR594	Axis 4(Y6/Y7) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR595							
*SR596	Axis 5(Y8/Y9) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR597							
*SR598	Axis 5(Y8/Y9) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR599							

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR600	Axis 6(Y10/Y11) positive limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR601							
*SR602	Axis 6(Y10/Y11) negative limit in ISPSoft (unit: number of pulse)	-	-	-	H	R/W	0
*SR603							
*SR604	Axis 1 (Y0/Y1) S curve mode	0	-	-	N	R/W	0
*SR605	Axis 2 (Y2/Y3) S curve mode	0	-	-	N	R/W	0
*SR606	Axis 3 (Y4/Y5) S curve mode	0	-	-	N	R/W	0
*SR607	Axis 4 (Y6/Y7) S curve mode	0	-	-	N	R/W	0
SR610	Axis 1 (Y0/Y1) current output speed (unit: Hz)	0	0	0	N	R	0
SR611							
SR612	axis 2 (Y2/Y3) current output speed (unit: Hz)	0	0	0	N	R	0
SR613							
SR614	Axis 3 (Y4/Y5) current output speed (unit: Hz)	0	0	0	N	R	0
SR615							
SR616	Axis 4 (Y6/Y7) current output speed (unit: Hz)	0	0	0	N	R	0
SR617							
SR618	Axis 5 (Y8/Y9) current output speed (unit: Hz)	0	0	0	N	R	0
SR619							
SR620	Axis 6 (Y10/Y11) current output speed (unit: Hz)	0	0	0	N	R	0
SR621							
SR623	External interrupt condition: the X0–X15 input points are falling-edge triggered	FFFF	FFFF	-	N	R	FFFF
SR624	External interrupt condition: the X0–X15 input points are rising-edge triggered	FFFF	FFFF	-	N	R	FFFF
SR625	Condition of the high-speed comparison interrupt I200–I233	FFFF	FFFF	-	N	R	FFFF
SR626	Condition of the high-speed comparison interrupt I240–I253	FFFF	FFFF	-	N	R	FFFF
SR627	Condition of the high-speed comparison interrupt I260–I267	FFFF	FFFF	-	N	R	FFFF
SR628	Condition of the communication interrupts I300–I307	FFFF	FFFF	-	N	R	FFFF
SR629	Condition of the output interrupts I500–I505	FFFF	FFFF	-	N	R	FFFF
SR630	Condition of the output interrupts I510–I519	FFFF	FFFF	-	N	R	FFFF
SR632	Condition of the timer interrupts I601–I604	FFFF	FFFF	-	N	R	FFFF
SR633	Condition of the extension module interrupts I400–I415	FFFF	FFFF	-	N	R	FFFF
SR634	Condition of the extension module interrupts I416–I431	FFFF	FFFF	-	N	R	FFFF
SR640	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y0 output pulse	0	-	-	N	R/W	0
SR641	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y1 output pulse	0	-	-	N	R/W	0
SR642	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y2 output pulse	0	-	-	N	R/W	0
SR643	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y3 output pulse	0	-	-	N	R/W	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR644	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y4 output pulse	0	-	-	N	R/W	0
SR645	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y5 output pulse	0	-	-	N	R/W	0
SR646	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y6 output pulse	0	-	-	N	R/W	0
SR647	Set the outputting time 0-20ms sooner (unit: 1ms) to work with the Y7 output pulse	0	-	-	N	R/W	0
*SR652	Pulse number in acceleration from the latest position output (lower 16 bits)	0	0	-	N	R	0
*SR653	Pulse number in acceleration from the latest position output (higher 16 bits)	0	0	-	N	R	0
*SR654	Pulse number in deceleration from the latest position output (lower 16 bits)	0	0	-	N	R	0
*SR655	Pulse number in deceleration from the latest position output (higher 16 bits)	0	0	-	N	R	0
SR658	The number of the Delta CANopen communication axis from the Delta servo which has a communication error	0	-	-	N	R	0
SR659	Delta CANopen communication error	0	-	-	N	R	0
SR661	The PR command of the Delta CANopen communication axis 1 from the Delta servo	0	-	-	N	R	0
SR662	PR command of the Delta CANopen communication axis 2 from the Delta servo	0	-	-	N	R	0
SR663	PR command of the Delta CANopen communication axis 3 from the Delta servo	0	-	-	N	R	0
SR664	PR command of the Delta CANopen communication axis 4 from the Delta servo	0	-	-	N	R	0
SR665	PR command of the Delta CANopen communication axis 5 from the Delta servo	0	-	-	N	R	0
SR666	PR command of the Delta CANopen communication axis 6 from the Delta servo	0	-	-	N	R	0
SR667	PR command of the Delta CANopen communication axis 7 from the Delta servo	0	-	-	N	R	0
SR668	PR command of the Delta CANopen communication axis 8 from the Delta servo	0	-	-	N	R	0
SR671	Alarm code of the Delta CANopen communication axis 1 from the Delta servo	0	-	-	N	R	0
SR672	Alarm code of the Delta CANopen communication axis 2 from the Delta servo	0	-	-	N	R	0
SR673	Alarm code of the Delta CANopen communication axis 3 from the Delta servo	0	-	-	N	R	0
SR674	Alarm code of the Delta CANopen communication axis 4 from the Delta servo	0	-	-	N	R	0
SR675	Alarm code of the Delta CANopen communication axis 5 from the Delta servo	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR676	Alarm code of the Delta CANopen communication axis 6 from the Delta servo	0	-	-	N	R	0
SR677	Alarm code of the Delta CANopen communication axis 7 from the Delta servo	0	-	-	N	R	0
SR678	Alarm code of the Delta CANopen communication axis 8 from the Delta servo	0	-	-	N	R	0
SR681	The DO state of the Delta CANopen communication axis 1 from the Delta servo	0	-	-	N	R	0
SR682	The DO state of the Delta CANopen communication axis 2 from the Delta servo	0	-	-	N	R	0
SR683	The DO state of the Delta CANopen communication axis 3 from the Delta servo	0	-	-	N	R	0
SR684	The DO state of the Delta CANopen communication axis 4 from the Delta servo	0	-	-	N	R	0
SR685	The DO state of the Delta CANopen communication axis 5 from the Delta servo	0	-	-	N	R	0
SR686	The DO state of the Delta CANopen communication axis 6 from the Delta servo	0	-	-	N	R	0
SR687	The DO state of the Delta CANopen communication axis 7 from the Delta servo	0	-	-	N	R	0
SR688	The DO state of the Delta CANopen communication axis 8 from the Delta servo	0	-	-	N	R	0
SR691 SR692	Current position of the Delta CANopen communication axis 1 from the Delta servo (32-bit)	0	-	-	N	R	0
SR693 SR694	Current position of the Delta CANopen communication axis 2 from the Delta servo (32-bit)	0	-	-	N	R	0
SR695 SR696	Current position of the Delta CANopen communication axis 3 from the Delta servo (32-bit)	0	-	-	N	R	0
SR697 SR698	Current position of the Delta CANopen communication axis 4 from the Delta servo (32-bit)	0	-	-	N	R	0
SR699 SR700	Current position of the Delta CANopen communication axis 5 from the Delta servo (32-bit)	0	-	-	N	R	0
SR701 SR702	Current position of the Delta CANopen communication axis 6 from the Delta servo (32-bit)	0	-	-	N	R	0
SR703 SR704	Current position of the Delta CANopen communication axis 7 from the Delta servo (32-bit)	0	-	-	N	R	0
SR705 SR706	Current position of the Delta CANopen communication axis 8 from the Delta servo (32-bit)	0	-	-	N	R	0
SR711 SR712	Target position of the Delta CANopen communication axis 1 from the Delta servo (32-bit)	0	-	-	N	R	0
SR713 SR714	Target position of the Delta CANopen communication axis 2 from the Delta servo (32-bit)	0	-	-	N	R	0
SR715	Target position of the Delta CANopen communication axis 3	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR716	from the Delta servo (32-bit)						
SR717 SR718	Target position of the Delta CANopen communication axis 4 from the Delta servo (32-bit)	0	-	-	N	R	0
SR719 SR720	Target position of the Delta CANopen communication axis 5 from the Delta servo (32-bit)	0	-	-	N	R	0
SR721 SR722	Target position of the Delta CANopen communication axis 6 from the Delta servo (32-bit)	0	-	-	N	R	0
SR723 SR724	Target position of the Delta CANopen communication axis 7 from the Delta servo (32-bit)	0	-	-	N	R	0
SR725 SR726	Target position of the Delta CANopen communication axis 8 from the Delta servo (32-bit)	0	-	-	N	R	0
SR731	Current DI state of Delta CANopen communication ID 1 from the Delta servo	0	-	-	N	R	0
SR732	Current DI state of Delta CANopen communication ID 2 from the Delta servo	0	-	-	N	R	0
SR733	Current DI state of Delta CANopen communication ID 3 from the Delta servo	0	-	-	N	R	0
SR734	Current DI state of Delta CANopen communication ID 4 from the Delta servo	0	-	-	N	R	0
SR735	Current DI state of Delta CANopen communication ID 5 from the Delta servo	0	-	-	N	R	0
SR736	Current DI state of Delta CANopen communication ID 6 from the Delta servo	0	-	-	N	R	0
SR737	Current DI state of Delta CANopen communication ID 7 from the Delta servo	0	-	-	N	R	0
SR738	Current DI state of Delta CANopen communication ID 8 from the Delta servo	0	-	-	N	R	0
SR741	Current torque of Delta CANopen communication ID 1 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR742	Current torque of Delta CANopen communication ID 2 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR743	Current torque of Delta CANopen communication ID 3 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR744	Current torque of Delta CANopen communication ID 4 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR745	Current torque of Delta CANopen communication ID 5 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR746	Current torque of Delta CANopen communication ID 6 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR747	Current torque of Delta CANopen communication ID 7 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR748	Current torque of Delta CANopen communication ID 8 from the Delta servo (unit: 0.1%)	0	-	-	N	R	0
SR751	Current state of Delta CANopen communication slave ID 21	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	from the Delta motor						
SR752	Current state of Delta CANopen communication slave ID 22 from the Delta motor	0	-	-	N	R	0
SR753	Current state of Delta CANopen communication slave ID 23 from the Delta motor	0	-	-	N	R	0
SR754	Current state of Delta CANopen communication slave ID 24 from the Delta motor	0	-	-	N	R	0
SR755	Current state of Delta CANopen communication slave ID 25 from the Delta motor	0	-	-	N	R	0
SR756	Current state of Delta CANopen communication slave ID 26 from the Delta motor	0	-	-	N	R	0
SR757	Current state of Delta CANopen communication slave ID 27 from the Delta motor	0	-	-	N	R	0
SR758	Current state of Delta CANopen communication slave ID 28 from the Delta motor	0	-	-	N	R	0
SR761	Current RPM of Delta CANopen communication slave ID 21 from the Delta motor	0	-	-	N	R	0
SR762	Current RPM of Delta CANopen communication slave ID 22 from the Delta motor	0	-	-	N	R	0
SR763	Current RPM of Delta CANopen communication slave ID 23 from the Delta motor	0	-	-	N	R	0
SR764	Current RPM of Delta CANopen communication slave ID 24 from the Delta motor	0	-	-	N	R	0
SR765	Current RPM of Delta CANopen communication slave ID 25 from the Delta motor	0	-	-	N	R	0
SR766	Current RPM of Delta CANopen communication slave ID 26 from the Delta motor	0	-	-	N	R	0
SR767	Current RPM of Delta CANopen communication slave ID 27 from the Delta motor	0	-	-	N	R	0
SR768	Current RPM of Delta CANopen communication slave ID 28 from the Delta motor	0	-	-	N	R	0
SR771	Current torque of Delta CANopen communication ID 21 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR772	Current torque of Delta CANopen communication ID 22 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR773	Current torque of Delta CANopen communication ID 23 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR774	Current torque of Delta CANopen communication ID 24 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR775	Current torque of Delta CANopen communication ID 25 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR776	Current torque of Delta CANopen communication ID 26 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR777	Current torque of Delta CANopen communication ID 27 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR778	Current torque of Delta CANopen communication ID 28 from the Delta motor (unit: 0.1%)	0	-	-	N	R	0
SR781	Current DI state of Delta CANopen communication ID 21 from the Delta motor	0	-	-	N	R	0
SR782	Current DI state of Delta CANopen communication ID 22 from the Delta motor	0	-	-	N	R	0
SR783	Current DI state of Delta CANopen communication ID 23 from the Delta motor	0	-	-	N	R	0
SR784	Current DI state of Delta CANopen communication ID 24 from the Delta motor	0	-	-	N	R	0
SR785	Current DI state of Delta CANopen communication ID 25 from the Delta motor	0	-	-	N	R	0
SR786	Current DI state of Delta CANopen communication ID 26 from the Delta motor	0	-	-	N	R	0
SR787	Current DI state of Delta CANopen communication ID 27 from the Delta motor	0	-	-	N	R	0
SR788	Current DI state of Delta CANopen communication ID 28 from the Delta motor	0	-	-	N	R	0
SR791	Read the self-defined value from Delta CANopen communication axis 1 (lower 16 bits)	0	-	-	N	R	0
SR792	Read the self-defined value from Delta CANopen communication axis 1 (higher16 bits)	0	-	-	N	R	0
SR793	Read the self-defined value from Delta CANopen communication axis 2 (lower 16 bits)	0	-	-	N	R	0
SR794	Read the self-defined value from Delta CANopen communication axis 2 (higher16 bits)	0	-	-	N	R	0
SR795	Read the self-defined value from Delta CANopen communication axis 3 (lower 16 bits)	0	-	-	N	R	0
SR796	Read the self-defined value from Delta CANopen communication axis 3 (higher16 bits)	0	-	-	N	R	0
SR797	Read the self-defined value from Delta CANopen communication axis 4 (lower 16 bits)	0	-	-	N	R	0
SR798	Read the self-defined value from Delta CANopen communication axis 4 (higher16 bits)	0	-	-	N	R	0
SR799	Read the self-defined value from Delta CANopen communication axis 5 (lower 16 bits)	0	-	-	N	R	0
SR800	Read the self-defined value from Delta CANopen communication axis 5 (higher16 bits)	0	-	-	N	R	0
SR801	Read the self-defined value from Delta CANopen communication axis 6 (lower 16 bits)	0	-	-	N	R	0
SR802	Read the self-defined value from Delta CANopen communication axis 6 (higher16 bits)	0	-	-	N	R	0
SR803	Read the self-defined value from Delta CANopen communication axis 7 (lower 16 bits)	0	-	-	N	R	0
SR804	Read the self-defined value from Delta CANopen	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	communication axis 7 (higher16 bits)						
SR805	Read the self-defined value from Delta CANopen communication axis 8 (lower 16 bits)	0	-	-	N	R	0
SR806	Read the self-defined value from Delta CANopen communication axis 8 (higher16 bits)	0	-	-	N	R	0
SR811	The setting value for the Delta CANopen communication axis 1 to complete positioning. (If the setting value is 0, it indicates the current position should be the same as the target position to complete the positioning. If the setting value is 5, it indicates the distance between the current position and the target position can be +5 to -5 to complete the positioning.)	0	-	-	N	RW	0
SR812	The setting value for the Delta CANopen communication axis 2 to complete positioning.	0	-	-	N	RW	0
SR813	The setting value for the Delta CANopen communication axis 3 to complete positioning.	0	-	-	N	RW	0
SR814	The setting value for the Delta CANopen communication axis 4 to complete positioning.	0	-	-	N	RW	0
SR815	The setting value for the Delta CANopen communication axis 5 to complete positioning.	0	-	-	N	RW	0
SR816	The setting value for the Delta CANopen communication axis 6 to complete positioning.	0	-	-	N	RW	0
SR817	The setting value for the Delta CANopen communication axis 7 to complete positioning.	0	-	-	N	RW	0
SR818	The setting value for the Delta CANopen communication axis 8 to complete positioning.	0	-	-	N	RW	0
SR820	Code for the state of the master/slave in CANopen DS301 communication	-	-	-	Y	R	0
SR821	CANopen DS301 version code	-	-	-	Y	R	-
SR822	CANopen communication baudrate (unit: 1kbps)	-	-	-	H	R	125
*SR823	The number of hardware receiving error during CANopen communication (Max. 255)	0	-	-	N	R	0
*SR824	The number of hardware sending error during CANopen communication (Max. 255)	0	-	-	N	R	0
SR825	Code for the master state in CANopen DS301 communication	-1	-	-	N	R	-1
SR826	State of slave ID 1–16 in CANopen DS301 communication	-1	-	-	N	R	-1
SR827	State of slave ID 17–32 in CANopen DS301 communication	-1	-	-	N	R	-1
SR828	State of slave ID 33–48 in CANopen DS301 communication	-1	-	-	N	R	-1
SR829	State of slave ID 49–64 in CANopen DS301 communication	-1	-	-	N	R	-1
SR830	State of slave ID 1 in CANopen DS301 communication	-1	-	-	N	R	-1
SR831	State of slave ID 2 in CANopen DS301 communication	-1	-	-	N	R	-1
SR832	State of slave ID 3 in CANopen DS301 communication	-1	-	-	N	R	-1
SR833	State of slave ID 4 in CANopen DS301 communication	-1	-	-	N	R	-1
SR834	State of slave ID 5 in CANopen DS301 communication	-1	-	-	N	R	-1
SR835	State of slave ID 6 in CANopen DS301 communication	-1	-	-	N	R	-1

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR836	State of slave ID 7 in CANopen DS301 communication	-1	-	-	N	R	-1
SR837	State of slave ID 8 in CANopen DS301 communication	-1	-	-	N	R	-1
SR838	State of slave ID 9 in CANopen DS301 communication	-1	-	-	N	R	-1
SR839	State of slave ID 10 in CANopen DS301 communication	-1	-	-	N	R	-1
SR840	State of slave ID 11 in CANopen DS301 communication	-1	-	-	N	R	-1
SR841	State of slave ID 12 in CANopen DS301 communication	-1	-	-	N	R	-1
SR842	State of slave ID 13 in CANopen DS301 communication	-1	-	-	N	R	-1
SR843	State of slave ID 14 in CANopen DS301 communication	-1	-	-	N	R	-1
SR844	State of slave ID 15 in CANopen DS301 communication	-1	-	-	N	R	-1
SR845	State of slave ID 16 in CANopen DS301 communication	-1	-	-	N	R	-1
SR846	State of slave ID 17 in CANopen DS301 communication	-1	-	-	N	R	-1
SR847	State of slave ID 18 in CANopen DS301 communication	-1	-	-	N	R	-1
SR848	State of slave ID 19 in CANopen DS301 communication	-1	-	-	N	R	-1
SR849	State of slave ID 20 in CANopen DS301 communication	-1	-	-	N	R	-1
SR850	State of slave ID 21 in CANopen DS301 communication	-1	-	-	N	R	-1
SR851	State of slave ID 22 in CANopen DS301 communication	-1	-	-	N	R	-1
SR852	State of slave ID 23 in CANopen DS301 communication	-1	-	-	N	R	-1
SR853	State of slave ID 24 in CANopen DS301 communication	-1	-	-	N	R	-1
SR854	State of slave ID 25 in CANopen DS301 communication	-1	-	-	N	R	-1
SR855	State of slave ID 26 in CANopen DS301 communication	-1	-	-	N	R	-1
SR856	State of slave ID 27 in CANopen DS301 communication	-1	-	-	N	R	-1
SR857	State of slave ID 28 in CANopen DS301 communication	-1	-	-	N	R	-1
SR858	State of slave ID 29 in CANopen DS301 communication	-1	-	-	N	R	-1
SR859	State of slave ID 30 in CANopen DS301 communication	-1	-	-	N	R	-1
SR860	State of slave ID 31 in CANopen DS301 communication	-1	-	-	N	R	-1
SR861	State of slave ID 32 in CANopen DS301 communication	-1	-	-	N	R	-1
SR862	State of slave ID 33 in CANopen DS301 communication	-1	-	-	N	R	-1
SR863	State of slave ID 34 in CANopen DS301 communication	-1	-	-	N	R	-1
SR864	State of slave ID 35 in CANopen DS301 communication	-1	-	-	N	R	-1
SR865	State of slave ID 36in CANopen DS301 communication	-1	-	-	N	R	-1
SR866	State of slave ID 37 in CANopen DS301 communication	-1	-	-	N	R	-1
SR867	State of slave ID 38 in CANopen DS301 communication	-1	-	-	N	R	-1
SR868	State of slave ID 39 in CANopen DS301 communication	-1	-	-	N	R	-1
SR869	State of slave ID 40 in CANopen DS301 communication	-1	-	-	N	R	-1
SR870	State of slave ID 41 in CANopen DS301 communication	-1	-	-	N	R	-1
SR871	State of slave ID 42 in CANopen DS301 communication	-1	-	-	N	R	-1
SR872	State of slave ID 43 in CANopen DS301 communication	-1	-	-	N	R	-1
SR873	State of slave ID 44 in CANopen DS301 communication	-1	-	-	N	R	-1
SR874	State of slave ID 45 in CANopen DS301 communication	-1	-	-	N	R	-1
SR875	State of slave ID 46 in CANopen DS301 communication	-1	-	-	N	R	-1
SR876	State of slave ID 47 in CANopen DS301 communication	-1	-	-	N	R	-1

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR877	State of slave ID 48 in CANopen DS301 communication	-1	-	-	N	R	-1
SR878	State of slave ID 49 in CANopen DS301 communication	-1	-	-	N	R	-1
SR879	State of slave ID 50 in CANopen DS301 communication	-1	-	-	N	R	-1
SR880	State of slave ID 51 in CANopen DS301 communication	-1	-	-	N	R	-1
SR881	State of slave ID 52 in CANopen DS301 communication	-1	-	-	N	R	-1
SR882	State of slave ID 53 in CANopen DS301 communication	-1	-	-	N	R	-1
SR883	State of slave ID 54 in CANopen DS301 communication	-1	-	-	N	R	-1
SR884	State of slave ID 55 in CANopen DS301 communication	-1	-	-	N	R	-1
SR885	State of slave ID 56 in CANopen DS301 communication	-1	-	-	N	R	-1
SR886	State of slave ID 57 in CANopen DS301 communication	-1	-	-	N	R	-1
SR887	State of slave ID 58 in CANopen DS301 communication	-1	-	-	N	R	-1
SR888	State of slave ID 59 in CANopen DS301 communication	-1	-	-	N	R	-1
SR889	State of slave ID 60 in CANopen DS301 communication	-1	-	-	N	R	-1
SR890	State of slave ID 61 in CANopen DS301 communication	-1	-	-	N	R	-1
SR891	State of slave ID 62 in CANopen DS301 communication	-1	-	-	N	R	-1
SR892	State of slave ID 63 in CANopen DS301 communication	-1	-	-	N	R	-1
SR893	State of slave ID 64 in CANopen DS301 communication	-1	-	-	N	R	-1
SR900 SR901	Number of samples in the data logger (32-bit)	0	-	-	N	R	0
SR902	Code for the executions of data logger and the memory card (works with SM456). For example, H5AA5: write the sampling data from the data logger into the memory card.	0	-	-	N	R/W	0
*SR1000 *SR1001	Ethernet IP address (32-bit)	-	-	-	H	R/W	0
*SR1002 *SR1003	Ethernet netmask address (32-bit)	-	-	-	H	R/W	0
*SR1004 *SR1005	Ethernet gateway address (32-bit)	-	-	-	H	R/W	0
*SR1006	Duration of the TCP connection (sec.)	-	-	-	H	R/W	30
*SR1007	Ethernet transmission speed	0	-	-	N	R	0
*SR1009	Number of TCP connections	0	-	-	N	R	0
*SR1010	Specific time when to resend through the TCP connection (ms)	-	-	-	N	R/W	20
*SR1011	Connection number of the MODBUS/TCP Server	0	-	-	N	R	0
*SR1012	Connection number of the MODBUS/TCP Client	0	-	-	N	R	0
*SR1013	Connection number of the EtherNet/IP Adapter	0	-	-	N	R	0
*SR1014	Connection number of the EtherNet/IP Scanner	0	-	-	N	R	0
*SR1015	MODBUS/TCP Server packet receiving timeout (ms)	10	-	-	N	R/W	10
SR1100 SR1101	Value of the input packet counter (32-bit)	0	-	-	N	R	0
SR1102	Value of the input octet counter (32-bit)	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
SR1103							
SR1104	Value of the output packet counter (32-bit)	0	-	-	N	R	0
SR1105							
SR1106	Value of the output octet counter (32-bit)	0	-	-	N	R	0
SR1107							
SR1116	Email counter	0	-	-	N	R	0
SR1117	Email error counter	0	-	-	N	R	0
*SR1120	Actual connection time for data exchange through the Ethernet connection 1	0	-	-	N	R	0
*SR1121	Actual connection time for data exchange through the Ethernet connection 2	0	-	-	N	R	0
*SR1122	Actual connection time for data exchange through the Ethernet connection 3	0	-	-	N	R	0
*SR1123	Actual connection time for data exchange through the Ethernet connection 4	0	-	-	N	R	0
*SR1124	Actual connection time for data exchange through the Ethernet connection 5	0	-	-	N	R	0
*SR1125	Actual connection time for data exchange through the Ethernet connection 6	0	-	-	N	R	0
*SR1126	Actual connection time for data exchange through the Ethernet connection 7	0	-	-	N	R	0
*SR1127	Actual connection time for data exchange through the Ethernet connection 8	0	-	-	N	R	0
*SR1128	Actual connection time for data exchange through the Ethernet connection 9	0	-	-	N	R	0
*SR1129	Actual connection time for data exchange through the Ethernet connection 10	0	-	-	N	R	0
*SR1130	Actual connection time for data exchange through the Ethernet connection 11	0	-	-	N	R	0
*SR1131	Actual connection time for data exchange through the Ethernet connection 12	0	-	-	N	R	0
*SR1132	Actual connection time for data exchange through the Ethernet connection 13	0	-	-	N	R	0
*SR1133	Actual connection time for data exchange through the Ethernet connection 14	0	-	-	N	R	0
*SR1134	Actual connection time for data exchange through the Ethernet connection 15	0	-	-	N	R	0
*SR1135	Actual connection time for data exchange through the Ethernet connection 16	0	-	-	N	R	0
*SR1152	The error code for data exchange through the Ethernet connection 1	0	-	-	N	R	0
*SR1153	The error code for data exchange through the Ethernet connection 2	0	-	-	N	R	0
*SR1154	The error code for data exchange through the Ethernet	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	connection 3						
*SR1155	The error code for data exchange through the Ethernet connection 4	0	-	-	N	R	0
*SR1156	The error code for data exchange through the Ethernet connection 5	0	-	-	N	R	0
*SR1157	The error code for data exchange through the Ethernet connection 6	0	-	-	N	R	0
*SR1158	The error code for data exchange through the Ethernet connection 7	0	-	-	N	R	0
*SR1159	The error code for data exchange through the Ethernet connection 8	0	-	-	N	R	0
*SR1160	The error code for data exchange through the Ethernet connection 9	0	-	-	N	R	0
*SR1161	The error code for data exchange through the Ethernet connection 10	0	-	-	N	R	0
*SR1162	The error code for data exchange through the Ethernet connection 11	0	-	-	N	R	0
*SR1163	The error code for data exchange through the Ethernet connection 12	0	-	-	N	R	0
*SR1164	The error code for data exchange through the Ethernet connection 13	0	-	-	N	R	0
*SR1165	The error code for data exchange through the Ethernet connection 14	0	-	-	N	R	0
*SR1166	The error code for data exchange through the Ethernet connection 15	0	-	-	N	R	0
*SR1167	The error code for data exchange through the Ethernet connection 16	0	-	-	N	R	0
*SR1312	Communication code for RTU-EN01 connection 1	0	-	-	N	R	0
*SR1313	Communication code for RTU-EN01 connection 2	0	-	-	N	R	0
*SR1314	Communication code for RTU-EN01 connection 3	0	-	-	N	R	0
*SR1315	Communication code for RTU-EN01 connection 4	0	-	-	N	R	0
SR1318	Socket input counter	0	-	-	N	R	0
SR1319	Socket output counter	0	-	-	N	R	0
SR1320	Socket error counter	0	-	-	N	R	0
*SR1335	Actual cycle time of connection 1–32 for data exchange through COM1	0	-	-	N	R	0
*SR1336	Number of the connection that is currently performing a cyclical data exchange through COM1	0	-	-	N	R	0
*SR1340	Error code for data exchange through the COM1 connection 1	0	-	-	N	R	0
*SR1341	Error code for data exchange through the COM1 connection 2	0	-	-	N	R	0
*SR1342	Error code for data exchange through the COM1 connection 3	0	-	-	N	R	0
*SR1343	Error code for data exchange through the COM1 connection 4	0	-	-	N	R	0
*SR1344	Error code for data exchange through the COM1 connection 5	0	-	-	N	R	0
*SR1345	Error code for data exchange through the COM1 connection 6	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR1346	Error code for data exchange through the COM1 connection 7	0	-	-	N	R	0
*SR1347	Error code for data exchange through the COM1 connection 8	0	-	-	N	R	0
*SR1348	Error code for data exchange through the COM1 connection 9	0	-	-	N	R	0
*SR1349	Error code for data exchange through the COM1 connection 10	0	-	-	N	R	0
*SR1350	Error code for data exchange through the COM1 connection 11	0	-	-	N	R	0
*SR1351	Error code for data exchange through the COM1 connection 12	0	-	-	N	R	0
*SR1352	Error code for data exchange through the COM1 connection 13	0	-	-	N	R	0
*SR1353	Error code for data exchange through the COM1 connection 14	0	-	-	N	R	0
*SR1354	Error code for data exchange through the COM1 connection 15	0	-	-	N	R	0
*SR1355	Error code for data exchange through the COM1 connection 16	0	-	-	N	R	0
*SR1356	Error code for data exchange through the COM1 connection 17	0	-	-	N	R	0
*SR1357	Error code for data exchange through the COM1 connection 18	0	-	-	N	R	0
*SR1358	Error code for data exchange through the COM1 connection 19	0	-	-	N	R	0
*SR1359	Error code for data exchange through the COM1 connection 20	0	-	-	N	R	0
*SR1360	Error code for data exchange through the COM1 connection 21	0	-	-	N	R	0
*SR1361	Error code for data exchange through the COM1 connection 22	0	-	-	N	R	0
*SR1362	Error code for data exchange through the COM1 connection 23	0	-	-	N	R	0
*SR1363	Error code for data exchange through the COM1 connection 24	0	-	-	N	R	0
*SR1364	Error code for data exchange through the COM1 connection 25	0	-	-	N	R	0
*SR1365	Error code for data exchange through the COM1 connection 26	0	-	-	N	R	0
*SR1366	Error code for data exchange through the COM1 connection 27	0	-	-	N	R	0
*SR1367	Error code for data exchange through the COM1 connection 28	0	-	-	N	R	0
*SR1368	Error code for data exchange through the COM1 connection 29	0	-	-	N	R	0
*SR1369	Error code for data exchange through the COM1 connection 30	0	-	-	N	R	0
*SR1370	Error code for data exchange through the COM1 connection 31	0	-	-	N	R	0
*SR1371	Error code for data exchange through the COM1 connection 32	0	-	-	N	R	0
*SR1375	Actual cycle time of connection 1–32 for data exchange through COM2	0	-	-	N	R	0
*SR1376	Number of the connection that is currently performing a cyclical data exchange through COM2	0	-	-	N	R	0
*SR1380	Error code for data exchange through the COM2 connection 1	0	-	-	N	R	0
*SR1381	Error code for data exchange through the COM2 connection 2	0	-	-	N	R	0
*SR1382	Error code for data exchange through the COM2 connection 3	0	-	-	N	R	0
*SR1383	Error code for data exchange through the COM2 connection 4	0	-	-	N	R	0
*SR1384	Error code for data exchange through the COM2 connection 5	0	-	-	N	R	0
*SR1385	Error code for data exchange through the COM2 connection 6	0	-	-	N	R	0
*SR1386	Error code for data exchange through the COM2 connection 7	0	-	-	N	R	0
*SR1387	Error code for data exchange through the COM2 connection 8	0	-	-	N	R	0
*SR1388	Error code for data exchange through the COM2 connection 9	0	-	-	N	R	0
*SR1389	Error code for data exchange through the COM2 connection 10	0	-	-	N	R	0
*SR1390	Error code for data exchange through the COM2 connection 11	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
*SR1391	Error code for data exchange through the COM2 connection 12	0	-	-	N	R	0
*SR1392	Error code for data exchange through the COM2 connection 13	0	-	-	N	R	0
*SR1393	Error code for data exchange through the COM2 connection 14	0	-	-	N	R	0
*SR1394	Error code for data exchange through the COM2 connection 15	0	-	-	N	R	0
*SR1395	Error code for data exchange through the COM2 connection 16	0	-	-	N	R	0
*SR1396	Error code for data exchange through the COM2 connection 17	0	-	-	N	R	0
*SR1397	Error code for data exchange through the COM2 connection 18	0	-	-	N	R	0
*SR1398	Error code for data exchange through the COM2 connection 19	0	-	-	N	R	0
*SR1399	Error code for data exchange through the COM2 connection 20	0	-	-	N	R	0
*SR1400	Error code for data exchange through the COM2 connection 21	0	-	-	N	R	0
*SR1401	Error code for data exchange through the COM2 connection 22	0	-	-	N	R	0
*SR1402	Error code for data exchange through the COM2 connection 23	0	-	-	N	R	0
*SR1403	Error code for data exchange through the COM2 connection 24	0	-	-	N	R	0
*SR1404	Error code for data exchange through the COM2 connection 25	0	-	-	N	R	0
*SR1405	Error code for data exchange through the COM2 connection 26	0	-	-	N	R	0
*SR1406	Error code for data exchange through the COM2 connection 27	0	-	-	N	R	0
*SR1407	Error code for data exchange through the COM2 connection 28	0	-	-	N	R	0
*SR1408	Error code for data exchange through the COM2 connection 29	0	-	-	N	R	0
*SR1409	Error code for data exchange through the COM2 connection 30	0	-	-	N	R	0
*SR1410	Error code for data exchange through the COM2 connection 31	0	-	-	N	R	0
*SR1411	Error code for data exchange through the COM2 connection 32	0	-	-	N	R	0
*SR1553	Once the CPU is supplied with power, it checks if the parameters of the extension module DI are correct.	-	-	-	Y	R/W	0
*SR1554	Once the CPU is supplied with power, it checks if the parameters of the extension module DO are correct.	-	-	-	Y	R/W	0
*SR1555	Once the CPU is supplied with power, it checks if the number of the connected modules is correct.	-	-	-	Y	R/W	0
SR1557	Total DI points of the connected modules (CPU built-in DI points NOT included)	0	-	-	N	R	0
SR1558	Total DO points of the connected modules (CPU built-in DO points NOT included)	0	-	-	N	R	0
*SR1560	Number of the right-side modules connected to the CPU module	0	-	-	N	R	0
*SR1561	Model code of the 1st right-side module connected to the CPU module	0	-	-	N	R	0
*SR1562	Model code of the 2nd right-side module connected to the CPU module	0	-	-	N	R	0
*SR1563	Model code of the 3rd right-side module connected to the CPU module	0	-	-	N	R	0
*SR1564	Model code of the 4th right-side module connected to the CPU module	0	-	-	N	R	0
*SR1565	Model code of the 5th right-side module connected to the CPU module	0	-	-	N	R	0
*SR1566	Model code of the 6th right-side module connected to the CPU module	0	-	-	N	R	0
*SR1567	Model code of the 7th right-side module connected to the CPU	0	-	-	N	R	0

SR	Function	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Latched	Attribute	Default
	module						
*SR1568	Model code of the 8th right-side module connected to the CPU module	0	-	-	N	R	0

Note 1: for SR*, refer to SM/SR table for more details

2.2.15 Special Data Registers Refresh Conditions

Special data register	Time to refresh
SR0–SR2	Refresh the register when there is a program execution error.
SR4–SR6	Refresh the register when there is a grammar check error
SR8–SR9	Refresh the register when there is a watchdog timer error.
SR23	Refresh the register when there is a watchdog timer error.
SR28	Refresh the register when the output is used repeatedly by more than one high-speed output instruction.
SR32	Refresh the register once when there is an error. -1: no error occurred.
SR36	The register is refreshed by you. You set the flag SM36 to ON and the system saves the data to the memory card. After saving is complete, the system automatically resets it to OFF
SR40–SR161	Refresh the register when there is an error.
SR162–SR163	After the manufactured PLC leaves the factory, every minute the PLC is supplied with power is counted and the register is refreshed every minute.
SR180	Refresh the register when the PLC is powered on and powered off.
SR190–SR197	Register is refreshed by you.
SR198–SR199	After a cycle scan when the PLC is powered on.
SR201–SR213	The register is refreshed according to the settings in HWCONFIG when the PLC is powered on. You can edit the settings afterwards.
SR217–SR218	The register is refreshed according to the settings in HWCONFIG when powered on. You can edit the settings afterwards.
SR220–SR226	Refresh the register every scan cycle.
SR227–SR308	Refresh the register when the program is downloaded to the PLC.
SR309–SR390	Refresh the register when the status of the PLC changes.
SR391–SR397	Refresh the register every scan cycle.
SR407	Refresh the register every second.
SR408–SR416	Refresh the register whenever the instruction END is executed.
SR421–SR424	Register is refreshed by you.
SR440–SR443	Refresh the register when the PLC is powered on.
SR444–SR451	Refresh the register when the PLC is powered on.
SR453	Refresh the register when there is an error.
SR460	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR462–SR466	Register is refreshed by you.
SR467	Register is refreshed by the system.
SR468–SR469	When the PLC is supplied with power, the register is refreshed according to the position planning

Special data register	Time to refresh
	table. You can edit the settings afterwards.
SR470	Register is refreshed by the system.
SR472	Register is refreshed you.
SR474	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR476–SR477	Register is refreshed by you.
SR480	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR482–SR486	Register is refreshed by you.
SR487	Register is refreshed by the system.
SR488–SR489	Refresh the register when the PLC is powered on according to the position planning table. You can edit the settings afterwards.
SR490	Register is refreshed by the system.
SR492	Register is refreshed by you.
SR494	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR496–SR497	Register is refreshed by you.
SR500	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR502–SR506	Register is refreshed by you.
SR507	Register is refreshed by the system.
SR508–SR509	Refresh the register when the PLC is powered on according to the position planning table. You can edit the settings afterwards.
SR510	Register is refreshed by the system.
SR512	Register is refreshed by you.
SR514	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR516–SR517	Register is refreshed by you.
SR520	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the settings.
SR522–SR526	Register is refreshed by you.
SR527	Register is refreshed by the system.
SR528–SR529	Refresh the register when the PLC is powered on according to the position planning table. You can edit the settings afterwards.
SR530	Register is refreshed by the system.
SR532	Register is refreshed by you.
SR534	Refresh the register whenever the high-speed output instruction is executed and the program is scanned. If the instruction is not executed, you can edit the setting.
SR536–SR537	Register is refreshed by a user.
SR580–SR595	Refresh the register according to the settings in HWCONFIG when the PLC is powered on. You can edit the settings afterwards.
SR604–SR607	Register is refreshed by you.
SR610–SR617	Refresh the register whenever the output instruction is executed.
SR623–SR634	Refresh the register whenever the EIX or DIX instruction is executed. ON: interrupt is enabled OFF: interrupt is disabled
SR640–SR647	Register is refreshed by you.

Special data register	Time to refresh
SR652–SR655	Register is refreshed by the system.
SR658–SR726	Register is refreshed by the system.
SR731–SR738	Register is refreshed by the system.
SR741–SR748	Register is refreshed by the system.
SR751-SR768	Register is refreshed by you.
SR771–SR778	Register is refreshed by the system.
SR781–SR788	Register is refreshed by the system.
SR791–SR806	Register is refreshed by the system.
SR811–SR818	Register is refreshed by you.
SR820	Refresh the register according to the settings in CANopen Builder.
SR821	Refresh the register when the firmware is updated.
SR822	Refresh the register according to the settings in HWCONFIG.
SR823–SR893	Register is refreshed by the system.
SR900–SR901	Register is refreshed by the system.
SR902	Register is refreshed by you.
SR1000–SR1006	1. Register is refreshed by you. 2. Register is refreshed after power-on or after HWCONFIG is downloaded.
SR1007	Register is refreshed by the system.
SR1009	Register is refreshed by the system.
SR1010	Register is refreshed by you.
SR1011–SR1014	Register is refreshed by the system.
SR1015	Register is refreshed by you.
SR1020–SR1107	1. Refresh the register when a connection is established. 2. Refresh the register every scan cycle.
SR1116–SR1117	Refresh the register when the program is downloaded to the PLC.
SR1120–SR1183	Refresh the register when the communication is complete.
SR1312-SR1315	Refresh the register during communication
SR1318–SR1320	Refresh the register when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1335–SR1336	Refresh the register every scan cycle after the function of data exchange is enabled.
SR1340–SR1371	Refresh the register when there is an error.
SR1375–SR1376	Refresh the register every scan cycle after the function of data exchange is enabled.
SR1380–SR1411	Refresh the register when there is an error.
SR1553~SR1555	Register is refreshed by you.
SR1556~SR1568	Register is refreshed by the system.

2.2.16 Additional Remarks on Special Auxiliary Relays and Special Data Registers

1. Scan timeout timer

- SM8/SR8

  When a scan timeout occurs during the execution of the program, the error LED on the PLC changes to continuous ON, and SM8 changes to ON.

  SR8 contains the step address at which the watchdog timer changed to ON.

2. Clearing the warning light

- SM22

  Setting SM22 to ON clears the error log and the warning light.

3. The real-time clock

- SM220, SR220–SR226, and SR391–SR397

  SM220: Calibrate the real-time clock within ±30 seconds

  When SM220 changes from OFF to ON, the system calibrates the real-time clock.

  If the second value in the real-time clock is in the range 0–29, the minutes value is fixed, and the seconds value is cleared to zero.

  If the value of the second in the real-time clock is in the range 30–59, the minutes value increases by one, and the seconds value is cleared to zero.

  This table lists the corresponding functions and values of SR220–SR226 and SR391–SR397.

Device		Function	Value
Binary-coded decimal system	Decimal system		
SR220	SR391	Year	00–99 (A.D.)
SR221	SR392	Month	1–12
SR222	SR393	Day	1–31
SR223	SR394	Hour	0–23
SR224	SR395	Minute	0–59
SR225	SR396	Second	0–59
SR226	SR397	Week	1–7

SR391–SR397 correspond to SR220–SR226. The difference between SR220–SR226 and SR391–SR397 is that the former uses the binary-coded decimal while the latter uses the decimal system. For example, December is represented as 12 in SR392 while it is represented as 1100 in the binary-coded decimal. Refer to Chapter 6 for more information on the real-time clock.

4. Communication functions

- SM96–SM107, SM209–SM212, SR201–SR202, and SR209–SR213

  SR209 records the communication format of the COM1 port on the PLC, and SR212 records the communication format of the COM2 port on the PLC. The following table lists the settings for the communication protocols. Refer to Chapter 6 for more information on the communication instructions.

b0	Data length			7 (value=0)			8 (value=1)
b1 **b2**	Parity bits			00	:	None	
				01	:	Odd parity bits	
				10	:	Even parity bits	
b3	Stop bits			1 bit (value=0)			2 bits (value=1)
b4 **b5**	0001	(H 1)	:	4800			
	0010	(H 2)	:	9600			

b6 **b7**	0011	(H 3)	:	19200	
	0100	(H 4)	:	38400	
	0101	(H 5)	:	57600	
	0110	(H 6)	:	115200	
	0111	(H 7)	:	230400	
	1000	(H 8)	:	500000	
	1001	(H 9)	:	921000	
	1010	(16#A)	:	Undefined	
	1011	(16#B)	:	Undefined	
	1100	(16#C)	:	Undefined	
	1101	(16#D)	:	Undefined	
	1110	(16#E)	:	Undefined	
	1111	(16#F)	:	User-defined*1	
b8-b15	Undefined (reserved)				

*1: Refer to the HWCONFIG settings in ISPSoft for the user-defined baudrate.
Use SR217/SR218 to set user-defined baudrate (unit: 100bps); when you input 96 in SR217/SR218, it means 9600bps.

*2: Refer to section 6.19.3 for the use of communication flags and registers.

5. Clearing the device contents

- SM204/SM205

Device number	Device which is cleared
SM204 All non-latched areas are cleared.	The non-latched areas in the input relays, the output relays, the stepping relays, and the auxiliary relays are cleared. The non-latched areas in the timers, the counters, and the 32-bit counters are cleared. The non-latched areas in the data registers and the index registers are cleared. The watchdog timer does not act during this period of time.
SM205 All latched areas are cleared.	The latched areas in the timers, counters, and 32-bit counters are cleared. The latched auxiliary relays are cleared. The latched data registers are cleared. The watchdog timer does not act during this period of time.

Refer to Section 2.1.4 for more information on the latched areas in the device range.

6. PLC error log

- SR40-SR161

SR40: The maximum number of error logs stored in SR40 is 20. Each error log occupies 6 registers.

SR41: The error log pointer points to the latest error log. When an error occurs, the value of the error log pointer increases by one. The range of pointer values is 0–19. For example, the error log pointer points to the fourth error log when the value in SR41 is 3.

The time of the error and the position where the error occurs are recorded in SR42–SR161. The following table lists the corresponding functions of these data registers.

Number	Slot	Module	Error	Time when the error occurs

		ID	code	Year	Month	Day	Hour	Minute	Second
1	SR42 Low byte	SR43	SR44	SR45 High byte	SR45 Low byte	SR46 High byte	SR46 Low byte	SR47 High byte	SR47 Low byte
2	SR48 Low byte	SR49	SR50	SR51 High byte	SR51 Low byte	SR52 High byte	SR52 Low byte	SR53 High byte	SR53 Low byte
3	SR54 Low byte	SR55	SR56	SR57 High byte	SR57 Low byte	SR58 High byte	SR58 Low byte	SR59 High byte	SR59 Low byte
4	SR60 Low byte	SR61	SR62	SR63 High byte	SR63 Low byte	SR64 High byte	SR64 Low byte	SR65 High byte	SR65 Low byte
5	SR66 Low byte	SR67	SR68	SR69 High byte	SR69 Low byte	SR70 High byte	SR70 Low byte	SR71 High byte	SR71 Low byte
6	SR72 Low byte	SR73	SR74	SR75 High byte	SR75 Low byte	SR76 High byte	SR76 Low byte	SR77 High byte	SR77 Low byte
7	SR78 Low byte	SR79	SR80	SR81 High byte	SR81 Low byte	SR82 High byte	SR82 Low byte	SR83 High byte	SR83 Low byte
8	SR84 Low byte	SR85	SR86	SR87 High byte	SR87 Low byte	SR88 High byte	SR88 Low byte	SR89 High byte	SR89 Low byte
9	SR90 Low byte	SR91	SR92	SR93 High byte	SR93 Low byte	SR94 High byte	SR94 Low byte	SR95 High byte	SR95 Low byte
10	SR96 Low byte	SR97	SR98	SR99 High byte	SR99 Low byte	SR100 High byte	SR100 Low byte	SR101 High byte	SR101 Low byte
11	SR102 Low byte	SR103	SR104	SR105 High byte	SR105 Low byte	SR106 High byte	SR106 Low byte	SR107 High byte	SR107 Low byte
12	SR108 Low byte	SR109	SR110	SR111 High byte	SR111 Low byte	SR112 High byte	SR112 Low byte	SR113 High byte	SR113 Low byte
13	SR114 Low byte	SR115	SR116	SR117 High byte	SR117 Low byte	SR118 High byte	SR118 Low byte	SR119 High byte	SR119 Low byte
14	SR120 Low byte	SR121	SR122	SR123 High byte	SR123 Low byte	SR124 High byte	SR124 Low byte	SR125 High byte	SR125 Low byte
15	SR126 Low byte	SR127	SR128	SR129 High byte	SR129 Low byte	SR130 High byte	SR130 Low byte	SR131 High byte	SR131 Low byte
16	SR132 Low byte	SR133	SR134	SR135 High byte	SR135 Low byte	SR136 High byte	SR136 Low byte	SR137 High byte	SR137 Low byte
17	SR138 Low byte	SR139	SR140	SR141 High byte	SR141 Low byte	SR142 High byte	SR142 Low byte	SR143 High byte	SR143 Low byte
18	SR144 Low byte	SR145	SR146	SR147 High byte	SR147 Low byte	SR148 High byte	SR148 Low byte	SR149 High byte	SR149 Low byte
19	SR150 Low byte	SR151	SR152	SR153 High byte	SR153 Low byte	SR154 High byte	SR154 Low byte	SR155 High byte	SR155 Low byte
20	SR156 Low byte	SR157	SR158	SR159 High byte	SR159 Low byte	SR160 High byte	SR160 Low byte	SR161 High byte	SR161 Low byte

7. PLC download log

- SR227-SR308

SR227: The maximum number of download logs which are stored in SR227 is 20. Every download log occupies 4 registers. The download actions which are recorded are numbered, as shown in the following table.

Download action	Number
Downloading the program	1

Download action	Number
Downloading the PLC setting	2
Downloading the module table	3

SR228: The download log pointer points to the latest download log. When a download action is executed, the value of the download log pointer increases by one. The range of pointer values is 0–19. For example, the download log pointer points to the fourth download log when the value in SR228 is 3.

The time of the downloading and the action numbers are recorded in SR229–SR30. The following table lists the corresponding functions of these data registers.

Number	Action number	*Time when the download occurs					
		Year	Month	Day	Hour	Minute	Second
1	SR229	SR230 High byte	SR230 Low byte	SR231 High byte	SR231 Low byte	SR232 High byte	SR232 Low byte
2	SR233	SR234 High byte	SR234 Low byte	SR235 High byte	SR235 Low byte	SR236 High byte	SR236 Low byte
3	SR237	SR238 High byte	SR238 Low byte	SR239 High byte	SR239 Low byte	SR240 High byte	SR240 Low byte
4	SR241	SR242 High byte	SR242 Low byte	SR243 High byte	SR243 Low byte	SR244 High byte	SR244 Low byte
5	SR245	SR246 High byte	SR246 Low byte	SR247 High byte	SR247 Low byte	SR248 High byte	SR248 Low byte
6	SR249	SR250 High byte	SR250 Low byte	SR251 High byte	SR251 Low byte	SR252 High byte	SR252 Low byte
7	SR253	SR254 High byte	SR254 Low byte	SR255 High byte	SR255 Low byte	SR256 High byte	SR256 Low byte
8	SR257	SR258 High byte	SR258 Low byte	SR259 High byte	SR259 Low byte	SR260 High byte	SR260 Low byte
9	SR261	SR262 High byte	SR262 Low byte	SR263 High byte	SR263 Low byte	SR264 High byte	SR264 Low byte
10	SR265	SR266 High byte	SR266 Low byte	SR267 High byte	SR267 Low byte	SR268 High byte	SR268 Low byte
11	SR269	SR270 High byte	SR270 Low byte	SR271 High byte	SR271 Low byte	SR272 High byte	SR272 Low byte
12	SR273	SR274 High byte	SR274 Low byte	SR275 High byte	SR275 Low byte	SR276 High byte	SR276 Low byte
13	SR277	SR278 High byte	SR278 Low byte	SR279 High byte	SR279 Low byte	SR280 High byte	SR280 Low byte
14	SR281	SR282 High byte	SR282 Low byte	SR283 High byte	SR283 Low byte	SR284 High byte	SR284 Low byte
15	SR285	SR286 High byte	SR286 Low byte	SR287 High byte	SR287 Low byte	SR288 High byte	SR288 Low byte
16	SR289	SR290 High byte	SR290 Low byte	SR291 High byte	SR291 Low byte	SR292 High byte	SR292 Low byte
17	SR293	SR294 High byte	SR294 Low byte	SR295 High byte	SR295 Low byte	SR296 High byte	SR296 Low byte
18	SR297	SR298	SR298	SR299	SR299	SR300	SR300

Number	Action number	*Time when the download occurs					
		Year	Month	Day	Hour	Minute	Second
		High byte	Low byte	High byte	Low byte	High byte	Low byte
19	SR301	SR302 High byte	SR302 Low byte	SR303 High byte	SR303 Low byte	SR304 High byte	SR304 Low byte
20	SR305	SR306 High byte	SR306 Low byte	SR307 High byte	SR307 Low byte	SR308 High byte	SR308 Low byte

* The format for the download action time: the data is stored as a binary-coded decimal. The following table lists the range of values.

Function	Value
Year	00–99 (A.D.)
Month	01–12
Day	01–31
Hour	00–23
Minute	00–59
Second	00–59

8. The PLC status change log

- SR309–SR390

SR309: The maximum number of PLC status change logs stored in SR309 is 20. Each PLC status change log occupies 4 registers. The recorded PLC status change actions are numbered, as shown in the following table.

PLC status change	Number
PLC powered on.	1
PLC disconnected.	2
PLC starts to run.	3
PLC stops.	4
Default setting in the PLC	5

SR310: The PLC status change log pointer points to the latest PLC status change log. When the PLC status changes, the value of the PLC status change log pointer increases by one. The range of pointer values is 0–19. For example, PLC status change log pointer points to the fourth PLC status change log when the value in SR310 is 3.

The time when the PLC status change actions occur is recorded in SR311–SR390. The flowing table lists the corresponding functions of these data registers.

Number	Action number	*Time when the PLC status change occurs					
		Year	Month	Day	Hour	Minute	Second
1	SR311	SR312 High byte	SR312 Low byte	SR313 High byte	SR313 Low byte	SR314 High byte	SR314 Low byte
2	SR315	SR316 High byte	SR316 Low byte	SR317 High byte	SR317 Low byte	SR318 High byte	SR318 Low byte
3	SR319	SR320	SR320	SR321	SR321	SR322	SR322

Number	Action number	*Time when the PLC status change occurs					
		Year	Month	Day	Hour	Minute	Second
		High byte	Low byte	High byte	Low byte	High byte	Low byte
4	SR323	SR324 High byte	SR324 Low byte	SR325 High byte	SR325 Low byte	SR326 High byte	SR326 Low byte
5	SR327	SR328 High byte	SR328 Low byte	SR329 High byte	SR329 Low byte	SR330 High byte	SR330 Low byte
6	SR331	SR332 High byte	SR332 Low byte	SR333 High byte	SR333 Low byte	SR334 High byte	SR334 Low byte
7	SR335	SR336 High byte	SR336 Low byte	SR337 High byte	SR337 Low byte	SR338 High byte	SR338 Low byte
8	SR339	SR340 High byte	SR340 Low byte	SR341 High byte	SR341 Low byte	SR342 High byte	SR342 Low byte
9	SR343	SR344 High byte	SR344 Low byte	SR345 High byte	SR345 Low byte	SR346 High byte	SR346 Low byte
10	SR347	SR348 High byte	SR348 Low byte	SR349 High byte	SR349 Low byte	SR350 High byte	SR350 Low byte
11	SR351	SR352 High byte	SR352 Low byte	SR353 High byte	SR353 Low byte	SR354 High byte	SR354 Low byte
12	SR355	SR356 High byte	SR356 Low byte	SR357 High byte	SR357 Low byte	SR358 High byte	SR358 Low byte
13	SR359	SR360 High byte	SR360 Low byte	SR361 High byte	SR361 Low byte	SR362 High byte	SR362 Low byte
14	SR363	SR364 High byte	SR364 Low byte	SR365 High byte	SR365 Low byte	SR366 High byte	SR366 Low byte
15	SR367	SR368 High byte	SR368 Low byte	SR369 High byte	SR369 Low byte	SR370 High byte	SR370 Low byte
16	SR371	SR372 High byte	SR372 Low byte	SR373 High byte	SR373 Low byte	SR374 High byte	SR374 Low byte
17	SR375	SR376 High byte	SR376 Low byte	SR377 High byte	SR377 Low byte	SR378 High byte	SR378 Low byte
18	SR379	SR380 High byte	SR380 Low byte	SR381 High byte	SR381 Low byte	SR382 High byte	SR382 Low byte
19	SR383	SR384 High byte	SR384 Low byte	SR385 High byte	SR385 Low byte	SR386 High byte	SR386 Low byte
20	SR387	SR388 High byte	SR388 Low byte	SR389 High byte	SR389 Low byte	SR390 High byte	SR390 Low byte

*Format for the PLC status change time: the data is stored as a binary-coded decimal. The following table lists the range of values.

Function	Value
Year	00–99 (A.D.)
Month	01–12
Day	01–31
Hour	00–23
Minute	00–59
Second	00–59

9. The PLC operation flag

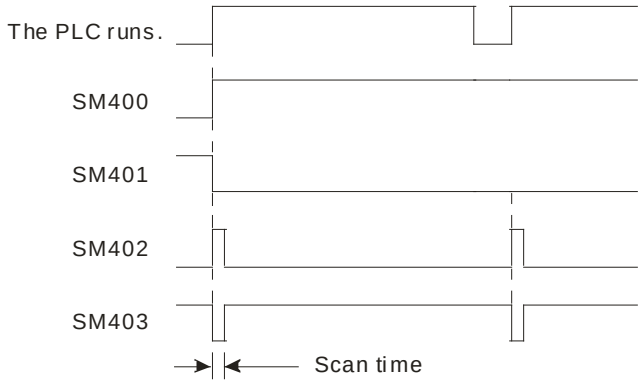
- SM400-SM403

SM400: The flag is always ON when CPU runs.

SM401: The flag is always OFF when CPU runs.

SM402: The flag is ON only at the first scan. The pulse width equals one scan time. You can use this contact for the initial value setting.

SM403: The flag is OFF only at the first scan. That is, the negative pulse is generated the moment the PLC runs.



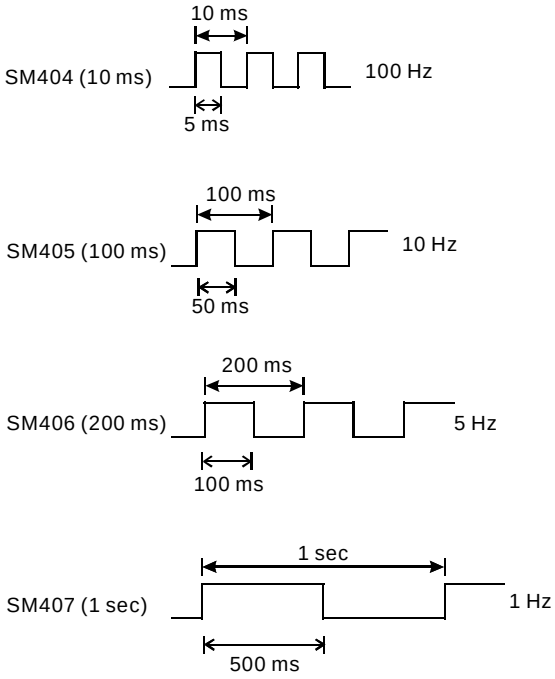
10. The initial clock pulse

- SM404 · SM405 · SM406 · SM407

The PLC provides seven types of clock pulses. When the PLC is powered on, the seven types of clock pulses act automatically.

Device	Function
SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and then is OFF for 5 milliseconds
SM405	100 millisecond clock pulse during which the pulse is ON for 50 milliseconds and then is OFF for 50 milliseconds
SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and then is OFF for 100 milliseconds
SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and then is OFF for 500 milliseconds

The clock pulses are illustrated in the following graphs.



11. The flags related to the memory card

- SM36, SM450-SM453, SM456-SR36, SR453, SR902

You use the memory card to back up the data in the PLC. Refer to Chapter 6 for instructions concerning the memory card.

Device	Function
SM36	Enable saving data to the memory card. When ON, the PLC runs according to the value in the SR36.
SM450	Memory card is present. ON: memory card is present. OFF: memory card is not present.
SM451	Memory card is write-protected.
SM452	Data in the memory card is being accessed. ON: data in the memory card is being accessed. OFF: data in the memory card is not accessed.
SM453	Error during the operation of the memory card. ON: error occurred. OFF: no error.
SM456	Execution of data logger and the memory card. ON: execution by the values in SR902.
SR453	When an error occurs in the memory card, the error codes are stored here. A. 16#005D: no memory card installed B. 16#005E: an error occurs when the memeory card is initialized C. 16#005F: the path is incorrect or the file does not exist D. 16#0060: cannot create the default folder E. 16#0061: the memory space is insufficient F. 16#0062: the memory card is write protected G. 16#0063: an error occurs when the data is written into the file H. 16#0064: the data cannot be read from the memeory card

Device	Function
	I. 16#0065: the file is a read-only file J. 16#0066: when restoring the memory card PLC is in RUN status K. 16#0067: when restoring the memory card PLC ID is not matched when restoring the memory card L. 16#0068: when restoring the memory card the password is not matched M. 16#0069: when restoring the memory card, the file path and the file name contain the CPU module name, but it is not matched with the actual CPU module
SR902	The code for the executions of data logger and the memory card (works with SM456); for example, H5AA5: write the sampling data from the data logger into the memory card.
SR36	Use the following function codes to backup or restore data. Enable SM36 for the function codes to work. A. K1122 (backup) #1, it indicates backing up the PLC programs to the memory card. (PLC ID and password are included) Execute this backup when the PLC is in the RUN/Stop status. Default backup file name: SDCard\PLC CARD\ES3\SysDup\user_program.dup B. K2211 (restore) #2, it indicates restoring the PLC programs from the memeory card. (PLC ID and password are included) Execute the restoring when the PLC is in the Stop status. Default restore file name: SDCard\PLC CARD\ES3\SysDup\user_program.dup C. K3344 (backup) #1, it indicates backing up the latched values (D, S, M, C, HC) and the latched range to the memory card. Execute this backup when the PLC is in the RUN/Stop status. Default backup file name: SDCard\PLC CARD\ES3\SysDup\retained_data.dup D. K4433 (restore) #2, it indicates resoring the latched values (D, S, M, C, HC) and the latched range from the memory card. Execute the restoring when the PLC is in the Stop status. Default restore file name: SDCard\PLC CARD\ES3\SysDup\retained_data.dup E. K5566 (backup) #1, it indicates backing up the PLC programs and parameters to the memory card. (the status and values of latched devices are NOT included). Execute this backup when the PLC is in the Stop status. F. K5577 (backup) #1, it indicates restoring the PLC programs and parameters from the memory card. (the status and values of latched devices are included). Execute this backup when the PLC is in the Stop status. *E & F functions are the same as the backup and resotre functions in ISPSoft Card Utility. Default file name: SDCard\PLC CARD\ES3\SysDup\ES3_save_backup.dup G. K6755 (restore) #2, it indicates restoring all PLC data (according to the backup value E or F) from the memeory card. Execute the restoring when the PLC is in the Stop status. Default file name: SDCard\PLC CARD\ES3\SysDup\ES3_save_backup.dup Note: if you need to restore from POWER-ON, you need to modify the backup file name to SDCard\PLC CARD\ES3\SysDup\ES3_BACKUP.dup before performing restoring from POWER-ON. H. K5588 (backup) #1, it indicates backing up all FR file registers from the PLC to the memory card. Execute this backup when the PLC is in the Stop status. Default file name: SDCard\PLC CARD\ES3\SysDup\ES3_FR_backup.dup I. K8855 (restore) #2, it indicates restoring all FR file registers from the memory card to the PLC. Execute the restoring when the PLC is in the Stop status. The file name should be the same as the file name in H shows. (Default file name: SDCard\PLC CARD\ES3\SysDup\ES3_FR_backup.dup)

#1: While backing up, if there is a file with the same file name, the system overwrites the existing file.

#2: While restoring, if the file path and the file name are not identical to the backup file, the system does not excute the restoring.

- **While restoring the PLC program from the memeory card, the system verifies the backup file, PLC ID and password.**
  - PLC ID check: if the backup file is set with PLC ID, it prevents restoring the PLC program to the wrong PLC.

	CPU module with PLC ID	CPU module without PLC ID
Backup file with PLC ID	PLC ID matched: checking the	Not restoring

	password PLC ID not matched: not restoring	
Backup file without PLC ID	Not restoring	Checking the password

◆ Password check: if the backup file is set with password, it ensures the security of the PLC program and prevents restoring the PLC program to the wrong PLC.

	CPU module with password	CPU module without password
Backup file with password	Password matched: restoring Password not matched: not restoring	Restoring and you can also set the CPU module with password
Backup file without password	Not restoring	Restoring

- SR36 only stores 2 logs：

A. The number 1234 means the PLC stores the error log (SR40–SR161) in the memory card.
B. The number 3456 means the PLC stores the error log (SR40–SR161) and the PLC state changing log (SR309–SR390) in the memory card.

12. The high-speed output instruction is being executed. The output immediately stops when the instruction is disabled or stops.

- SM587: while using the position planning table to control the output, the axis should refer to the following flags for the action to stop.
- SM476, SM477, SM496, SM497, SM516, SM517, SM536, SM537, SM556, SM557, SM576, SM577:

OFF (default): deceleration stop

ON: immediate stop

- SM463, SM474, SM483, SM494, SM503, SM514, SM523, SM534, SM543, SM554, SM563, SM574：use these flags to pause output.
  When the flag changes from OFF to ON, it means stop the output. This works with these flags; refer to the section above for more actions on stop.

When the flag changes from ON to OFF, it means execute the rest of the outputs.

- The flag is for position output axis 1-4 (even number high speed pulse output points) in ramp up / ramp down area.
- SM586

OFF (default): use high-intensity 250 steps to switch to different speeds in the ramp up/ramp down area and the error rate of switching time is about 10%.

ON：use high-intensity 25 steps to switch to different speeds in the ramp up/ramp down area and the error rate of switching time is about 1%.

13. Position control output limit in ISPSoft

- SR580-SR595

Positive output limit: set the limit in ISPSoft. When the output position is greater than the positive limit, the output stops immediately.

Negative output limit: set the limit in ISPSoft. When the output position is smaller than the negative limit, the output stops immediately.

When the positive and negative output limits are both 0, the function is disabled. This function works with the output instructions. The system only checks the limit set in ISPSoft when the instruction is executed. Thus the system does not come to an immediate stop even when it is beyond the output limit. If an immediate stop is needed, using the external input as the way to check the limit is recommended.

14. S curve mode

- SR604-SR607, SM468, SM488, SM508, SM528

There are 3 S curves, small, medium and large. The range is between 0–2. When the value exceeds the range, the system treats the value as the minimum 0 or the maximum 2.

The S curve mode works with the flags such as SM468, SM488, SM568. If the flag is ON, the parameters of the S curve are executed by the output instruction.

15. Backlash compensation function

- SR478, SR479, SR498, SR499, SR518, SR519, SR538, SR539

  For DVP-ES3 Series, up to 8 high-speed outputs (Y0-Y7) can be set. Each output works with a corresponding SR for users to set the output number for backlash compensations. The setting range is 0-32767. If the setting value is <= 0, this function is disabled.

- The output instrucitons that support odd number axis outputs and are directional output by default are JOG, DZRN, DPLSV, DDRVI, DDRVA, DPPMR, DPPMA, DCICR, DCICA, DCICCR, DCICCA, DCCMR, DCCMA, DPPGB and TPO. For even number axis outputs, you can use the following instructions JOG, DPLSV, DDRVI and DDRVA.

16. Flags to show the output number of the current positioning output in the ramp up / ramp down area.

- SR652-653 (32-bit value)

  The output number in the ramp up area: the accumulated output number counted from the acceleration starting point to the target speed point (not included).

- SR654-655 (32-bit value)

  The output number in the ramp down area: the accumulated output number counted from the target speed point (not included) to the output stopping point.

17. Flags to show the CAN communication quality

- SR823
  Hardware receiving error counter for CAN communication: when the error number exceeds 127, it indicates the hardware error occurance is too frequent, you need to check the communication cables or adjust the transmission speed or modify the sample points manually.

- SR824

  Hardware sending error counter for CAN communication: when the error number exceeds 127, it indicates the hardware error occurance is too frequent, you need to check the communication cables or adjust the transmission speed or modify the sample points manually.

18. Ethernet IP related flags

- SM1000, SR1000~SR1015, and SR1536~SR1540.

SM/SR	Function	Action
SM1000	Ethernet setting flag.	ON: the values in SR1000–SR1006 are written into the flash memory. After writing the values, the PLC resets it to OFF. NOTE 1: do not set the flag to ON continuously to avoid damage to the flash memory. NOTE 2: the PLC must be in STOP state before writing the values into the flash memory.
SR1000 SR1001	Ethernet IP address (32-bit)	For example: 192.168.1.5, SR1000 is 16#C0A8 and SR1001 is 0105.
SR1002 SR1003	Ethernet netmask address (32-bit)	For example: 255.255.255.0, SR1002 is 16#FFFF, and SR1003 is FF00
SR1004 SR1005	Ethernet gateway address (32-bit)	For example: 192.168.1.1, SR1004 is 16#C0A8, and SR1005 is 0101
SR1006	Duration of the TCP connection	Unit: second; default: 30 seconds; setting range: 1-32000; if the setting value exceeds the setting range, when When the value exceeds the range, the system treats the value as the minimum 1 or the maximum 32000.
SR1007	Current Ethernet speed	Unit: Mbps
SR1009	Current TCP connection number	Number of all the TCP connections
SR1010	Time to restart TCP communication	Unit: ms; default: 20 ms; setting range: 0-32000; when the value exceeds the range, the system treats the value as the minimum 0 or the maximum 32000. If the setting value is 0, it indicates system restarts communication automatically.
SR1011	MODBUS/TCP Server connection number	The system is refreshed automatically.
SR1012	MODBUS/TCP Client connection number	The system is refreshed automatically.
SR1013	EtherNet/IP Adapter connection number	The system is refreshed automatically.
SR1014	EtherNet/IP Scanner connection number	The system is refreshed automatically.
SR1015	MODBUS/TCP Server packet receiving timeout	Unit: ms; default: 10 ms; setting range: 0-32000; when the value exceeds the range, the system treats the value as the minimum 0 or the maximum 32000

- SM1090, SM1091, SM1106-SM1109

SM	Function	Action
SM1090	TCP connection is busy.	ON: TCP connection timeout
SM1091	UDP connection is busy.	ON: UDP connection timeout
SM1110	AS-FEN02 Ethernet - Filter Setting Error	When the program is downloaded to the PLC
SM1106	Ethernet connection error	ON: PHY initialization fails. OFF: PHY initialization succeeds.
SM1107	Ethernet basic setting error	ON: basic setting error OFF: correct basic setting
SM1109	TCP/UDP socket local port is already used.	ON: the same port in use

For the error codes, the corresponding LED indicators, and other troubleshooting, Refer to Chapter 12 in the DVP-ES3

Series Operation Manual.

19. Email settings

- SM1113, SM1116-SM1155

If the sending an email fails, the flag of the email service error SM1113 is ON.

The following table lists the triggers for sending email and the corresponding flags (SM1116–SM1155).

Item / Function	Trigger 1	Trigger 2	Trigger 3	Trigger 4
Email service	SM1116	SM1126	SM1136	SM1146
	ON: enabled, OFF: disabled			
Email sending	SM1117	SM1127	SM1137	SM1147
	ON: sending email now, OFF: email sent			
Emails sent successfully	SM1119	SM1129	SM1139	SM1149
	ON: email sent successfully			
Email sending error 1	SM1120	SM1130	SM1140	SM1150
	The email cannot be sent due to email content error			
SMTP response timeout	SM1122	SM1132	SM1142	SM1152
	After sending the email, the SMTP server response timeout.			
SMTP server response error	SM1123	SM1133	SM1143	SM1153
	After sending the email, the SMTP server response error.			
Email sending error 2 of	SM1124	SM1134	SM1144	SM1154
	After sending the email, the size of the attachment exceeds the limit.			
Email sending Error 3	SM1125	SM1135	SM1145	SM1155
	After sending the email, the attachment is not found, SM1125 is ON.			

20. Flags and registers concerning data exchange

- Flags for data exchange through COM1 connections

SM	Type	Function
SM750	R/W	Data exchange through COM1 enabled by ISPSoft.
SM752–SM783	R/W	Connection 1–32 through COM1 for data exchange started.
SM784–SM815	R	Data received through COM 1 connection 1–32 for data exchange.
SM816–SM847	R	Error in the COM1 connection 1–32 for data exchange.

- Flags for data exchange through COM2 connections

SM	Type	Function

SM862	R/W	Data exchange through COM2 enabled by ISPSoft.
SM864–SM895	R/W	Connection 1–32 through COM2 for data exchange started.
SM896–SM927	R	Data received through COM2 connection 1–32 for data exchange.
SM928–SM959	R	Error in the COM2 connection 1–32 for data exchange

- Registers for data exchange through COM1 and COM2 connections

SR	Function
SR1335	Actual cycle time of connection 1–32 for data exchange through COM1
SR1336	Number of the connection that is currently performing a cyclical data exchange through COM1
SR1340–SR1371	Error code for data exchange through the COM1 connection 1–32
SR1375	Actual cycle time of connection 1–32 for data exchange through COM2
SR1376	Number of connection that is currently performing a cyclical data exchange through COM2
SR1380–SR1411	Error code for data exchange through the COM2 connection 1–32

The error codes 1–7 are the standard response error codes of the Modbus protocol. Error code 9 means timeout.

- Flags for data exchange through Ethernet connections

SM	Type	Function
SM1167	R/W	Data exchange through Ethernet port enabled by ISPSoft.
SM1168–SM1183	R/W	Connection 1–16 through Ethernet port for data exchange started.
SM1200–SM1215	R	Data received through Ethernet port connection 1–16 for data exchange.
SM1232–SM1247	R	Error in the Ethernet port connection 1–16 for data exchange

- Error codes for Ethernet connections

Error code	Description
16#00XX	Remote module response error Code 1~7 are standard Modbus TCP protocol error codes
16#F000	Ethernet connection is not established
16#F001	Remote module response timeout
16#F003	TCP connection timeout
16#F007	Response error
16#F009	Connection lost in the remote module

- The list of SM/SR states when connecting to RTU-EN01 throught Ethernet port

SM / SR		Description
SM1312 - SM1315		The communication state flags of RTU-EN01 connection ID 1-4
SR1312 - SR1315		The communication state codes of RTU-EN01 connection ID 1-4
SR	**SM state**	**Description**
0	Off	Connection closed
1	On	Successful connection
2	Off	Communiction timeout
3	Off	Connection closed by force
4	Off	RTU-EN01 response: error in contents
5	Off	RTU-EN01 response: error
6	Off	Network not connected or connection failed

SM state	RX & RCR Read data (input mapping area)	RY & RCR Write data (output mapping area)
Off → On	Clear to 0	Clear to 0
On → Off	No change	No change

Note: Before the connection is established, it is suggested NOT to use data (RX/RY/RCR Read/RCR Write) in the mapping area.

21. You can set SM227 to ON and use SR1553, SR1554, SR1555 to check if the parameters of the extension module DI/DI and the number of the connected modules are correct, once the CPU is supplied with power. If the result is incorrect, the CPU stops running and store the error (16#1414) in SR180 and SM10 = ON.

22. SM228 and D2800 ~ D28079

When SM228 is ON, it is to disable the data exchange among the CPU and its connected modules.

When SM228 is OFF, data exchange among the CPU and its connected modules is enabled and data is stored in D2800 ~ D28079. If the PLC is connected with a special extension module, the PLC uses registers in D28000~D28079 and the registers in this area correspond to CRs to update data. If you need to use this area, you need to pay attention not to use the same area repetively.

See the example below to learn how this works.

Order number	1	2	3	4	5	6
Model	DVP04AD-E2	DVP06XA-E2	DVP04DA-E2	DVP04TC-E2	DVP04PT-E2	DVP06PT-E2
Reading data	Reading values from AD channels 1~4 (D28000 ~ D28003)	Reading values from AD channels 1~4 (D28010 ~ D28013)	N/A	Reading values from TC channels 1~4 (D28030 ~ D28033)	Reading values from PT channels 1~4 (D28040 ~ D28043)	Reading values from PT channels 1~6 (D28050 ~ D28055)
Writing data	N/A	Writing values into DA channels 1~2 (D28014 ~ D28015)	Writing values into DA channels 1~2 (D28020 ~ D28023)	N/A	N/A	N/A

23. Descriptions of the values in SR1560~SR1568

SR	Description
SR1560	Number of the right-side modules connected to the CPU module
SR1561~SR1568	Model code of the 1st – 8th right-side module connected to the CPU module

Number of modules and the device codes

Model	Device code
DVP04AD-E2	16#0080
DVP04DA-E2	16#0081
DVP02DA-E2	16#0041
DVP06XA-E2	16#00C4
DVP04PT-E2	16#0082
DVP04TC-E2	16#0083
DVP06PT-E2	16#00C2

2.2.17 Index Register (E)

The Index register is a 16-bit data register. It is similar to the General register in that data can be read from it and written to it; however, it is mainly used as the index register. The range of index registers is E0–E9. Refer to Section 4.4 in the DVP-ES3 Series Programming Manual for more information about using index registers.

2.2.18 File Registers (FR)

- The DVP-ES3 Series PLC provides you with File registers for storing larger numbers of parameters.
- You can edit, upload, and download the parameters in the File registers through ISPSoft.
- You can read the values in File registers while operating the PLC. Refer to the MEMW instruction (API 2303) in the DVP-ES3 Series Programming Manual for more information about how to write to the File registers.

3

Chapter 3 Instruction Tables

Table of Contents

3.1 Types of Instructions

Instructions used in the DVP-ES3 Series PLC include basic instructions and applied instructions.

Classification	Description
Contact instructions	Instructions such as loading the contact, connecting the contact in series, and connecting the contact in parallel
Output instructions	Bit device output; pulse output
Master control Instructions	Setting and resetting the master control
Rising-edge/Falling-edge detection contact instructions	Triggering the instructions that load the contact, connect the contacts in series, and connect the contacts in parallel
Rising-edge/Falling-edge differential output instructions	Bit device differential output
Other instructions	Other instructions

3.1.2 Applied Instructions

API	Classification	Description
0000-0083	Comparison instructions	Comparisons such as =, <>, >, >=, <, <=
0100-0118	Arithmetic instructions	Using binary numbers or binary-coded decimal numbers to add, subtract, multiply, or divide
0200-0217	Data conversion instructions	Converting a binary-coded decimal number into a binary number, and converting a binary number into a binary-coded decimal number
0300-0310	Data transfer instructions	Transfer the specified data
0400-0402	Jump instructions	Control jumps to a different part of the program
0500-0504	Program execution instructions	Enabling or disabling the interrupt
0600-0602	I/O refreshing instructions	Refreshing the I/O
0700-0711	Miscellaneous instructions	Instructions such as those that apply to the counters, the teach mode timers, and the special timers
0800-0817	Logic instructions	Logical operations such as logical addition and logical multiplication
0900-0904	Rotation instructions	Rotating/Shifting the specified data
1000-1011	Basic instructions	Timer instructions and counter instructions
1100-1115	Shift instructions	Shifting the specified data
1200-1226	Data processing instructions	16-bit data processing such as decoding and encoding
1300-1302	Structure creation instructions	Nested loops
1400-1410	Module instructions	Reading the data from a specific module and writing the data into a specific module
1500-1517	Floating-point number instructions	Floating-point number operations
1600-1608	Real-time clock instructions	Reading and writing, adding, subtracting and comparing the time
1700-1704	Peripheral instructions	I/O points connected to the peripheral
1806-1820	Communication instructions	Controlling the peripheral though communication
1900-1906	Other instructions	Watchdog timer, program delay timer, pulse width, and index registers

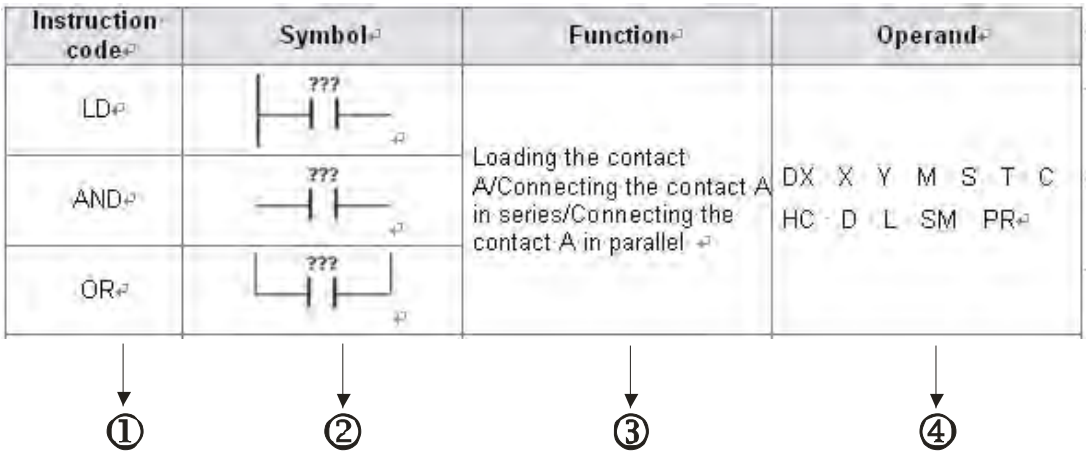
API	Classification	Description
2100-2119	String processing instructions	Conversion between binary or binary-coded decimal numbers and ASCII codes; conversion between binary numbers and strings; conversion between floating-point numbers and strings; string processing
2200-2212	Ethernet instructions	Controlling the Ethernet data exchange
2300-2304	Memory card / File register instructions	Reading the data from the memory card / file register and writing the data to the memory card / file register
2400-2401	Task control instructions	Controlling tasks in the program
2500-2502	Sequential function charts (SFC) instructions	Controlling the SFC instructions
2700-2724	High-speed output instructions	High-speed output and position control instructions
2800-2812	Delta special CANopen communication instructions	CANopen communication instructions especially for Delta devices

3.2 Understanding Instruction Tables

This section describes the table format that this chapter and the rest of this manual uses to describe each instruction. The format is different depending on the type of instruction: Basic or Applied.

3.2.1 Basic Instructions

This section describes the table format that this chapter uses to describe basic instructions in Section 3.3.



Description:

①: The instruction name

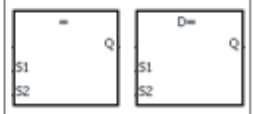
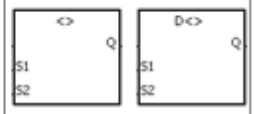
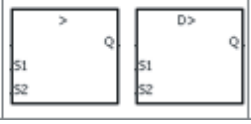
②: The symbol for the instruction in the ladder diagram in ISPSoft

③: The function of the instruction

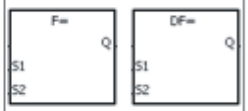
④: The operands supported by the instruction

3.2.2 Applied Instructions (Sorted numerically)

This section describes the table format that this chapter uses to describe applied instructions (sorted by API number) in Section 3.4.1.

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0000	LD=	DLD=	–		Comparing the contact types ON: S1 = S2 OFF: S1≠S2
0001	LD<>	DLD<>	–		Comparing the contact types ON: S1≠S2 OFF: S1 = S2
0002	LD>	DLD>	–		Comparing the contact types ON: S1 > S2 OFF: S1≦S2

① ② ③ ④ ⑤ ⑥

API	Instruction code		Pulse Instruction	Symbol	Function
	32-bit	64-bit			
0018	FLD=	DFLD=	–		Comparing the floating-point number contact types ON: S1 = S2 OFF: S1≠S2

⑦

Description:

①: The applied instruction number

②: The instruction name

③: If a 16-bit instruction can be used as a 32-bit instruction, add a D in front of the 16-bit instruction to form the 32-bit instruction.

④: ✓ indicates that you can use the instruction as a pulse instruction, whereas ─ indicates that it cannot be used as a pulse instruction. For pulse instructions, add a P in back of the instruction.

⑤: The symbol for the instruction in the ladder diagram in ISPSoft

⑥: The function of the instruction

⑦: For single-precision floating-point instructions (32-bit), an F appears in the instruction.

3.2.3 Applied Instructions (Sorted Alphabetically)

This section describes the table format that this chapter uses to describe applied instructions (sorted alphabetically) in Section 3.4.2.

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
E	0501	EI	–	–	Enabling the interrupt
	2208	EIPRW	–	–	Reading and writing via Ethernet/IP connection
	0503	EIX	–	–	Disabling the specific interrupt
	1203	ENCO	–	✓	Encoder
	1905	EPOP	–	✓	Reading the data into the index registers
	1904	EPUSH	–	✓	Storing the contents of the index registers
	①	②	③	④	⑤ ⑥

Description:

①: The initial of the instruction name

②: The applied instruction number

③ ~ ④ : The instruction names

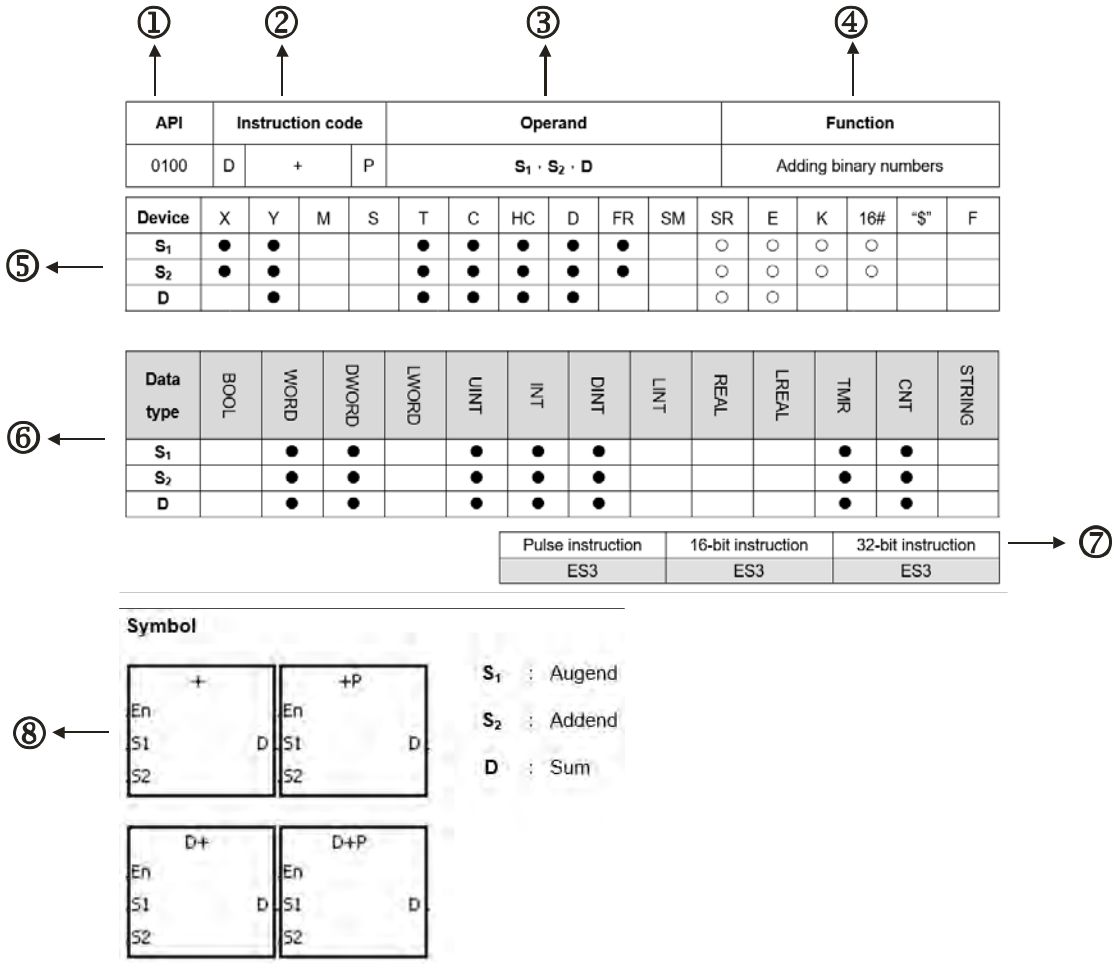
If the 16-bit instruction can be used as the 32-bit instruction, add a D in front of the 16-bit instruction to form the 32-bit instruction.

⑤ : ✓ indicates that you can use the instruction as a pulse instruction, whereas ─ indicates that it cannot be used as a pulse instruction. For the pulse instruction, add a P at the end of the instruction.

⑥: The function of the instruction

3.2.4 Device Tables

This section describes the table format used in the rest of this manual to describe each instruction.



Description:

①: The applied instruction number

②: The instruction name

If the 16-bit instruction can be used as the 32-bit instruction, add a D in front of the 16-bit instruction to form the 32-bit instruction.

③: The operand

④: The function of the instructions

⑤: The devices that are supported by the operand

1. The decimal forms are indicated by “K”, but you enter them directly in ISPSoft. For example, enter the decimal number 30 in ISPSoft.
2. The hexadecimal forms are indicated by 16#. For example, the decimal number 30 is represented by 16#1E in the hexadecimal system.
3. The floating-point numbers are indicated by “F” or ”DF”, but they are represented by decimal points in ISPSoft. For example, the floating-point number F500 is represented by 500.0 in ISPSoft.
4. The strings are indicated by “$”, but they are represented by quotes (“ ”) in ISPSoft. For example, the string 1234

is represented by “1234” in ISPSoft.

5. ○: The hollow circle

   Indicates that the device cannot be modified by an index register.

6. ●: The solid circle

   Indicates that the device cannot be modified by an index register.

⑥：The unit of the operand

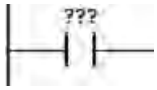
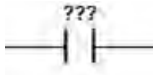
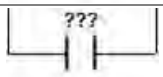
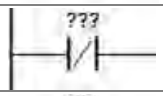
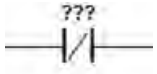
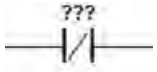
⑦：The format of the instruction

Indicates whether the instruction can be used as a pulse instruction, a 16-bit instruction, or a 32-bit instruction.

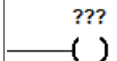
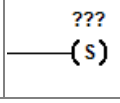
⑧：The symbol for the instruction in the ladder diagram in ISPSoft.

3.3 Lists of Basic Instructions

- Contact instructions

Instruction code	Symbol	Function	Operand
LD		Loading contact A/Connecting contact A in series/Connecting contact A in parallel	DX, X, Y, M, SM, S, T, C, HC, D
AND			
OR			
LDI		Loading contact B/Connecting contact B in series/Connecting contact B in parallel	DX, X, Y, M, SM, S, T, C, HC, D
ANI			
ORI			

- Output instructions

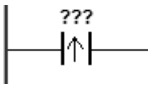
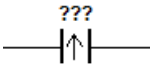
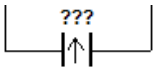
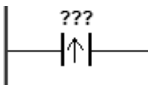
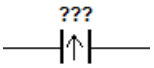
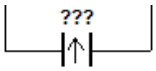
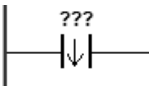
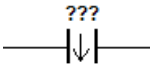
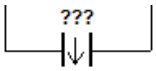
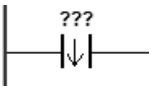
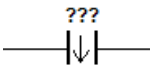
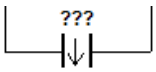
Instruction code	Symbol	Function	Execution condition	Operand
OUT		Driving the coil		DY, Y, M, SM, S, T, C, HC, D
SET		Keeping the device on		DY, Y, M, SM, S, T, C, HC, D

- Master control instructions

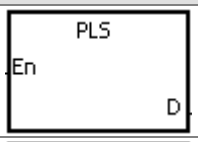
Instruction code	Symbol	Function	Operand

Instruction code	Symbol	Function	Operand
MC		Setting the master control	N
MCR		Resetting the master control	N

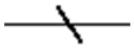
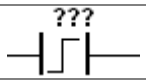
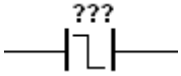
- Rising-edge/Falling-edge detection contact instructions

Instruction code	Symbol	Function	Operand	Instruction code
LDP		Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel		DX, X, Y, M, SM, S, T, C, HC, D
ANDP				
ORP				
PED				X, Y,M, SM, S, T, C, HC, D
APED				
OPED				
LDF		Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel		DX, X, Y, M, SM, S, T, C, HC, D
ANDF				
ORF				
NED				X, Y,M, SM, S, T, C, HC, D
ANED				
ONED				

- Rising-edge/Falling-edge differential output instructions

Instruction code	Symbol	Function	Execution condition	Operand
PLS		Rising-edge differential output		Y, M, SM, S
PLF		Falling-edge differential output		Y, M, SM, S

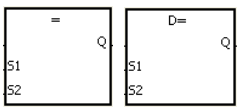
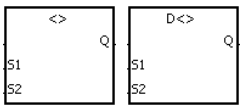
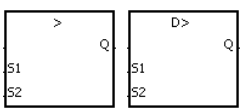
- Other instructions

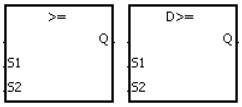
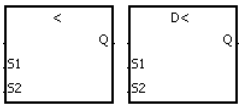
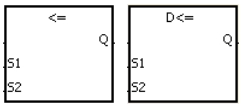
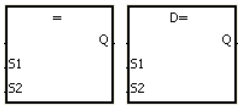
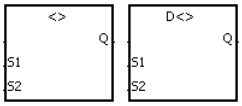
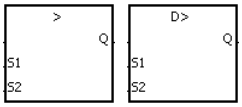
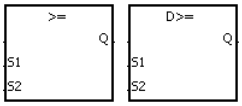
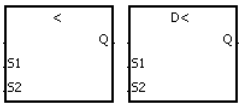
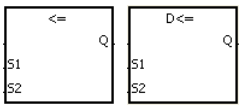
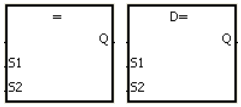
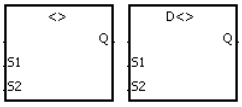
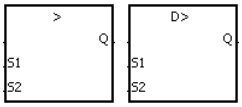
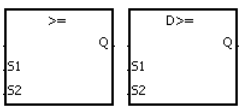
Instruction code	Symbol	Function	Operand
INV		Inverting the logical operation result	–
NP		The circuit is rising edge-triggered.	–
PN		The circuit is falling edge-triggered.	–
FB_NP		The circuit is rising edge-triggered.	Y, M, S, D
FB_PN		The circuit is falling edge-triggered.	Y, M, S, D

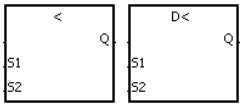
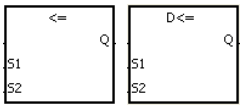
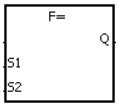
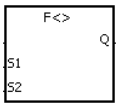
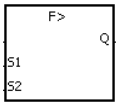
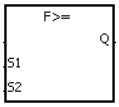
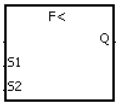
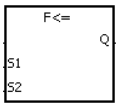
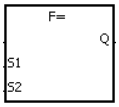
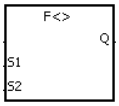
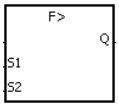
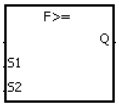
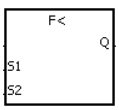
3.4 Lists of Applied Instructions

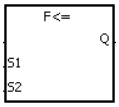
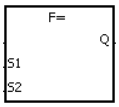
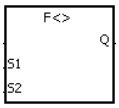
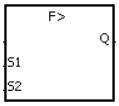
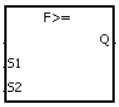
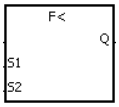
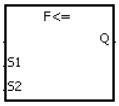
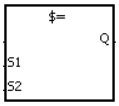
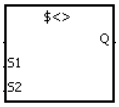
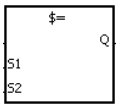
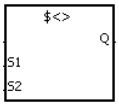
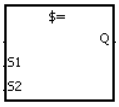
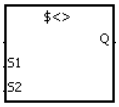
3.4.1 Applied Instructions (Sorted Numerically by API number)

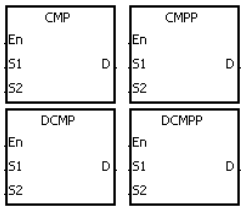
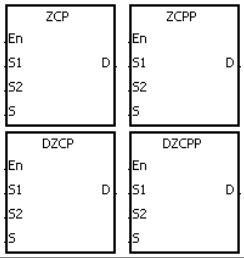
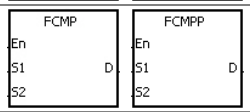
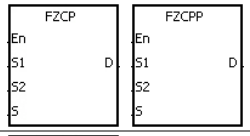
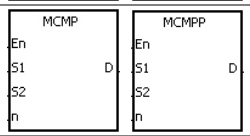
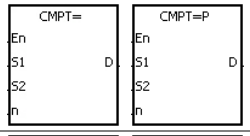
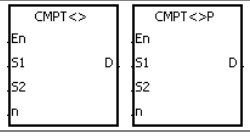
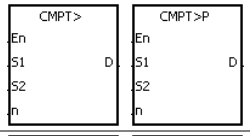
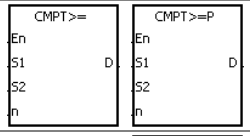
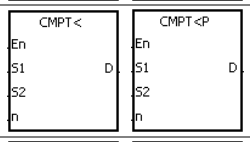
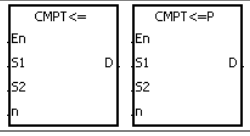
- Comparison instructions

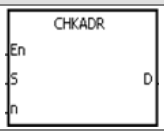
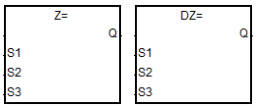
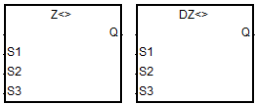
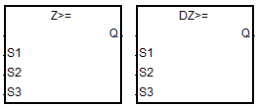
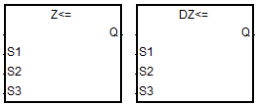
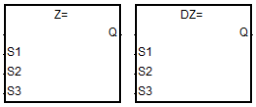
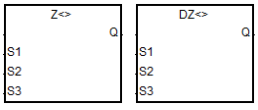
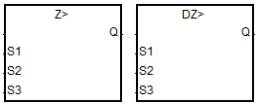
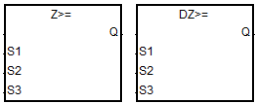
API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0000	LD=	DLD=	–		Comparing values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0001	LD<>	DLD<>	–		Comparing values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0002	LD>	DLD>	–		Comparing values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$

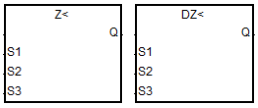
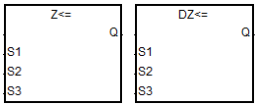
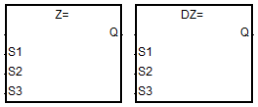
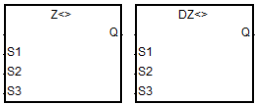
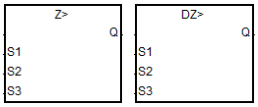
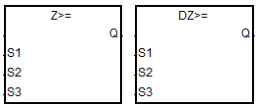
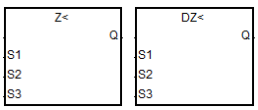
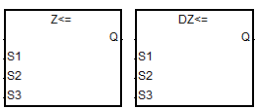
API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0003	LD>=	DLD>=	—		Comparing values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0004	LD<	DLD<	—		Comparing values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0005	LD<=	DLD<=	—		Comparing values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0006	AND=	DAND=	—		Comparing values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0007	AND<>	DAND<>	—		Comparing values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0008	AND>	DAND>	—		Comparing values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0009	AND>=	DAND>=	—		Comparing values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0010	AND<	DAND<	—		Comparing values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0011	AND<=	DAND<=	—		Comparing values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0012	OR=	DOR=	—		Comparing values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0013	OR<>	DOR<>	—		Comparing values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0014	OR>	DOR>	—		Comparing values ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0015	OR>=	DOR>=	—		Comparing values ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0016	OR<	DOR<	—		Comparing values ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0017	OR<=	DOR<=	—		Comparing values ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0018	—	FLD=	—		Comparing floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0019	—	FLD<>	—		Comparing floating-point numbers ON: $S_1 1 \neq S_2$ OFF: $S_1 = S_2$
0020	—	FLD>	—		Comparing floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0021	—	FLD>=	—		Comparing floating-point numbers ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0022	—	FLD<	—		Comparing floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$
0023	—	FLD<=	—		Comparing floating-point numbers ON: $S_1 \leqq S_2$ OFF: $S_1 > S_2$
0024	—	FAND=	—		Comparing floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
0025	—	FAND<>	—		Comparing floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
0026	—	FAND>	—		Comparing floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leqq S_2$
0027	—	FAND>=	—		Comparing floating-point numbers ON: $S_1 \geqq S_2$ OFF: $S_1 < S_2$
0028	—	FAND<	—		Comparing floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geqq S_2$

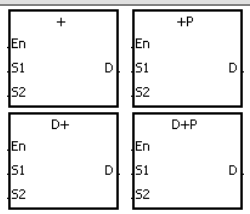
API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0029	—	FAND<=	—		Comparing floating-point numbers ON: **$S_1 \leqq S_2$** OFF: **$S_1 > S_2$**
0030	—	FOR=	—		Comparing floating-point numbers ON: **$S_1 = S_2$** OFF: **$S_1 \neq S_2$**
0031	—	FOR<>	—		Comparing floating-point numbers ON: **$S_1 \neq S_2$** OFF: **$S_1 = S_2$**
0032	—	FOR>	—		Comparing floating-point numbers ON: **$S_1 > S_2$** OFF: **$S_1 \leqq S_2$**
0033	—	FOR>=	—		Comparing floating-point numbers ON: **$S_1 \geqq S_2$** OFF: **$S_1 < S_2$**
0034	—	FOR<	—		Comparing floating-point numbers ON: **$S_1 < S_2$** OFF: **$S_1 \geqq S_2$**
0035	—	FOR<=	—		Comparing floating-point numbers ON: **$S_1 \leqq S_2$** OFF: **$S_1 > S_2$**
0036	LD$=	—	—		Comparing strings ON: **$S_1 = S_2$** ON: **$S_1 \neq S_2$**
0037	LD$<>	—	—		Comparing strings ON: **$S_1 \neq S_2$** OFF: **$S_1 = S_2$**
0042	AND$=	—	—		Comparing strings ON: **$S_1 = S_2$** OFF **$S_1 \neq S_2$**
0043	AND$<>	—	—		Comparing strings ON: **$S_1 \neq S_2$** OFF: **$S_1 = S_2$**
0048	OR$=	—	—		Comparing strings ON: **$S_1 = S_2$** OFF: **$S_1 \neq S_2$**
0049	OR$<>	—	—		Comparing strings ON: **$S_1 \neq S_2$** OFF: **$S_1 = S_2$**

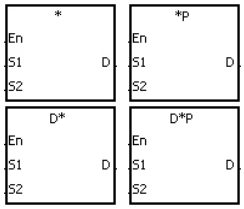
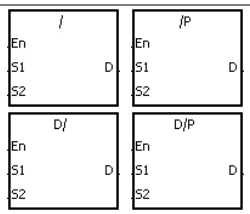
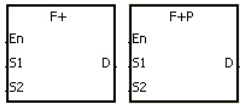
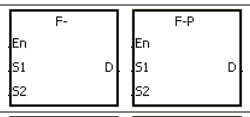
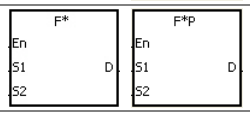
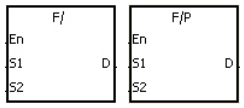
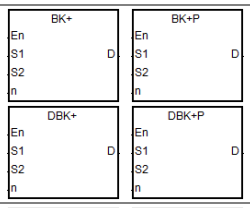
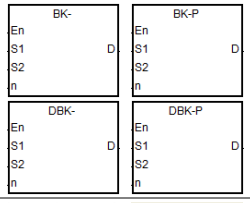
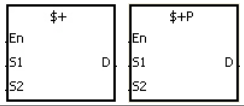
API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0054	CMP	DCMP	✓		Comparing values
0055	ZCP	DZCP	✓		Zone comparison
0056	—	FCMP	✓		Comparing floating-point numbers
0057	—	FZCP	✓		Floating-point zone comparison
0058	MCMP	—	✓		Matrix comparison
0059	CMPT=	—	✓		Comparing tables ON: =
0060	CMPT<>	—	✓		Comparing tables ON: ≠
0061	CMPT>	—	✓		Comparing tables ON: >
0062	CMPT>=	—	✓		Comparing tables ON: ≧
0063	CMPT<	—	✓		Comparing tables ON: <
0064	CMPT<=	—	✓		Comparing tables ON: ≦

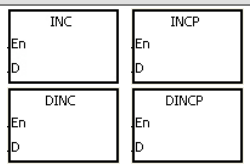
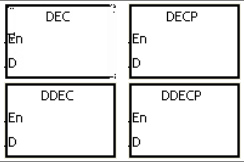
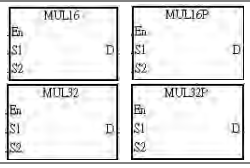
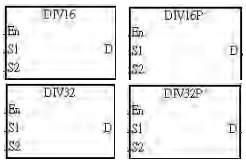
API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0065	CHKADR	—	—		Checking the address of the contact type of a pointer register
0066	LDZ=	DLDZ=	—		Comparing the absolute result of the contact type ON: | S_1- S_2| = | S_3| OFF: | S_1- S_2| ≠ | S_3|
0067	LDZ<>	DLDZ<>	—		Comparing the absolute result of the contact type ON: | S_1- S_2|≠| S_3| OFF: | S_1- S_2| = | S_3|
0068	LDZ>	DLDZ>	—		Comparing the absolute result of the contact type ON: | S_1- S_2| > | S_3| OFF: | S_1- S_2| ≦ | S_3|
0069	LDZ>=	DLDZ>=	—		Comparing the absolute result of the contact type ON: | S_1- S_2|≧| S_3| OFF: | S_1- S_2| < | S_3|
0070	LDZ<	DLDZ<	—		Comparing the absolute result of the contact type ON: | S_1- S_2| < | S_3| OFF: | S_1- S_2|≧ | S_3|
0071	LDZ<=	DLDZ<=	—		Comparing the absolute result of the contact type ON:| S_1- S_2|≦| S_3| OFF: | S_1- S_2| > | S_3|
0072	ANDZ=	DANDZ=	—		Comparing the absolute result of the contact type ON: | S_1- S_2| = | S_3| OFF: | S_1- S_2| ≠ | S_3|
0073	ANDZ<>	DANDZ<>	—		Comparing the absolute result of the contact type ON: | S_1- S_2|≠| S_3| OFF: | S_1- S_2| = | S_3|
0074	ANDZ>	DANDZ>	—		Comparing the absolute result of the contact type ON: | S_1- S_2| > | S_3| OFF: | S_1- S_2| ≦ | S_3|
0075	ANDZ>=	DANDZ>=	—		Comparing the absolute result of the contact type ON: | S_1- S_2|≧| S_3| OFF: | S_1- S_2| < | S_3|

API	Instruction code		Pulse Instruction	Symbol	Function
	16-bit	32-bit			
0076	ANDZ<	DANDZ<	—		Comparing the absolute result of the contact type ON: | S_1- S_2| < | S_3| OFF: | S_1- S_2|≧ | S_3|
0077	ANDZ<=	DANDZ<=	—		Comparing the absolute result of the contact type ON: | S_1- S_2|≦| S_3| OFF: | S_1- S_2| > | S_3|
0078	ORZ=	DORZ=	—		Comparing the absolute result of the contact type ON: | S_1- S_2| = | S_3| OFF: | S_1- S_2| ≠ | S_3|
0079	ORZ<>	DORZ<>	—		Comparing the absolute result of the contact type ON: | S_1- S_2| ≠ | S_3| OFF: | S_1- S_2| = | S_3|
0080	ORZ>	DORZ>	—		Comparing the absolute result of the contact type ON: | S_1- S_2| > | S_3| OFF: | S_1- S_2| ≦ | S_3|
0081	ORZ>=	DORZ>=	—		Comparing the absolute result of the contact type ON: | S_1- S_2| ≧ | S_3| OFF: | S_1- S_2| < | S_3|
0082	ORZ<	DORZ<	—		Comparing the absolute result of the contact type ON: | S_1- S_2| < | S_3| OFF: | S_1- S_2| ≧ | S_3|
0083	ORZ<=	DORZ<=	—		Comparing the absolute result of the contact type ON: | S_1- S_2| ≦ | S_3| OFF: | S_1- S_2| > | S_3|

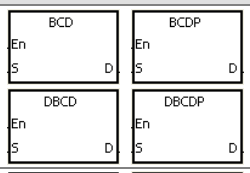
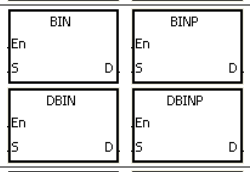
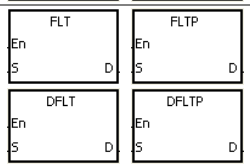
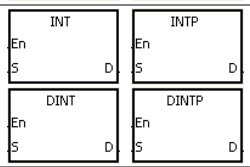
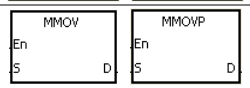
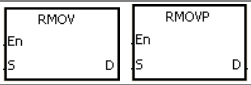
- Arithmetic instructions

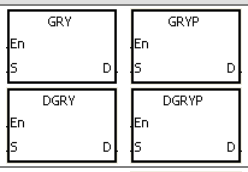
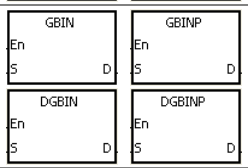
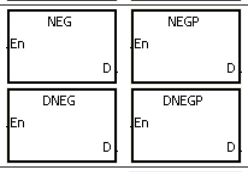
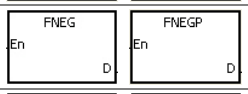
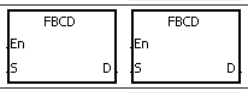
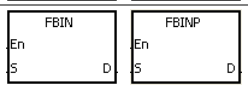
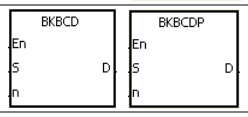
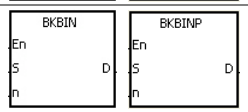
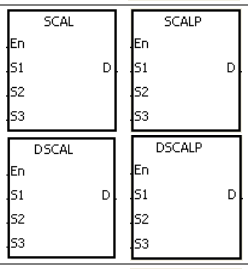
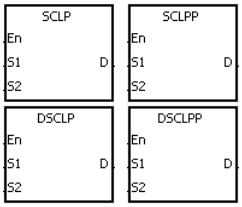
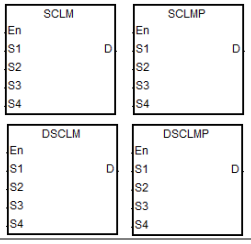
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0100	+	D+	✓		Adding binary numbers **S_1+S_2**=D

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0101	-	D-	✓		Subtracting binary numbers S_1-S_2=D
0102	*	D*	✓		Multiplying binary numbers S_1*S_2=D
0103	/	D/	✓		Dividing binary numbers S_1/S_2=D
0104	—	F+	✓		Adding floating-point numbers S_1+S_2=D
0105	—	F-	✓		Subtracting floating-point numbers S_1-S_2=D
0106	—	F*	✓		Multiplying floating-point numbers S_1*S_2=D
0107	—	F/	✓		Dividing floating-point numbers S_1/S_2=D
0112	BK+	DBK+	✓		Adding binary numbers in blocks
0113	BK-	DBK-	✓		Subtracting binary numbers in blocks
0114	$+	—	✓		Linking strings

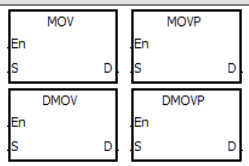
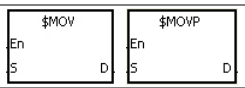
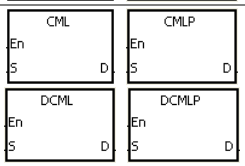
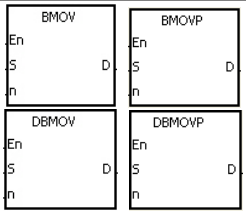
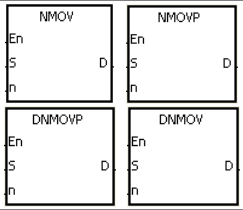
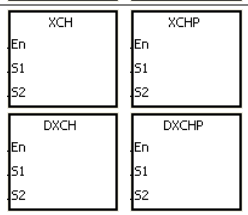
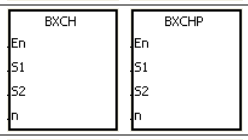
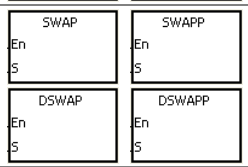
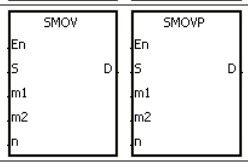
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0115	INC	DINC	✓		Adding one to a binary number
0116	DEC	DDEC	✓		Subtracting one from a binary number
0117	MUL16	MUL32	✓		Multiplying binary numbers for 16-bit multiplying binary numbers for 32-bit
0118	DIV16	DIV32	✓		Dividing binary numbers for 16-bit Dividing binary numbers for 32-bit

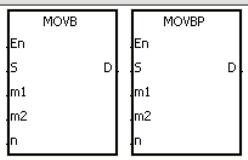
- Data conversion instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0200	BCD	DBCD	✓		Converting a binary number into the binary-coded decimal number
0201	BIN	DBIN	✓		Converting a binary-coded decimal number into a binary number
0202	FLT	DFLT	✓		Converting a binary integer into a binary floating-point number
0204	INT	DINT	✓		Converting a 32-bit floating-point number into a binary integer
0206	MMOV	—	✓		Converting a 16-bit value into a 32-bit value
0207	RMOV	—	✓		Converting a 32-bit value into a 16-bit value

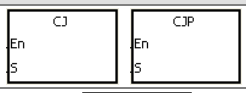
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0208	GRY	DGRY	✓		Converting a binary number into a Gray code
0209	GBIN	DGBIN	✓		Converting a Gray code into a binary number
0210	NEG	DNEG	✓		Two's complement of a number
0211	—	FNEG	✓		Reversing the sign of a 32-bit floating-point number
0212	—	FBCD	✓		Converting a binary floating-point number into a decimal floating-point number
0213	—	FBIN	✓		Converting a decimal floating-point number into a binary floating-point number
0214	BKBCD	—	✓		Converting a binary numbers in blocks into a binary-coded decimal numbers in blocks
0215	BKBIN	—	✓		Converting a binary numbers in blocks into a binary-coded decimal numbers in blocks
0216	SCAL	DSCAL	✓		Finding a scaled value (point-slope)
0217	SCLP	DSCLP	✓		Finding a scaled value (two points)
0222	SCLM	DSCLM	✓		Multi-point area ratio operation

- Data transfer instructions

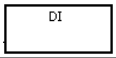
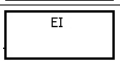
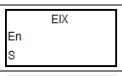
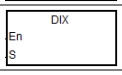
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0300	MOV	DMOV	✓		Transferring data S: Data source D: Data destination
0302	$MOV	—	✓		Transferring a string
0303	CML	DCML	✓		Inverting data
0304	BMOV	DBMOV	✓		Transferring all data
0305	NMOV	DNMOV	✓		Transferring data to several devices
0306	XCH	DXCH	✓		Exchanging the data
0307	BXCH	—	✓		Exchanging all data
0308	SWAP	DSWAP	✓		Exchanging the high byte with the low byte
0309	SMOV	—	✓		Transferring the digits

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0310	MOVB	—	✓		Transferring several bits

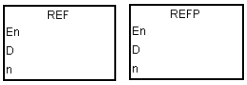
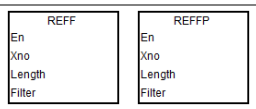
- Jump instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0400	CJ	—	✓		Conditional jump
0401	JMP	—	—		Unconditional jump
0402	GOEND	—	—		Jump to END

- Program execution instructions

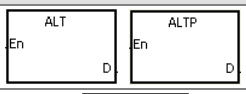
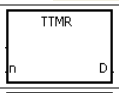
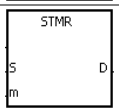
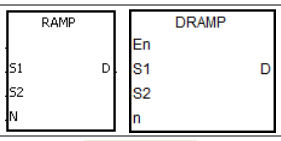
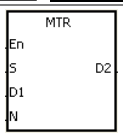
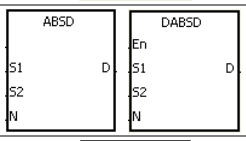
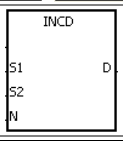
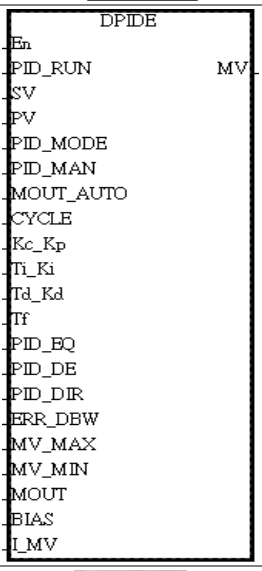
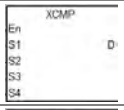
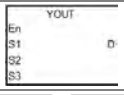
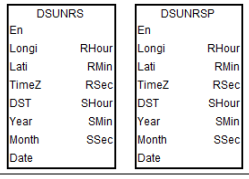
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0500	DI	—	—		Disabling an interrupt
0501	EI	—	—		Enabling an interrupt
0503	EIX	—	—		Disabling a specific interrupt
0504	DIX	—	—		Enabling a specific interrupt

- I/O refreshing instructions

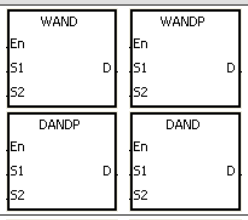
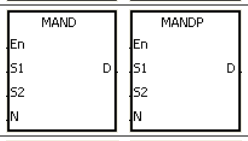
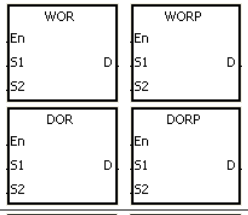
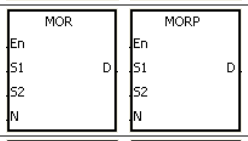
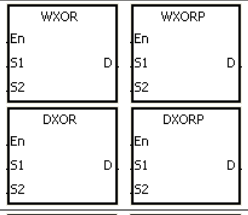
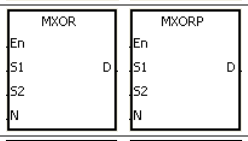
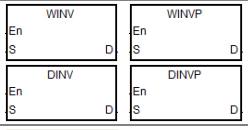
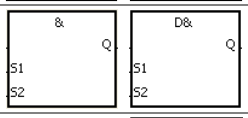
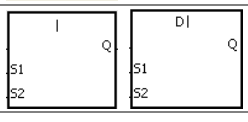
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0600	REF	—	✓		Refreshing the I/O
0601	—	DHSRF	✓		Refreshing the values of high-speed comparison
0602	REFF	—	✓		Refreshing the I/O filtering time

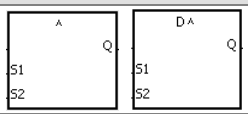
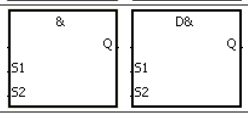
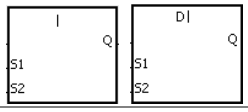
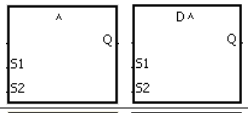
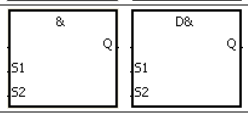
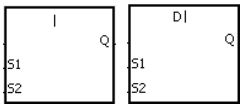
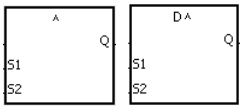
- Miscellaneous instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			

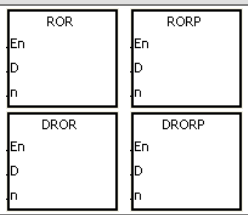
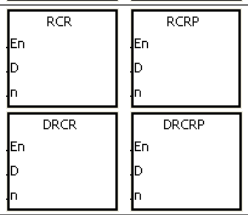
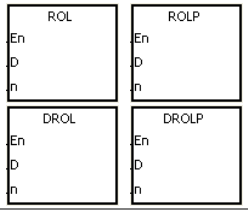
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0700	ALT	—	✓		Alternating between ON and OFF
0701	TTMR	—	—		Teach mode timer
0702	STMR	—	—		Special timer
0703	RAMP	DRAMP	—		Cyclic ramp signal
0704	MTR	—	—		Matrix input
0705	ABSD	DABSD	—		Absolute drum sequencer
0706	INCD	—	—		Incremental drum sequencer
0708	—	DPIDE	—		PID algorithm
0709	XCMP	—	—		Setting up to compare the inputs of multiple work stations
0710	YOUT	—	—		Comparing the outputs of multiple work stations
0711	—	DSUNRS	✓		Setting up the sunrise and sunset times

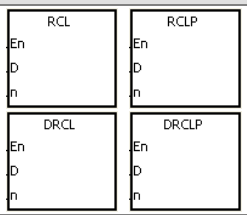
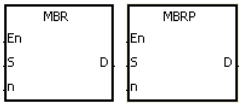
- Logic instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0800	WAND	DAND	✓		Logical AND operation
0801	MAND	—	✓		Matrix AND operation
0802	WOR	DOR	✓		Logical OR operation
0803	MOR	—	✓		Matrix OR operation
0804	WXOR	DXOR	✓		Logical exclusive OR operation
0805	MXOR	—	✓		Matrix exclusive OR operation
0808	WINV	DINV	✓		Logical reversed INV operation
0809	LD&	DLD&	—		S_1&S_2
0810	LD|	DLD|	—		S_1|S_2

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0811	LD^	DLD^	—		S_1^S_2
0812	AND&	DAND&	—		S_1&S_2
0813	AND|	DAND|	—		S_1|S_2
0814	AND^	DAND^	—		S_1^S_2
0815	OR&	DOR&	—		S_1&S_2
0816	OR|	DOR|	—		S_1|S_2
0817	OR^	DOR^	—		S_1^S_2

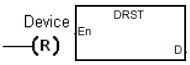
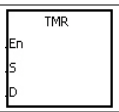
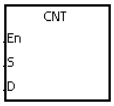
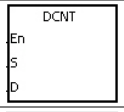
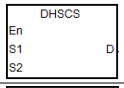
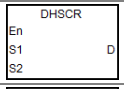
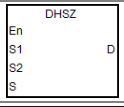
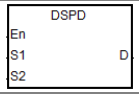
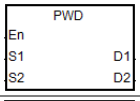
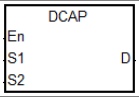
- Rotation instructions

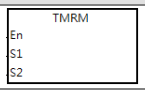
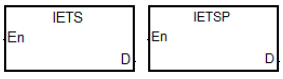
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0900	ROR	DROR	✓		Rotating bits in a group to the right
0901	RCR	DRCR	✓		Rotating bits in a group to the right with the carry flag
0902	ROL	DROL	✓		Rotating bits in a group to the left

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
0903	RCL	DRCL	✓		Rotating bits in a group to the left with the carry flag
0904	MBR	—	✓		Rotating bits to the right or the left in a matrix

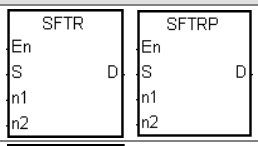
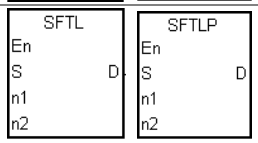
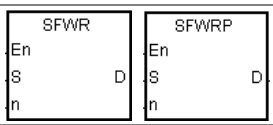
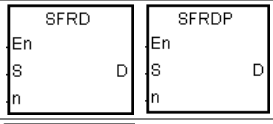
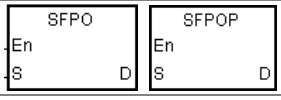
3

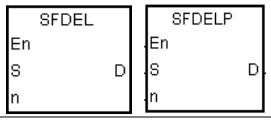
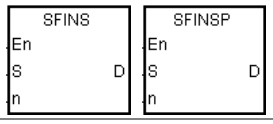
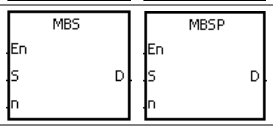
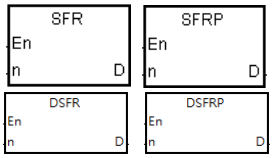
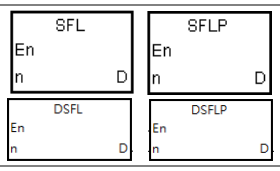
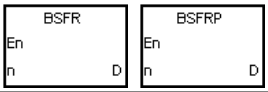
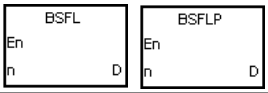
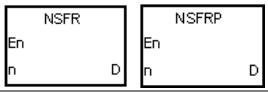
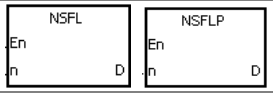
- Timer and counter instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1000	RST	DRST	—		Resetting the contact to OFF or clearing the value in the register
1001	TMR	—	—		16-bit timer (Unit: 100ms)
1002	TMRH	—	—		16-bit timer (Unit: 1ms)
1003	CNT	—	—		16-bit counter
1004	—	DCNT	—		32-bit counter (Including the use of high-speed counters)
1005	—	DHSCS	—		Setting high-speed comparison
1006	—	DHSCR	—		Resetting high-speed comparison
1007	—	DHSZ	—		High-speed input zone comparison
1008	—	DSPD	—		Speed detection
1009	PWD	—	—		Pulse width detection
1010	—	DCAP	—		Capturing the high-speed count value in the external input interrupt

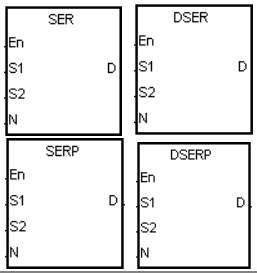
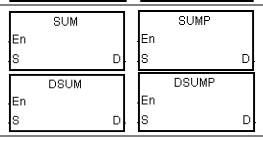
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1011	TMRM	—	—		16-bit timer (Unit: 10ms)
1012	IETS	—	✓		The start of the instruction execution time measurement
1013	IETE	—	✓		The end of the instruction execution time measurement

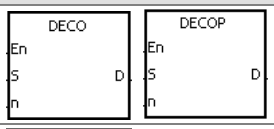
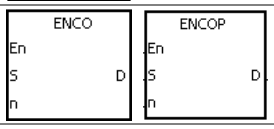
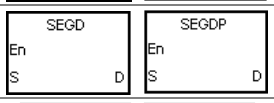
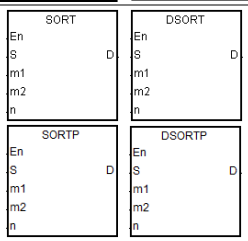
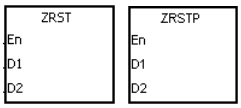
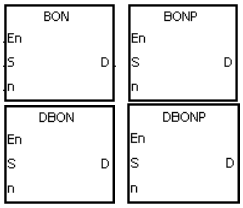
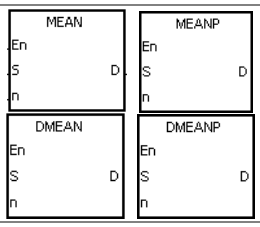
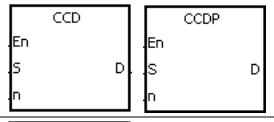
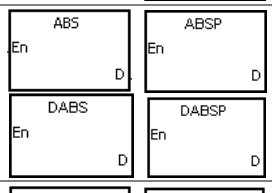
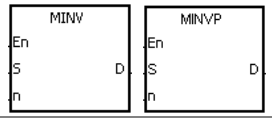
- Shift instructions

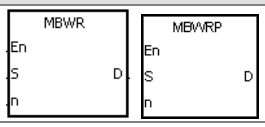
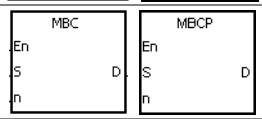
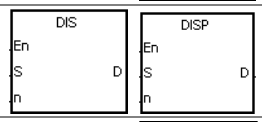
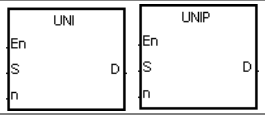
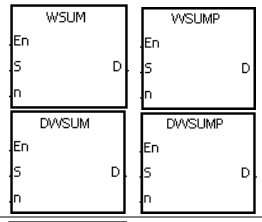
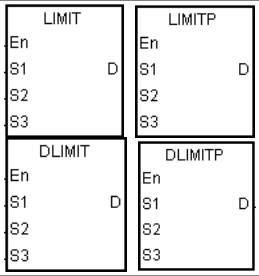
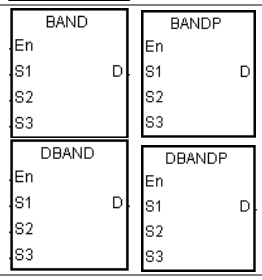
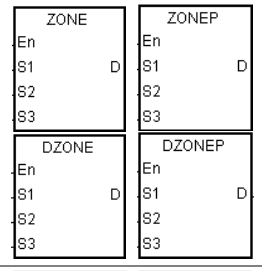
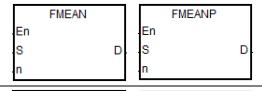
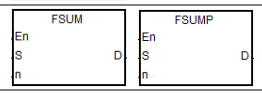
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1100	SFTR	—	✓		Shifting the states of the devices to the right
1101	SFTL	—	✓		Shifting the states of the devices to the left
1102	WSFR	—	✓		Shifting the data in the word devices to the right
1103	WSFL	—	✓		Shifting the data in the word devices to the left
1104	SFWR	—	✓		Shifting the data and writing it to the word device
1105	SFRD	—	✓		Shifting the data and reading it from the word device
1106	SFPO	—	✓		Reading the latest data from the data list

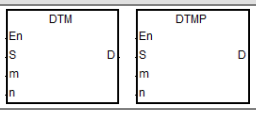
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1107	SFDEL	—	✓		Deleting the data from the data list
1108	SFINS	—	✓		Inserting the data into the data list
1109	MBS	—	✓		Shifting the matrix bits
1110	SFR	DSFR	✓		Shifting the values of the bits in the registers by **n** bits to the right
1111	SFL	DSFL	✓		Shifting the values of the bits in the registers by **n** bits to the left
1112	BSFR	—	✓		Shifting the states of the **n** bit devices by one bit to the right
1113	BSFL	—	✓		Shifting the states of the **n** bit devices by one bit to the left
1114	NSFR	—	✓		Shifting **n** registers to the right
1115	NSFL	—	✓		Shifting **n** registers to the left

- Data processing instructions

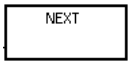
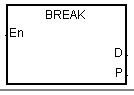
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1200	SER	DSER	✓		Searching the data
1201	SUM	DSUM	✓		Number of bits whose states are ON

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1202	DECO	—	✓		Decoder
1203	ENCO	—	✓		Encoder
1204	SEGD	—	✓		Seven-segment decoding
1205	SORT	DSORT	✓		Sorting the data
1206	ZRST	—	✓		Resetting the zone
1207	BON	DBON	✓		Checking the state of the bit
1208	MEAN	DMEAN	✓		Mean
1209	CCD	—	✓		Sum check
1210	ABS	DABS	✓		Absolute value
1211	MINV	—	✓		Inverting the matrix bits
1212	MBRD	—	✓		Reading the matrix bit

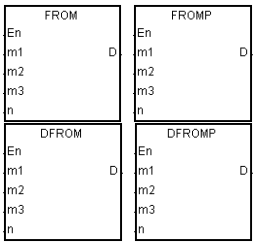
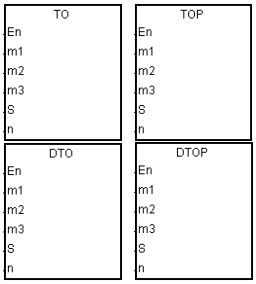
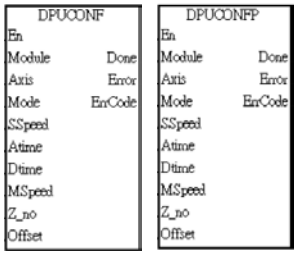
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1213	MBWR	—	✓		Writing the matrix bit
1214	MBC	—	✓		Counting the bits with the value 0 or 1
1215	DIS	—	✓		Disuniting 16-bit data
1216	UNI	—	✓		Uniting 16-bit data
1217	WSUM	DWSUM	✓		Getting the sum
1221	LIMIT	DLIMIT	✓		Confining a value within bounds
1222	BAND	DBAND	✓		Deadband control
1223	ZONE	DZONE	✓		Controlling the zone
1224	—	FMEAN	✓		Mean of the floating point numbers
1225	—	FSUM	✓		Sum of the floating point numbers

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1226	—	DTM	✓		Transfer and move data

- Structure creation instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1300	FOR	—	—		Start of the nested loop
1301	NEXT	—	—		End of the nested loop
1302	BREAK	—	—		Terminating the FOR-NEXT loop

- Module instructions

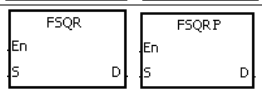
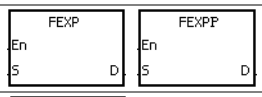
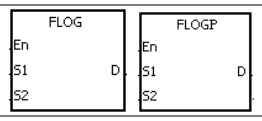
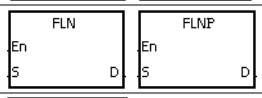
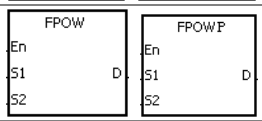
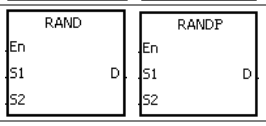
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1400	FROM	DFROM	✓		Reading the data from the control register in the special module
1401	TO	DTO	✓		Writing the data to the control register in the special module
1402	—	DPUCONF	✓		Setting output control parameters of PU module

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1403	PUSTAT	–	–	PUSTAT En Module C_Posi Axis Execute ZeroS Pause Error ErrCode	Reading PU module output state
1404	–	DPUPLS	–	DPUPLS En Module Down Axis Error TarPulse ErrCode TarSpeed	PU module pulse output (no acceleration)
1405	–	DPUDRI	–	DPUDRI En Module Down Axis Error RTarPosi ErrCode TarSpeed	Relative position output of PU module (with acceleration and deceleration)
1406	–	DPUDRA	–	DPUDRA En Module Down Axis Error ATarPosi ErrCode TarSpeed	Absolute addressing output of PU module (with acceleration and deceleration)
1407	–	DPUZRN	–	DPUZRN En Module Done Axis Error Mode ErrCode TarSpeed JogSpeed	PU module homing
1408	–	DPUJOG	–	DPUJOG En Module Busy Axis Error JogSpeed ErrCode	PU module jog output

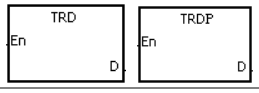
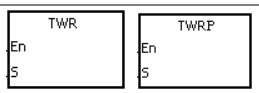
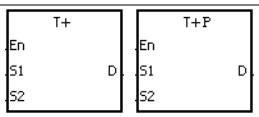
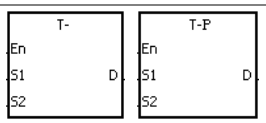
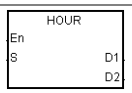
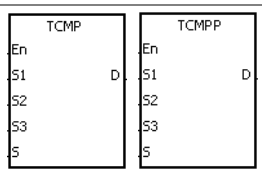
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1409	—	DPUMPG	—	DPUMPG En Module OPulse Axis OSpeed InMode Error InPulse ErrCode InSpeed Rate	PU module MPG output
1410	—	DPUCNT	—	DPUCNT En Module InPulse InMode InSpeed Period Error ZeroS ErrCode	High-speed counter function of PU module

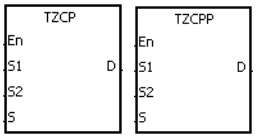
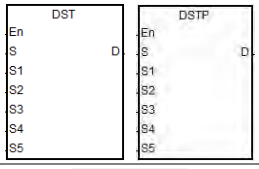
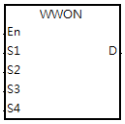
- Floating-point number instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1500	—	FSIN	✓	FSIN En S D / FSINP En S D	Sine of a floating-point number
1501	—	FCOS	✓	FCOS En S D / FCOSP En S D	Cosine of a floating-point number
1502	—	FTAN	✓	FTAN En S D / FTANP En S D	Tangent of a floating-point number
1503	—	FASIN	✓	FASIN En S D / FASINP En S D	Arcsine of a floating-point number
1504	—	FACOS	✓	FACOS En S D / FACOSP En S D	Arccosine of a floating-point number
1505	—	FATAN	✓	FATAN En S D / FATANP En S D	Arctangent of a floating-point number
1506	—	FSINH	✓	FSINH En S D / FSINHP En S D	Hyperbolic sine of a floating-point number
1507	—	FCOSH	✓	FCOSH En S D / FCOSHP En S D	Hyperbolic cosine of a floating-point number
1508	—	FTANH	✓	FTANH En S D / FTANHP En S D	Hyperbolic tangent of a floating-point number
1509	—	FRAD	✓	FRAD En S D / FRADP En S D	Converting from degrees to radians
1510	—	FDEG	✓	FDEG En S D / FDEGP En S D	Converting from radians to degrees

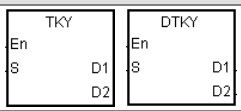
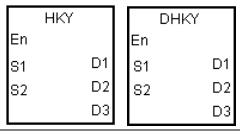
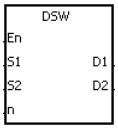
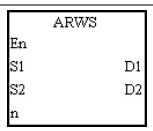
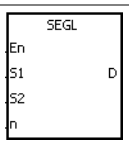
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1511	SQR	DSQR	✓		Square root of a binary number
1512	—	FSQR	✓		Square root of a floating-point number
1513	—	FEXP	✓		Exponent of a floating-point number
1514	—	FLOG	✓		Logarithm of a floating-point number
1515	—	FLN	✓		Natural logarithm of a binary floating-point number
1516	—	FPOW	✓		Raising a floating-point number to a power
1517	RAND	—	✓		Random number

- Real-time clock instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1600	TRD	—	✓		Reading the time
1601	TWR	—	✓		Writing the time
1602	T+	—	✓		Adding the time
1603	T-	—	✓		Subtracting the time
1604	HOUR	—	—		Running-time meter
1605	TCMP	—	✓		Comparing the time

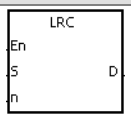
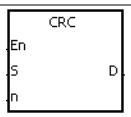
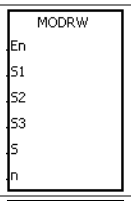
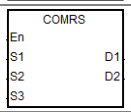
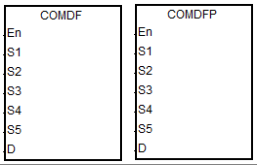
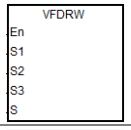
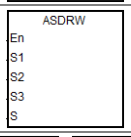
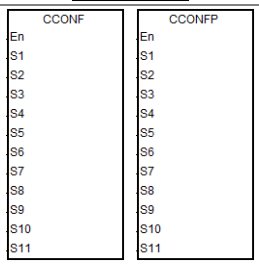
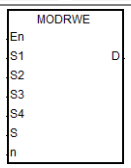
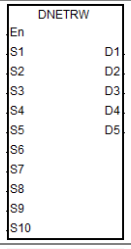
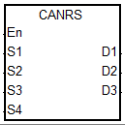
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1606	TZCP	—	✓		Time zone comparison
1607	DST	—	✓		Daylight saving time
1608	WWON	—	—		Weekly working time setup

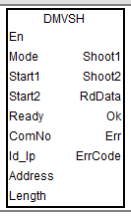
- Peripheral instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1700	TKY	DTKY	—		Ten-key keypad
1701	HKY	DHKY	—		Sixteen-key keypad
1702	DSW	—	—		DIP switch
1703	ARWS	—	—		Arrow keys
1704	SEGL	—	—		Seven-segment display with latches

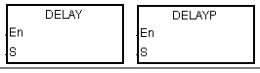
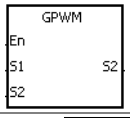
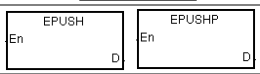
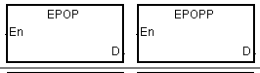
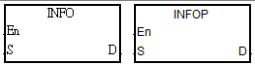
- Communication instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			

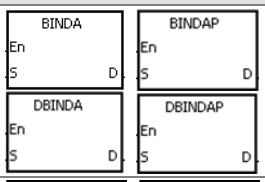
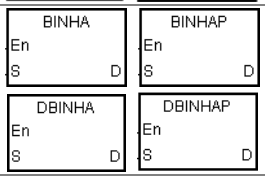
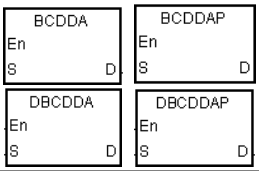
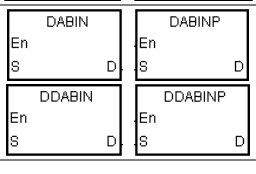
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1806	LRC	—	—		Longitudinal parity check
1807	CRC	—	—		Cyclic redundancy check
1808	MODRW	—	—		Reading and Writing MODBUS data
1812	COMRS	—	—		Sending and receiving communication data
1813	COMDF	—	✓		Setting the communication format for a serial communication port
1814	VFDRW	—	—		Serial communication instruction, exclusively for Delta AC motor drive
1815	ASDRW	—	—		Serial communication instruction, exclusively for Delta servo drive
1816	CCONF	—	✓		Setting the parameters in the data exchange table for a communication port
1817	MODRWE	—	—		Reading and writing Modbus data without using any flags
1818	DNETRW	—	—		Reading and writing DeviceNet communication data
1819	CANRS	—	—		User-defined CAN communication sending and receiving

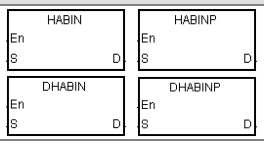
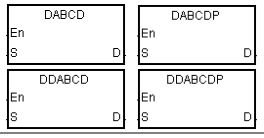
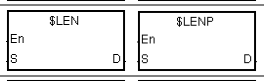
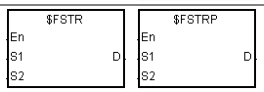
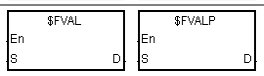
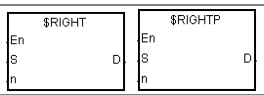
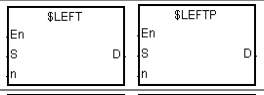
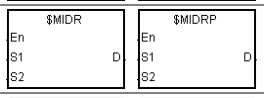
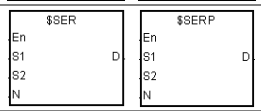
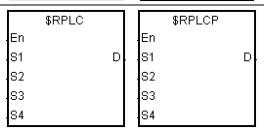
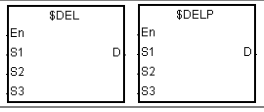
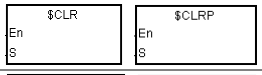
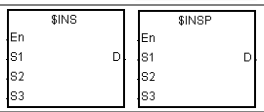
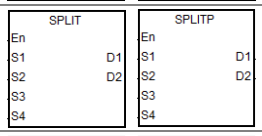
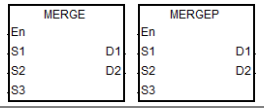
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1820	DMVSH	—	—		Enabling Delta DMV detection and communication

- Other instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
1900	WDT	—	✓		Watchdog timer
1901	DELAY	—	✓		Delaying the execution of a program
1902	GPWM	—	—		General pulse width modulation
1904	EPUSH	—	✓		Storing the contents of the index registers
1905	EPOP	—	✓		Reading data into the index registers
1906	INFO	—	✓		Reading the system data

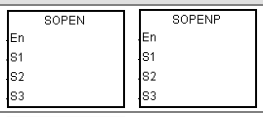
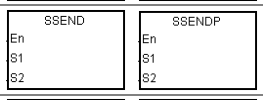
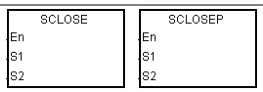
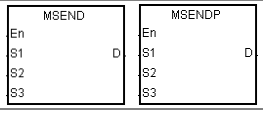
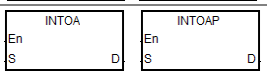
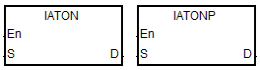
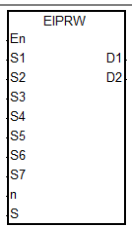
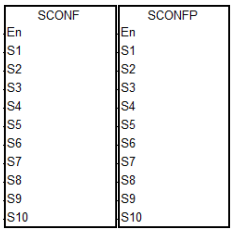
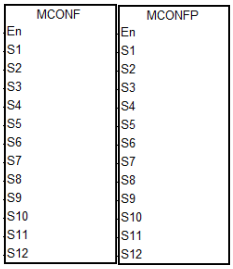
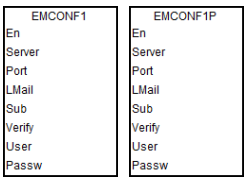
- String processing instructions

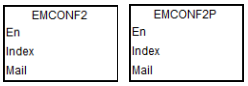
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2100	BINDA	DBINDA	✓		Converting a signed decimal number into ASCII code
2101	BINHA	DBINHA	✓		Converting a binary hexadecimal number into hexadecimal ASCII code
2102	BCDDA	DBCDDA	✓		Converting a binary-coded decimal number into ASCII code
2103	DABIN	DDABIN	✓		Converting a signed decimal ASCII code into a signed decimal binary number

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2104	HABIN	DHABIN	✓		Converting a hexadecimal ASCII code into a hexadecimal binary number
2105	DABCD	DDABCD	✓		Converting an ASCII code into a binary-coded decimal number
2106	$LEN	—	✓		Calculating the length of a string
2109	$FSTR	—	✓		Converting a floating-point number into a string
2110	$FVAL	—	✓		Converting a string into a floating-point number
2111	$RIGHT	—	✓		Retrieving characters from a string begins from the right.
2112	$LEFT	—	✓		Retrieving characters from a string begins from the left.
2113	$MIDR	—	✓		Retrieving a part of a string
2115	$SER	—	✓		Searching a string
2116	$RPLC	—	✓		Replacing the characters in a string
2117	$DEL	—	✓		Deleting the characters in a string
2118	$CLR	—	✓		Clearing a string
2119	$INS	—	✓		Inserting a string
2122	SPLIT	—	✓		Splitting a string
2123	MERGE	—	✓		Merging a string

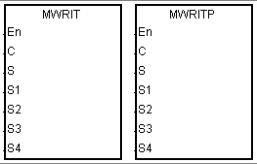
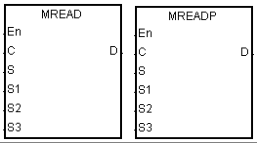
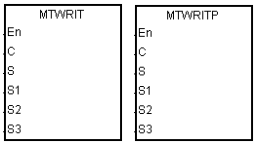
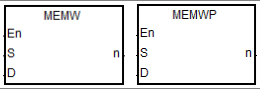
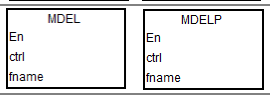
- Ethernet instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			

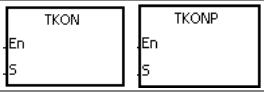
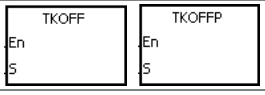
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2200	SOPEN	—	✓		Opening a socket
2201	SSEND	—	✓		Sending data through the socket
2203	SCLOSE	—	✓		Closing a socket
2204	MSEND	—	✓		Sending an email
2206	INTOA	—	✓		Converting an IP address of the integer type into an IP address of the string type
2207	IATON	—	✓		Converting an IP address of the string type into an IP address of the integer type
2208	EIPRW	—	—		Reading and writing EtherNet/IP data
2209	SCONF	—	✓		Setting TCP/UDP socket parameters
2210	MCONF	—	✓		Reading/Writing Modbus TCP data
2211	EMCONF1	—	□		Setting email server parameter values

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2212	EMCONF2	–	□		Setting email address

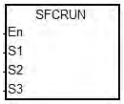
- Memory card / File register instructions

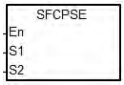
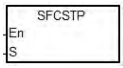
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2300	MWRIT	–	✓		Writing data from the PLC into a memory card
2301	MREAD	–	✓		Reading data from the memory card into the PLC
2302	MTWRIT	–	✓		Writing a string into the memory card
2303	MEMW	–	✓		Writing data into the file register
2304	MDEL	–	✓		Deleting files on the memory card

- Task control instructions

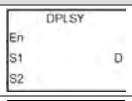
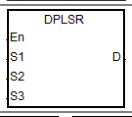
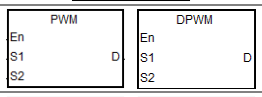
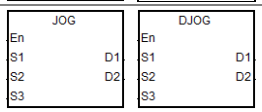
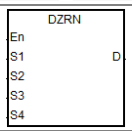
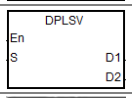
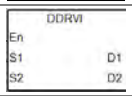
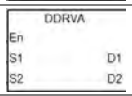
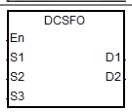
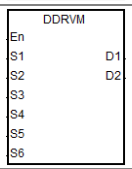
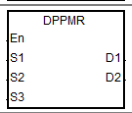
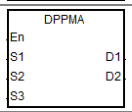
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2400	TKON	–	✓		Enabling a cyclic task
2401	TKOFF	–	✓		Disabling a cyclic task

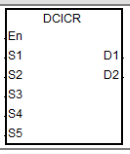
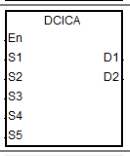
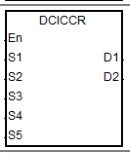
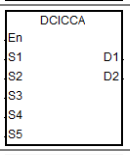
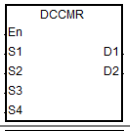
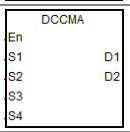
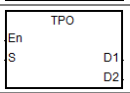
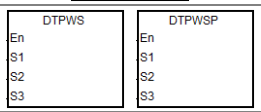
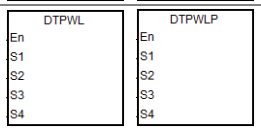
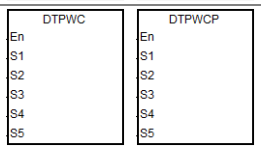
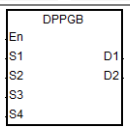
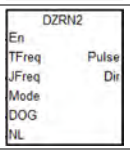
- Sequential function charts (SFC) instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2500	SFCRUN	–	–		Enabling SFC

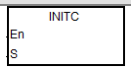
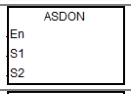
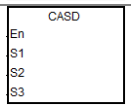
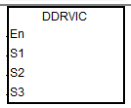
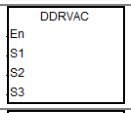
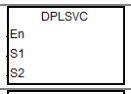
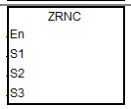
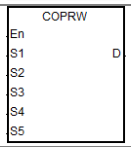
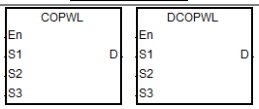
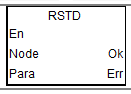
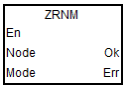
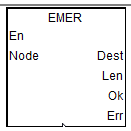
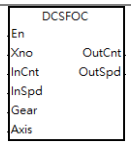
API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2501	SFCPSE	–	–		Pausing SFC
2502	SFCSTP	–	–		Stopping SFC

- High-speed output instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2700	–	DPLSY	–		High-speed pulse output (without ramp-up/down process)
2701	–	DPLSR	–		High-speed pulse output (with ramp-up/down process)
2702	PWM	DPWM	–		Pulse width modulation
2703	JOG	DJOG	–		JOG output
2704	–	DZRN	–		Zero return
2705	–	DPLSV	–		Adjustable pulse output
2706	–	DDRVI	–		Relative position control
2707	–	DDRVA	–		Absolute position control
2708	CSFO	–	–		Catch speed and proportional output
2709	–	DDRVM	–		Mark alignment positioning
2710	–	DPPMR	–		2-Axis relative-coordinate point-to-point synchronized motion
2711	–	DPPMA	–		2-Axis absolute-coordinate point-to-point synchronized motion

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2712	–	DCICR	–		2-Axis relative-position clockwise arc interpolation
2713	–	DCICA	–		2-Axis absolute-position clockwise arc interpolation
2714	–	DCICCR	–		2-Axis relative-position counterclockwise arc interpolation
2715	–	DCICCA	–		2-Axis absolute-position counterclockwise arc interpolation
2716	–	DCCMR	–		Relative-position circle drawing
2717	–	DCCMA	–		Absolute-position circle drawing
2718	TPO	–	–		The position planning table controls the output
2719	–	DTPWS	✓		Setting single-axis output parameters in the position planning table
2720	–	DTPWL	✓		Setting linear interpolation parameters in the position planning table
2721	–	DPTWC	✓		Setting arc interpolation parameters in the position planning table
2723	–	DPPGB	–		Point to point go back and forth
2724	–	DZRN2	–		Zero return 2 (output direction can be set)

- Delta Special CANopen Communication Instructions

API	Instruction code		Pulse instruction	Symbol	Function
	16-bit	32-bit			
2800	INITC	–	–		Initializing the servo for CANopen communication
2801	ASDON	–	–		Servo-ON and Servo-OFF
2802	CASD	–	–		Setting the acceleration time and deceleration time for a servo
2803	–	DDRVIC	–		Servo relative position control
2804	–	DDRVAC	–		Servo absolute position control
2805	–	DPLSVC	–		Servo speed control
2806	ZRNC	–	–		Servo homing
2807	COPRW	–	–		Reading and writing CANopen communication data
2808	COPWL	DCOPWL	–		Writing multiple CANopen parameter values
2809	RSTD	–	–		Sending Reset or NMT command
2810	ZRNM	–	–		Setting the homing mode for Delta servo
2811	EMER	–	–		Reading Emergency message
2812	–	DCSFOC	–		Controlling the tracking function of a servo via communication

3.4.2 Applied Instructions (Sorted Alphabetically)

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
Symbol	0114	$+	–	✓	Linking two strings
	2118	$CLR	–	✓	Clearing a string
	2117	$DEL	–	✓	Deleting the characters in a string
	2109	$FSTR	–	✓	Converting a floating-point number into a string
	2110	$FVAL	–	✓	Converting a string into a floating-point number
	2119	$INS	–	✓	Inserting a string
	2112	$LEFT	–	✓	Retrieving characters in a string begins from the left.
	2106	$LEN	–	✓	Calculating the length of a string
	2113	$MIDR	–	✓	Retrieving a part of a string
	0302	$MOV	–	✓	Transferring a string
	2111	$RIGHT	–	✓	Retrieving characters in a string begins from the right.
	2116	$RPLC	–	✓	Replacing the characters in a string
	2115	$SER	–	✓	Searching a string
	0102	*	D*	✓	Multiplication of binary numbers
	0103	/	D/	✓	Division of binary numbers
	0100	+	D+	✓	Addition of binary numbers
A	1210	ABS	DABS	✓	Absolute value
	0705	ABSD	DABSD	–	Absolute drum sequencer
	0700	ALT	–	✓	Alternating between ON and OFF
	0043	AND$<>	–	–	S1≠S2
	0042	AND$=	–	–	S1 = S2
	0812	AND&	DAND&	–	S1&S2
	0814	AND^	DAND^	–	S1^S2
	0813	AND|	DAND|	–	S1|S2
	0010	AND<	DAND<	–	S1 < S2
	0011	AND<=	DAND<=	–	S1 ≦S2
	0007	AND<>	DAND<>	–	S1≠S2
	0006	AND=	DAND=	–	S1 = S2
	0008	AND>	DAND>	–	S1 > S2
	0009	AND>=	DAND>=	–	S1 ≧S2
	0076	ANDZ<	DANDZ<	–	|S1-S2| < |S3|
	0077	ANDZ<=	DANDZ<=	–	|S1-S2| ≦|S3|
	0073	ANDZ<>	DANDZ<>	–	|S1-S2|≠|S3|
	0072	ANDZ=	DANDZ=	–	|S1-S2|=|S3|

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0074	ANDZ>	DANDZ>	–	|S1-S2| > |S3|
	0075	ANDZ>=	DANDZ>=	–	|S1-S2| ≧|S3|
	1703	ARWS	–	–	Arrow key input
	2801	ASDON	–	–	Servo-ON and servo-OFF
	1815	ASDRW	–	–	Serial communication instruction exclusive for Delta servo drive
B	1222	BAND	DBAND	✓	Deadband control
	0200	BCD	DBCD	✓	Converting a binary number into a binary-coded decimal number
	2102	BCDDA	DBCDDA	✓	Converting a binary-coded decimal number into ASCII code
	0201	BIN	DBIN	✓	Converting a binary-coded decimal number into a binary number
	2100	BINDA	DBINDA	✓	Converting a singed decimal number into ASCII code
	2101	BINHA	DBINHA	✓	Converting a binary hexadecimal number into the hexadecimal ASCII code
	0113	BK-	DBK-	✓	Subtracting binary numbers in blocks
	0112	BK+	DBK+	✓	Adding binary numbers in blocks
	0214	BKBCD	–	✓	Converting binary numbers in blocks into binary-coded decimal numbers in blocks
	0215	BKBIN	–	✓	Converting binary numbers in blocks into binary-coded decimal numbers in blocks
	0304	BMOV	DBMOV	✓	Transferring all data
	1207	BON	DBON	✓	Checking the state of a bit
	1302	BREAK	–	–	Terminating the FOR-NEXT loop
	1113	BSFL	–	✓	Shifting the states of the n bit devices by one bit to the left
	1112	BSFR	–	✓	Shifting the states of the n bit devices by one bit to the right
	0307	BXCH	–	✓	Exchanging all data
C	1819	CANRS	–	–	User-defined CAN communication sending and receiving
	2802	CASD	–	–	Setting the acceleration time and deceleration time for a servo
	1209	CCD	–	✓	Checksum
	1816	CCONF	–	✓	Setting the parameters in the data exchange table of a communication port
	0065	CHKADR	–	–	Checking the address of the pointer register
	0400	CJ	–	✓	Conditional jump
	0303	CML	DCML	✓	Inverting data
	0054	CMP	DCMP	✓	Comparing values
	0063	CMPT<	–	✓	Comparing tables

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
					ON: <
	0064	CMPT<=	–	✓	Comparing tables ON: ≦
	0060	CMPT<>	–	✓	Comparing tables ON: ≠
	0059	CMPT=	–	✓	Comparing tables ON: =
	0061	CMPT>	–	✓	Comparing tables ON: >
	0062	CMPT>=	–	✓	Comparing tables ON: ≧
	1003	CNT	–	–	16-bit counter
	1813	COMDF	–	✓	Setting the communication format for a serial communication port
	1812	COMRS	–	–	Sending and receiving communication data
	2807	COPRW	–	–	Reading and writing CANopen communication data
	2808	COPWL	DCOPWL	–	Writing multiple CANopen parameter values
	1807	CRC	–	–	Cyclic redundancy check
	2708	CSFO	–	–	Catch speed and proportional output
D	0101	–	D-	✓	Subtracting binary numbers **S_1-S_2**=D
	1010	–	DCAP	–	Capturing the high-speed count value in the external input interrupt
	2717	–	DCCMA	–	Absolute-position circle drawing
	2716	–	DCCMR	–	Relative-position circle drawing
	2713	–	DCICA	–	2-Axis absolute-position clockwise arc interpolation
	2715	–	DCICCA	–	2-Axis absolute-position counterclockwise arc interpolation
	2714	–	DCICCR	–	2-Axis relative-position counterclockwise arc interpolation
	2712	–	DCICR	–	2-Axis relative-position clockwise arc interpolation
	1004	–	DCNT	–	32-bit counter
	2812	–	DCSFOC	–	Controlling the tracking function of a servo via communication
	2707	–	DDRVA	–	Absolute position control
	2804	–	DDRVAC	–	Servo absolute position control
	2706	–	DDRVI	–	Relative position control
	2803	–	DDRVIC	–	Servo relative position control
	2709	–	DDRVM	–	Mark alignment positioning
	1006	–	DHSCR	–	Resetting high-speed comparison

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1005	–	DHSCS	–	Setting high-speed comparison
	0601	–	DHSRF	✓	Refreshing the values of high-speed comparison
	1007	–	DHSZ	–	High-speed input zone comparison
	0708	–	DPIDE	–	PID algorithm
	2701	–	DPLSR	–	High-speed pulse output (with ramp-up/down process)
	2705	–	DPLSV	–	Adjustable pulse output
	2805	–	DPLSVC	–	Servo speed control
	2700	–	DPLSY	–	High-speed pulse output (without ramp-up/down process)
	2723	–	DPPGB	–	Point to point go back and forth
	2711	–	DPPMA	–	2-Axis absolute-coordinate point-to-point synchronized motion
	2710	–	DPPMR	–	2-Axis relative-coordinate point-to-point synchronized motion
	2721	–	DPTWC	✓	Setting arc interpolation parameters in the position planning table
	1410	–	DPUCNT	–	High-speed counter function of PU module
	1402	–	DPUCONF	✓	Setting output control parameters of PU module
	1406	–	DPUDRA	–	Absolute addressing output of PU module (with acceleration and deceleration)
	1405	–	DPUDRI	–	Relative position output of PU module (with acceleration and deceleration)
	1408	–	DPUJOG	–	PU module jog output
	1409	–	DPUMPG	–	PU module MPG output
	1404	–	DPUPLS	–	PU module pulse output (no acceleration)
	1407	–	DPUZRN	–	PU module homing
	1008	–	DSPD	–	Detecting speed
	0711	–	DSUNRS	–	Setting up the sunrise and sunset times
	1226	–	DTM	✓	Transfer and move data
	2720	–	DTPWL	✓	Setting linear interpolation parameters in the position planning table
	2719	–	DTPWS	✓	Setting single-axis output parameters in the position planning table
	2704	–	DZRN	–	Zero return
	2724	–	DZRN2	–	Zero return 2 (output direction can be set)
	2105	DABCD	DDABCD	✓	Converting an ASCII code into a binary-coded decimal number
	2103	DABIN	DDABIN	✓	Converting a signed decimal ASCII code into a signed decimal binary number
	0116	DEC	DDEC	✓	Subtracting one from a binary number

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1202	DECO	–	✓	Decoder
	1901	DELAY	–	✓	Delaying the execution of a program
	0500	DI	–	–	Disabling the interrupt
	1215	DIS	–	✓	Disuniting the 16-bit data
	0118	DIV16	DIV32	✓	Division of binary numbers for 16-bit Division of binary numbers for 32-bit
	0504	DIX	–	–	Enabling a specific interrupt
	1820	DMVSH	–	–	Enabling Delta DMV detection and communication
	1818	DNETRW	–	–	Reading and writing DeviceNet communication data
	1607	DST	–	✓	Daylight saving time
	1702	DSW	–	–	DIP switch
	0501	EI	–	–	Enabling an interrupt
	2208	EIPRW	–	–	Reading and writing through an EtherNet/IP connection
	0503	EIX	–	–	Disabling a specific interrupt
	2211	EMCONF1	–	✓	Setting email server parameter values
E	2212	EMCONF2	–	✓	Setting email address
	2811	EMER	–	–	Reading Emergency message
	1203	ENCO	–	✓	Encoder
	1905	EPOP	–	✓	Reading data into the index registers
	1904	EPUSH	–	✓	Storing the contents of the index registers
	0105	–	F-	✓	Subtracting floating-point numbers S_1-S_2=D
	0106	–	F*	✓	Multiplying floating-point numbers S_1*S_2=D
	0107	–	F/	✓	Dividing floating-point numbers S_1/S_2=D
	0104	–	F+	✓	Adding floating-point numbers S1+S2=D
	1504	–	FACOS	✓	Arccosine of the floating-point number
F	0028	–	FAND<	–	S1 < S2
	0029	–	FAND<=	–	S1 ≦S2
	0025	–	FAND<>	–	S1≠S2
	0024	–	FAND=	–	S1 = S2
	0026	–	FAND>	–	S1 > S2
	0027	–	FAND>=	–	S1 ≧S2
	1503	–	FASIN	✓	Arcsine of a floating-point number
	1505	–	FATAN	✓	Arctangent of a floating-point number

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0212	–	FBCD	✓	Converting a binary floating-point number into a decimal floating-point number
	0213	–	FBIN	✓	Converting a decimal floating-point number into a binary floating-point number
	0056	–	FCMP	✓	Comparing floating-point numbers
	1501	–	FCOS	✓	Cosine of a floating-point number
	1507	–	FCOSH	✓	Hyperbolic cosine of a floating-point number
	1510	–	FDEG	✓	Converting radians to degrees
	1513	–	FEXP	✓	The exponent of a floating-point number
	0022	–	FLD<	–	S1＜S2
	0023	–	FLD<=	–	S1 ≦S2
	0019	–	FLD<>	–	S1≠S2
	0018	–	FLD=	–	S1 = S2
	0020	–	FLD>	–	S1 > S2
	0021	–	FLD>=	–	S1 ≧S2
	1515	–	FLN	✓	Natural logarithm of a binary floating-point number
	1514	–	FLOG	✓	Logarithm of a floating-point number
	1224	–	FMEAN	✓	The mean of floating point numbers
	0211	–	FNEG	✓	Reversing the sign of a floating-point number
	0034	–	FOR<	–	S1 < S2
	0035	–	FOR<=	–	S1 ≦S2
	0031	–	FOR<>	–	S1≠S2
	0030	–	FOR=	–	S1 = S2
	0032	–	FOR>	–	S1 > S2
	0033	–	FOR>=	–	S1 ≧S2
	1516	–	FPOW	✓	Raising a floating-point number to a power
	1509	–	FRAD	✓	Converting degrees to radians
	1500	–	FSIN	✓	Sine of a floating-point number
	1506	–	FSINH	✓	Hyperbolic sine of a floating-point number
	1512	–	FSQR	✓	Square root of a floating-point number
	1225	–	FSUM	✓	The sum of floating point numbers
	1502	–	FTAN	✓	Tangent of a floating-point number
	1508	–	FTANH	✓	Hyperbolic tangent of a floating-point number
	0057	–	FZCP	✓	Floating-point zone comparison
	0202	FLT	DFLT	✓	Converting a binary integer into a binary floating-point number
	1300	FOR	–	–	Start of a nested loop

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1400	FROM	DFROM	✓	Reading data from the control register in the extension module
G	0209	GBIN	DGBIN	✓	Converting a Gray code into a binary number
	0402	GOEND	–	–	Jumping to END
	1902	GPWM	–	–	General pulse width modulation
	0208	GRY	DGRY	✓	Converting a binary number into a Gray code
H	2104	HABIN	DHABIN	✓	Converting a hexadecimal ASCII code into a hexadecimal binary number
	1701	HKY	DHKY	–	Hexadecimal key input
	1604	HOUR	–	–	Running-time meter
I	2207	IATON	–	✓	Converting an IP address of the string type into an IP address of the integer type
	1012	IETS	–	✓	The start of the instruction execution time measurement
	1013	IETE	–	✓	The end of the instruction execution time measurement
	0115	INC	DINC	✓	Adding one to a binary number
	0706	INCD	–	–	Incremental drum sequencer
	1906	INFO	–	✓	Reading the system data
	2800	INITC	–	–	Initializing the servo for CANopen communication
	0204	INT	DINT	✓	Converting a 32-bit floating-point number into a binary integer
	2206	INTOA	–	✓	Converting an IP address of the integer type into an IP address of the string type
J	0401	JMP	–	–	Unconditional jump
	2703	JOG	DJOG	–	JOG output
L	0037	LD$<>	–	–	S1≠S2
	0036	LD$=	–	–	S1＝S2
	0809	LD&	DLD&	–	S1&S2
	0811	LD^	DLD^	–	S1^S2
	0810	LD|	DLD|	–	S1|S2
	0004	LD<	DLD<	–	S1＜S2
	0005	LD<=	DLD<=	–	S1≦S2
	0001	LD<>	DLD<>	–	S1≠S2
	0000	LD=	DLD=	–	S1＝S2
	0002	LD>	DLD>	–	S1＞S2
	0003	LD>=	DLD>=	–	S1≧S2
	0070	LDZ<	DLDZ<	–	|S1-S2|＜|S3|
	0071	LDZ<=	DLDZ<=	–	|S1-S2|≦|S3|

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0067	LDZ<>	DLDZ<>	–	|S1-S2|≠|S3|
	0066	LDZ=	DLDZ=	–	|S1-S2|=|S3|
	0068	LDZ>	DLDZ>	–	|S1-S2|＞|S3|
	0069	LDZ>=	DLDZ>=	–	|S1-S2|≧|S3|
	1221	LIMIT	DLIMIT	✓	Confining a value within bounds
	1806	LRC	–	–	Longitudinal parity check
M	0801	MAND	–	✓	Matrix AND operation
	2304	MDEL	–	✓	Deleting files on the memory card
	2123	MERGE	–	✓	Merging a string
	1214	MBC	–	✓	Counting the bits with the value 0 or 1
	0904	MBR	–	✓	Rotating the matrix bits
	1212	MBRD	–	✓	Reading the matrix bit
	1109	MBS	–	✓	Shifting the matrix bits
	1213	MBWR	–	✓	Writing the matrix bit
	0058	MCMP	–	✓	Matrix comparison
	2210	MCONF	–	✓	Reading/Writing Modbus TCP data
	1208	MEAN	DMEAN	✓	Mean
	2303	MEMW	–	✓	Writing data into the file register
	1211	MINV	–	✓	Inverting matrix bits
	0206	MMOV	–	✓	Converting a 16-bit value into a 32-bit value
	1808	MODRW	–	–	Reading/Writing MODBUS data
	1817	MODRWE	–	–	Reading and writing Modbus data without using any flags
	0803	MOR	–	✓	Matrix OR operation
	0300	MOV	DMOV	✓	Transferring data
	0310	MOVB	–	✓	Transferring several bits
	2301	MREAD	–	✓	Reading data from the memory card into the PLC
	2204	MSEND	–	✓	Sending an email
	0704	MTR	–	–	Matrix input
	2302	MTWRIT	–	✓	Writing a string into the memory card
	0117	MUL16	MUL32	✓	Multiplying binary numbers for 16-bit/32-bit
	2300	MWRIT	–	✓	Writing data from the PLC to the memory card
	0805	MXOR	–	✓	Matrix exclusive OR operation
N	0210	NEG	DNEG	✓	Two's complement
	1301	NEXT	–	–	End of the nested loop
	0305	NMOV	DNMOV	✓	Transferring data to several devices

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1115	NSFL	–	✓	Shifting **n** registers to the left
	1114	NSFR	–	✓	Shifting **n** registers to the right
O	0049	OR$<>	–	–	S1≠S2
	0048	OR$=	–	–	S1＝S2
	0815	OR&	DOR&	–	S1&S2
	0817	OR^	DOR^	–	S1^S2
	0816	OR|	DOR|	–	S1|S2
	0016	OR<	DOR<	–	S1＜S2
	0017	OR<=	DOR<=	–	S1≦S2
	0013	OR<>	DOR<>	–	S1 ≠ S2
	0012	OR=	DOR=	–	S1＝S2
	0014	OR>	DOR>	–	S1＞S2
	0015	OR>=	DOR>=	–	S1≧S2
	0082	ORZ<	DORZ<	–	|S1-S2|＜|S3|
	0083	ORZ<=	DORZ<=	–	|S1-S2|≦|S3|
	0079	ORZ<>	DORZ<>	–	|S1-S2|≠|S3|
	0078	ORZ=	DORZ=	–	|S1-S2|=|S3|
	0080	ORZ>	DORZ>	–	|S1-S2|＞|S3|
	0081	ORZ>=	DORZ>=	–	|S1-S2|≧|S3|
P	1403	PUSTAT	–	–	Reading PU module output state
	1009	PWD	–	–	Pulse width detection
	2702	PWM	DPWM	–	Pulse width modulation
R	0703	RAMP	DRAMP	–	Ramp signal
	1517	RAND	–	✓	Random number
	0903	RCL	DRCL	✓	Rotating to the left with the carry flag
	0901	RCR	DRCR	✓	Rotating to the right with the carry flag
	0600	REF	–	✓	Refreshing the I/O
	0602	REFF	–	✓	Refreshing the I/O filtering time
	0207	RMOV	–	✓	Converting a 32-bit value into a 16-bit value
	0902	ROL	DROL	✓	Rotating to the left
	0900	ROR	DROR	✓	Rotating to the right
	1000	RST	DRST	–	Resetting the contact or clearing the register
	2809	RSTD	–	–	Sending Reset or NMT command
S	0216	SCAL	DSCAL	✓	Scale value operation
	0222	SCLM	DSCLM	✓	Multi-point area ratio operation
	2203	SCLOSE	–	✓	Closing the socket

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	0217	SCLP	DSCLP	✓	Parameter type of scale value operation
	2209	SCONF	–	✓	Setting TCP/UDP socket parameters
	1204	SEGD	–	✓	Seven-segment decoding
	1704	SEGL	–	–	Seven-segment display with latches
	1200	SER	DSER	✓	Searching the data
	2501	SFCPSE	–	–	Causing SFC to pause
	2500	SFCRUN	–	–	Enabling SFC
	2502	SFCSTP	–	–	Stopping SFC
	1107	SFDEL	–	✓	Deleting the data from the data list
	1108	SFINS	–	✓	Inserting the data into the data list
	1111	SFL	DSFL	✓	Shifting the values of the bits in the registers by n bits to the left
	1106	SFPO	–	✓	Reading the latest data from the data list
	1110	SFR	DSFR	✓	Shifting the values of the bits in the registers by n bits to the right
	1105	SFRD	–	✓	Shifting the data and reading it from the word device
	1101	SFTL	–	✓	Shifting the states of the devices to the left
	1100	SFTR	–	✓	Shifting the states of the devices to the right
	1104	SFWR	–	✓	Shifting the data and writing it into the word device
	0309	SMOV	–	✓	Transferring the digits
	2122	SPLIT	–	✓	Splitting a string
	2200	SOPEN	–	✓	Opening the socket
	1205	SORT	DSORT	✓	Sorting the data
	1511	SQR	DSQR	✓	Square root of a binary number
	2201	SSEND	–	✓	Sending the data through the socket
	0702	STMR	–	–	Special timer
	1201	SUM	DSUM	✓	Number of bits whose states are ON
	0308	SWAP	DSWAP	✓	Exchange the high byte with the low byte
T	1603	T-	–	✓	Subtracting a time
	1602	T+	–	✓	Adding a time
	1605	TCMP	–	✓	Comparing a time
	2401	TKOFF	–	✓	Disabling a cyclic task
	2400	TKON	–	✓	Enabling a cyclic task
	1700	TKY	DTKY	–	Ten key input
	1001	TMR	–	–	16-bit timer (unit: 100 ms)
	1002	TMRH	–	–	16-bit timer (unit: 1 ms)
	1011	TMRM	–	–	16-bit timer (unit: 10 ms)

Classification	API	Instruction code		Pulse instruction	Function
		16-bit	32-bit		
	1401	TO	DTO	✓	Writing the data to the control register in the special module
	2718	TPO	–	–	The position planning table controls the output
	1600	TRD	–	✓	Reading the time
	0701	TTMR	–	–	Teach mode timer
	1601	TWR	–	✓	Writing the time
	1606	TZCP	–	✓	Time zone comparison
U	1216	UNI	–	✓	Uniting the 16-bit data
V	1814	VFDRW	–	–	Serial communication instruction exclusively for Delta AC motor drive
W	0800	WAND	DAND	✓	Logical AND operation
	1608	WWON	–	–	Weekly working time setup
	1900	WDT	–	✓	Watchdog timer
	0808	WINV	DINV	✓	Logical reversed INV operation
	0802	WOR	DOR	✓	Logical OR operation
	1103	WSFL	–	✓	Shifting data in the word devices to the left
	1102	WSFR	–	✓	Shifting data in the word devices to the right
	1217	WSUM	DWSUM	✓	Getting the sum
	0804	WXOR	DXOR	✓	Logical exclusive OR operation
X	0306	XCH	DXCH	✓	Exchanging data
	0709	XCMP	–	–	Setup for comparing the inputs of multiple work stations
Y	0710	YOUT	–	–	Comparing the outputs of multiple work stations
Z	0055	ZCP	DZCP	✓	Zone comparison
	1223	ZONE	DZONE	✓	Controlling the zone
	2806	ZRNC	–	–	Servo homing
	2810	ZRNM	–	–	Setting the homing mode for Delta servo
	1206	ZRST	–	✓	Resetting the zone

4

Chapter 4 Instruction Structure

Table of Contents

4.1 Applied Instructions - API Description

This section describes the way this manual documents each API instruction. Every instruction has its own instruction code and API number. The instructions are divided into sections based on the related functions of the instructions, so that all the arithmetic instructions are in one section, and all the comparison instructions are in another section. The following example uses the MOV instruction. The API number of the instruction in the table is 0300, the instruction code is MOV, and the function is transferring data.

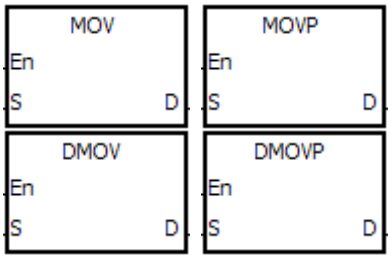
API	Instruction code			Operand	Function
0300	D	MOV	P	**S**、**D**	Transferring the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○	○	○	○		○
D		●			●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●		●		●	●	
D		●	●		●	●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol:



S : Data source

D : Data destination

1. The devices used by the instruction are listed in the operand column. **S**, **D**, **n**, and **m** are used as the operands according to their functions. When more than one operand is used, and these operands share the same function, they are differentiated by subscripts; for example, $\mathbf{S_1}$, and $\mathbf{S_2}$.
2. If you can use a 16-bit instruction as a 32-bit instruction, the letter **D** prepended to the 16-bit instruction code to form the 32-bit instruction form. If you can use the instruction as a pulse instruction, the letter **P** is appended to the instruction code. For example, "D***P" in which "***" is the instruction code indicates a 32-bit pulse instruction.
3. **F** in the operand area indicates a single precision floating point number (32-bit).
4. The solid circle ● indicates that the device can be modified by an index register, and the hollow circle ○ indicates that the device cannot be modified by an index register. For example, the data register designated by the operand **S** can be modified by an index register.
5. The applicable model is indicated in the table. You can check whether you can use the instruction as a pulse instruction, a 16-bit instruction, a 32-bit instruction, or a 64-bit instruction according to the information in the table.
6. If you want to use an instruction in a function block, and the output, input, and data devices are supported in the operands, you have to use the pointer registers. AS indicates that to use a timer, a 16-bit counter, and a 32-bit

counter that are supported in the operands, you have to use the timer pointer register, the 16-bit counter pointer register, and the 32-bit counter pointer register. Refer to Sections 4.4–4.7 for more information or Section 7.2.4 in the ISPSoft manual.

7. The symbols representing the MOV instruction in ISPSoft are:

   MOV, MOVP, DMOV, and DMOVP are the Instruction codes for this instruction

   En: Enable

   S: The data source (the applicable format of the operand is a word/double word.)

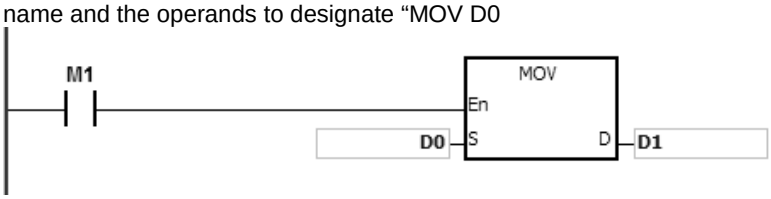
   D: The data destination (the applicable format of the operand is a word/double word.)

Applied instructions composition

Some applied instructions are composed of instruction codes. For example, the EI, DI, and WDT instructions; however, most applied instructions consist of instruction codes and several operands.

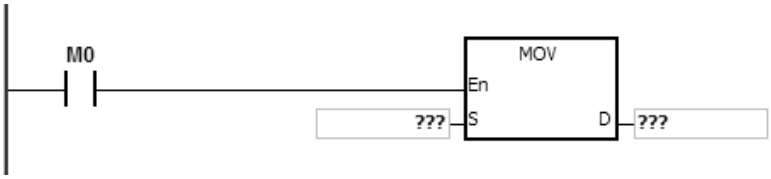
Every applied instruction has its own API number and instruction code. For example, the instruction code API 0300 is the MOV (transfer data) instruction. You can enter an applied instruction in three ways.

Enter the instruction directly: you can enter the instruction in ISPSoft. For the MOV instruction, enter the instruction name and the operands to designate “MOV D0



Enter the instruction by dragging: you can drag the MOV instruction from **APIs** in ISPSoft to the ladder diagram editor.

Enter the instruction from the toolbar: you can click **API/FB Selection** on the toolbar in ISPSoft, and then click **API**. Finally, click the MOV instruction in **Data Transfer**.



S	Source operand If there is more than one source operand, the source operands are represented by subscript (for example **S_1**, **S_2**).
D	Destination operand If there is more than one destination operand, the destination operands are represented by subscript (for example, **D_1**, **D_2**).
If the operand only can be a constant K/H or a register value, it is represented by **m**, **m_1**, **m_2**, **n**, **n_1**, or **n_2**.	

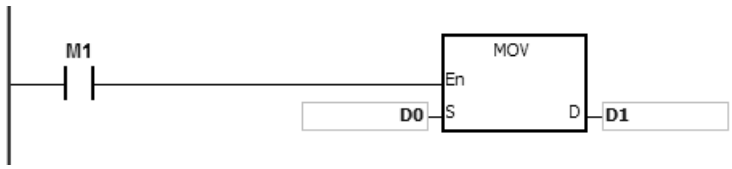
The length of the operand (6-bit, 32-bit, or floating-point number instructions):

16-bit or 32-bit instructions

Operand values in instructions are divided into 16-bit values and the 32-bit values. In order to process data of difference lengths, the instructions are divided into 16-bit and 32-bit instructions. To differentiate a 32-bit instruction from the 16-bit form, a D is added in front of the 16-bit instruction code (16-bit MOV and 32-bit DMOV).

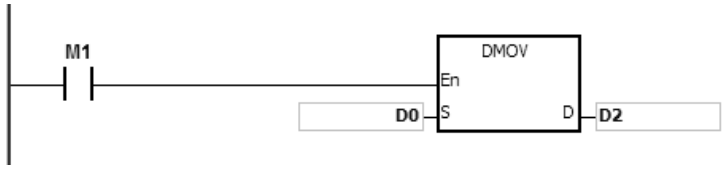
The floating-point number instruction

16-bit MOV instruction



When M1 is ON, the data in D0 is transferred to D1.

32-bit DMOV instruction

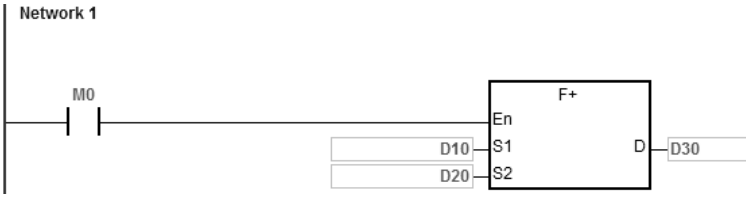


When M1 is ON, the data in (D1, D0) is transferred to (D3, D2).

Floating-point number instructions

Floating-point number instructions support 32-bit floating-point number instructions that correspond to the single-precision floating-point number instructions. Refer to Chapter 2 for more information about floating-point numbers.

32-bit single-precision floating-point number F+ instruction

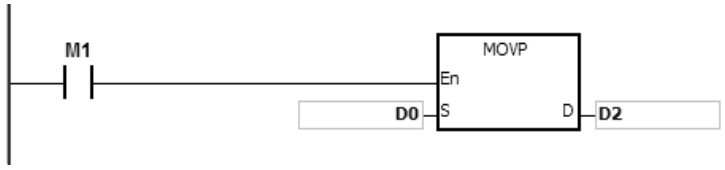


When M0 is ON, the data in (D11, D10) and (D21, D20) is transferred to (D31, D30).

Continuous execution and pulse execution of instructions

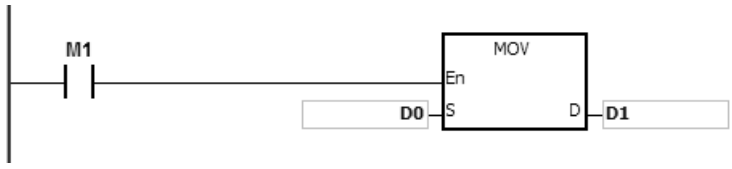
1. Instruction execution can be divided into continuous and pulse execution. You can reduce the scan cycle with pulse instructions because when the instruction is not executed, less time is needed to execute the program.
2. The pulse function allows the related instruction to enable the rising edge-triggered control input. The instruction is ON for one scan cycle.
3. If the control input stays ON, and the related instruction is not executed, the control input must be switched from OFF to ON again in order to execute the instruction.
4. The following shows the difference between pulse and continuous instruction:

Pulse execution



When M1 switches from OFF to ON, the MOVP instruction is executed once. The instruction is not executed again in the scan cycle. Therefore, it is called a pulse instruction.

Continuous execution



Whenever M1 is ON during the scan cycle, the MOV instruction is executed once. Therefore, the instruction is called a continuous instruction.

When the conditional contact M1 is OFF, neither instruction is executed, and the value in the destination operand D does not change.

4.2 Operand Usage Description

There are 2 types of operands in the DVP-ES2 Series: user-defined and system-defined.

User-defined operands

- Input relays: X0–X377 (octal format)
- Output relays: Y0–Y377 (octal format)
- Internal relays: M0–M8191
- Stepping relays: S0–S2047
- Timers: T0–T511
- 16-bit counters: C0–C511
- 32-bit counters: HC0–HC255
- Data registers: D0–D29999 or D0.0–D29999.15
- File registers: FR0–FR65535
- Special auxiliary flags: SM0–SM4095
- Special data registers: SR0–SR2047
- Index registers: E0–E9
- Constants: The decimal constants are indicated by K, and the hexadecimal constants are indicated by 16#.
- Strings: “$”
- Floating-point numbers: The single-precision floating-point numbers are indicated by F.
- The length of the data in one register is generally 16 bits. If you want to store 32-bit data in the register, designate two consecutive registers for the data.
- If the operand in a 32-bit instruction uses D0, it occupies the 32-bit data register composed of (D1, D0). D1 represents the higher 16 bits, and D0 represents the lower 16 bits. The same rule applies to the timer and the 16-bit counter.
- When you use the 32-bit counter HC as the data register, it can only be used by the operand in a 32-bit instruction.
- You can only use index registers in 16-bit instructions.

Refer to Chapter 2 Devices for more information.

System-defined operands

- The system assigns the variables to declare such as BOOL, WORD, INT and so on: U0–U16387 and W0–W29999.
- To start or stop a task use the TK0–TK31 instructions.

The following table lists the pointer type variable symbols, the supporting devices and usage.

Pointer type	Usage	
General pointer (Pointer)	Device range	PR0–PR15 · PR0.0–PR15.15
	Maximum quantity	Use up to 16 pointers in each function block
	Can be assigned to	Variable symbols of WORD/DWORD/LWORD/INT/DINT/LINT types or data register, input relay or output relay devices (e.g. X0, Y0, etc.)
Pointer for a timer (T_POINTER)	Device range	TR0–TR7
	Maximum quantity	Use up to 8 pointers in each function block
	Can be assigned to	Variable symbols of timer type or timer type devices
Pointer for a counter (C_POINTER)	Device range	CR0–CR7
	Maximum quantity	Use up to 8 pointers in each function block
	Can be assigned to	Variable symbols of counter type or counter type devices
Pointer for a high-speed counter (HC_POINTER)	Device range	HCR0–HCR7
	Maximum quantity	Use up to 8 pointers in each function block
	Can be assigned to	Variable symbols of 32-bit counter type or 32-bit counter type devices

4.3 Restrictions on the Use of Instructions

- You can use the following instructions only in function blocks:

  API0065 CHKADR, FB_NP, FB_PN, NED, ANED, ONED, PED, APED, OPED

- You cannot use the following instruction in interrupt tasks:

  GOEND

- You cannot use the in function blocks:

  LDP, ANDP, ORP, LDF, ANDF, ORF, PLS, PLF, NP, PN, MC/MCR, GOEND and all pulse instructions in applied instructions.

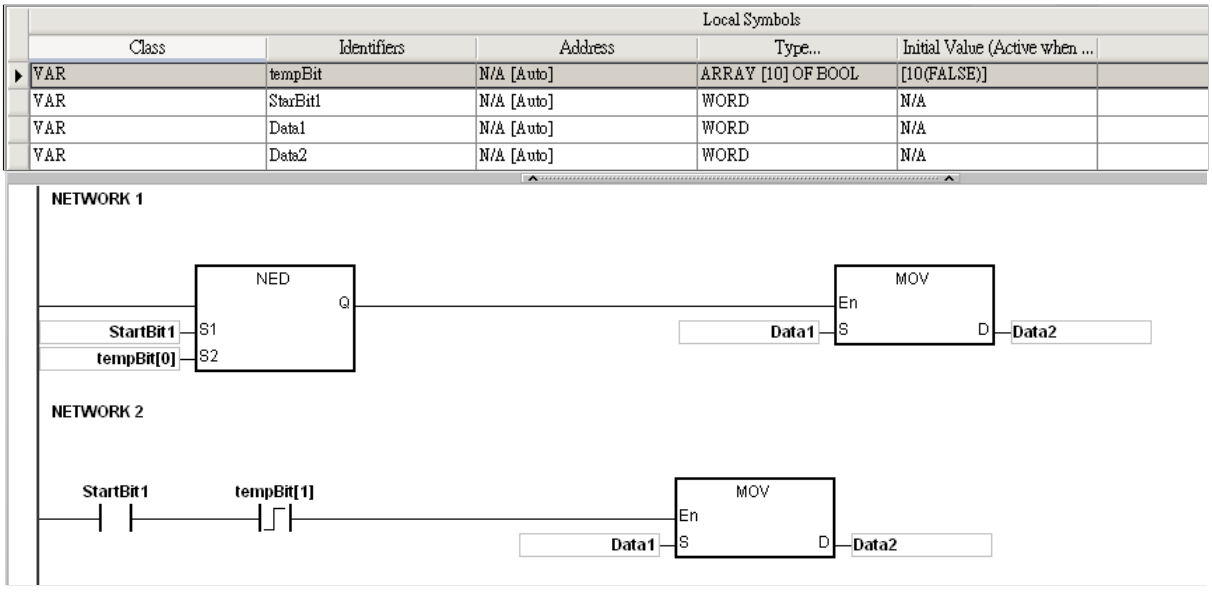
  If you want to use some of the instructions mentioned above, you can use the substitute instructions in the following table.

Instruction which cannot be used in the function block	Substitute instruction in the function block
LDP/ANDP/ORP	PED/APED/OPED
LDF/ANDF/ORF	NED/ANED/ONED
PLS	-
PLF	-
NP	FB_NP
PN	FB_PN
MC	-
MCR	-
All pulse instructions in applied commands	*1

*1: Pulse instructions cannot be used in function blocks. If you want to get the function of the pulse instruction in a function block, refer to the following example.

Example:

1. First, declare 10 bit variables tempBit[10] to be used in the system.
2. When StartBit1 switches from OFF to ON, method 1 (network 1) and method 2 (network 2) can only execute the MOV instruction once; you can choose which one to use.
3. You cannot use the variable tempBit in the system more than once.

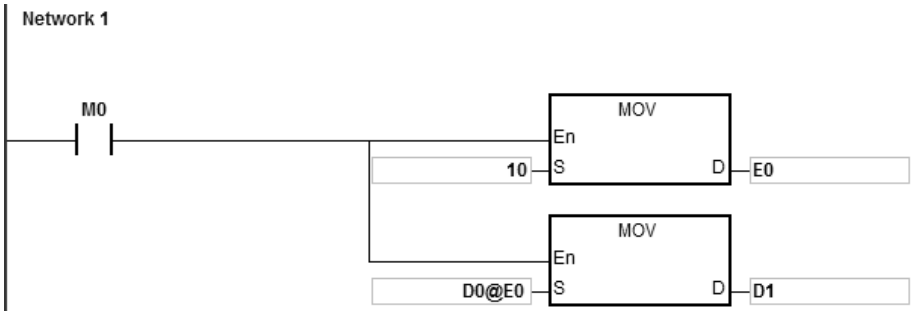


4.4 Index Registers

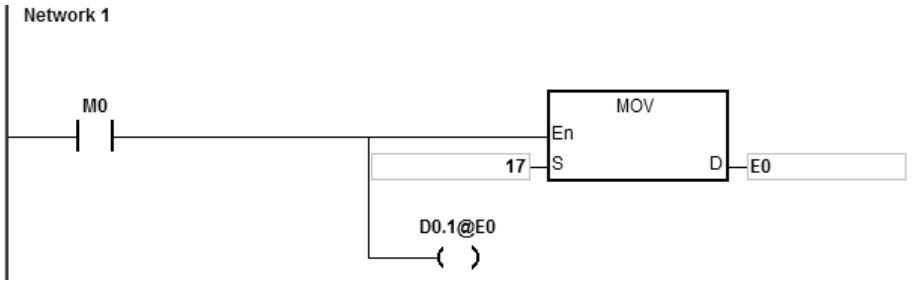
Index registers are 16-bit data registers. They are similar to the general register in that you can read data from them and write data into them. However, they are mainly used as index registers. The range of index registers is E0–E9. It is not recommended to use the index registers for global variables; they can only be used for partial variables and for temporary.

Index registers are used as follows.

1. Using the register name to modify the device:



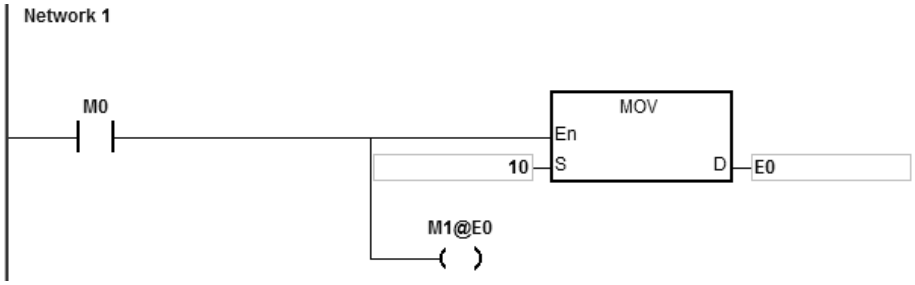
When M0 is ON, E0=10, E1=17, D1@E0=D (1+10)=D11, D11 is ON.



NOTE 1: DVP-ES3 Series support using the register name to modify the device; for example, D0.1@E0 but does not support 2-layered modification for example, D0@E1.1@E0.

NOTE 2: When E0=17, D0.1@E0=D0.(1+17)=D1.2, and D1.2 is ON. The bit part 1@E0=(1+17)=18. However, the maximum bit number is 15. Since m=18/16=1 and the remainder is 2, the last modification result is D (0+1).2=D1.2. D1.2 is ON.

When M0 is ON, E0=10, and M1@E0=M (1+10)=M11. M11 is ON.



2. Declaring the variables first, and then modifying the device:

   - Declare the three variables StartBit, Var1, and Var2 in ISPSoft.

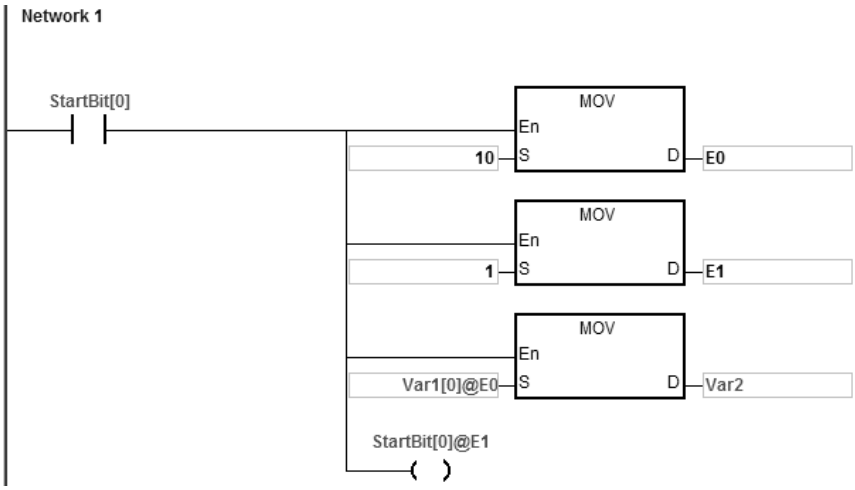
     The type of StartBit is a Boolean array, and its size is 2 bits. The range is from StartBit[0] to StartBit[1].

The type of Var1 is a word array, and its size is 11 words. The range is from Var1[0] to Var1[10].

The type of Var2 is a word, and its size is one word.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	N/A	
VAR	Var1	N/A [Auto]	ARRAY [11] OF WORD	N/A	
VAR	Var2	N/A [Auto]	WORD	N/A	

- When StartBit[0] is ON, E0=10, E1=1, Var1[0]@E0=Var1[10], Var2=Var1[10], and StarrtBit[0]@E1=StartBit[1]. StartBit[1] is ON.

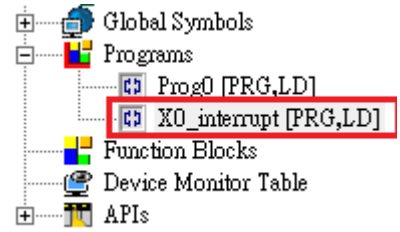


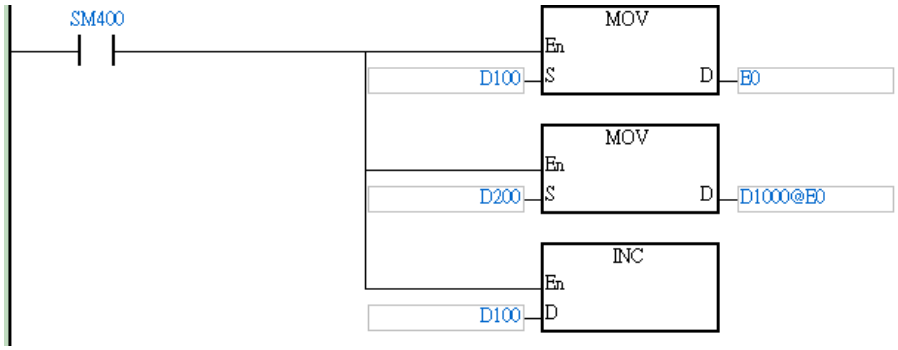
Additional remark: When you declare the variables in ISPSoft, and the variables are added to the contents of the registers to form the addresses to the actual data, you must note the addresses to prevent the program from being executed incorrectly.

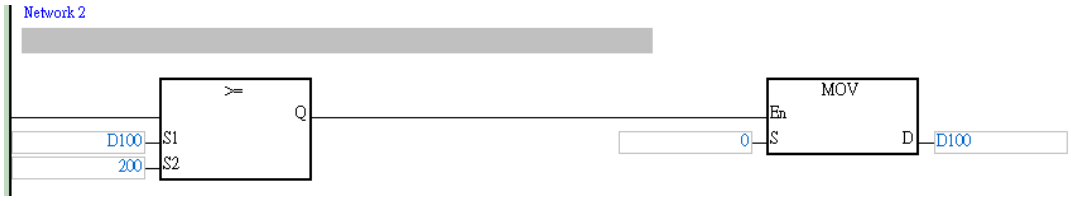
3. Using index registers in an interrupt instruction:

- The index registers E0~E9 in the main program (e.g. POU, SFC, and FB) are different from the ones (index registers E0~E9) in the interrupt program. They are independent groups of index registers and they work separately.
- You can use ISPSoft to monitor the values of index registers in the main program.
- Application: If you use E0 in the main program and in the interrupt program. When an interrupt occurs, before executing the interrupt program, the system stores the value of E0 in the main program. After the interrupt program is executed, the value of E0 in the main program will be retained. So that the interrupt program will not affect the value of E0 in the main program and the result of the main program execution can stay unaffected.
- Refer to the following example, if you need to monitor the values of index registers E0-E9 in the interrupt program, or if the main program is also using the values of index registers E0-E9 in the interrupt program. (Note: It is NOT recommended to use index registers as global variables. If you need to use it as a global variable, you need to declare or define other devices to use it as a global variable.)

- Example: Declare D100 as the variable of the index register E0. And this example use MOVE instruction to demonstrate. Whenever an external interrupt occurs at X0, the value in D200 accumulates the occurrences in the values in D1000 ~ D1199.







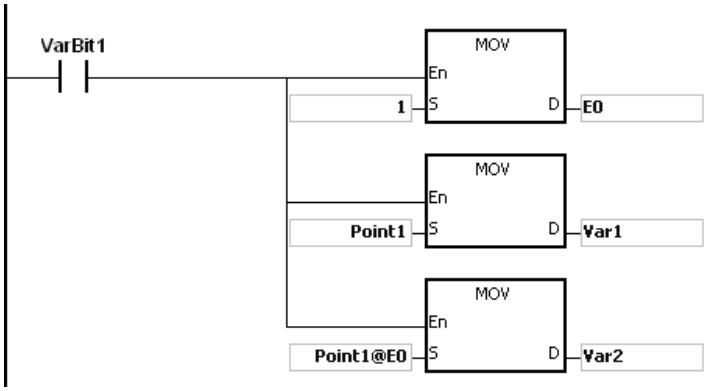
4.5 Pointer Registers

- ISPSoft supports function blocks. When the variable declaration type is VAR_IN_OUT, and the data type is POINTER, the variable is a pointer register. The value in the pointer register can refer directly to the value stored in a device X, Y, or D; and the pointer register can point to the address associated with the variable set automatically in ISPSoft.
- You can declare 16 pointer registers in every function block. The range is PR0–PR15, or PR0.0–PR15.15.

Example:

1. Create a program organization unit (POU) in ISPSoft.
2. Create a function block called FB0.

3. Create the program in the function block FB0.



4. Declare the variable in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, Point1 as the identifier, POINTER as the data type. The variable is the pointer register.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
	VAR	Var1	N/A [Auto]	WORD	0	
	VAR	Var2	N/A [Auto]	WORD	0	
▶	VAR_IN_OUT	Point1	N/A [Auto]	POINTER	N/A	

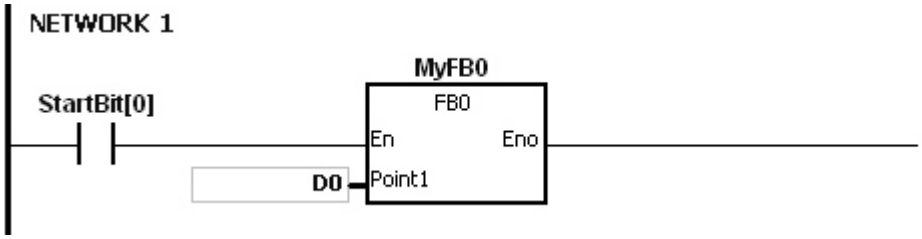
5. Declare the variable in the program organization unit (POU).

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	N/A	
	VAR	CVar1	N/A [Auto]	ARRAY [2] OF WORD	N/A	
▶	VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).

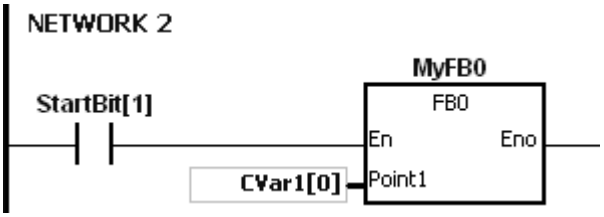
7. The program in the program organization unit (POU) operates as shown below.

   Network 1: When StartBit[0] is ON, the address of D0 is transmitted to Point 1 in FB0.



When VarBit1 in FB0 is ON, E0=1, Var1=D0, Point1@E0=D (0+1)=D1, and Var2=D1.

Network 2: When StartBit[1] is ON, the address of CVar1[0] is transmitted to Point1 in FB0.



Var2=CVar1[1] ◦ When VarBit1 in FB0 is ON, E0=1, Var1=CVar1[0], Point1@E0=CVar1 (0+1)=Cvar1[1], and Var2=CVar1[1].

4.6 Pointer Registers of Timers

- ISPSoft supports function blocks. If you want to use a timer in a function block, you must declare a timer pointer register in the function block. The address of the timer is transmitted to the timer pointer register when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is T_POINTER, the variable is the timer pointer register. The value in the timer pointer register can refer directly to the value stored in the device T, or in the variable which is the timer in ISPSoft.
- You can declare up to 8 timer pointer registers in every function block. The range is TR0–TR7.
- If you want to use an instruction in the function block, and the timer is supported by the operands, you must use a timer pointer register.

Example: using a timer in a function block.

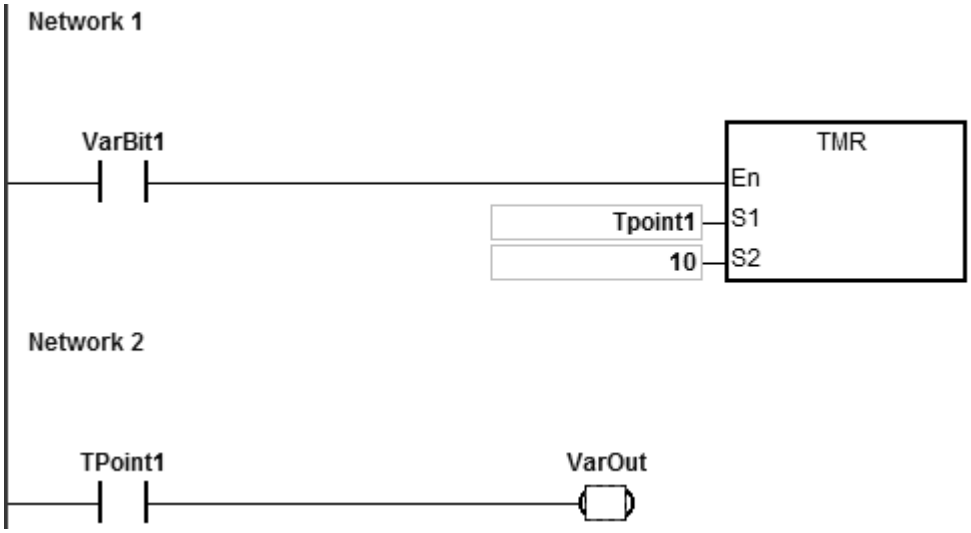
1. Create a program organization unit (POU) in ISPSoft.
2. Create a function block which is called FB0.

3. Declare the variable in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, TPoint1 as the identifier, and T_POINTER as the data type. The variable is the timer pointer register.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
	VAR_IN_OUT	TPoint1	N/A [Auto]	T_POINTER	N/A	
▶	VAR	VarOut	N/A [Auto]	BOOL	FALSE	

4. The program in the function block FB0 appears as shown below.



5. Declare the variable in the program organization unit (POU).

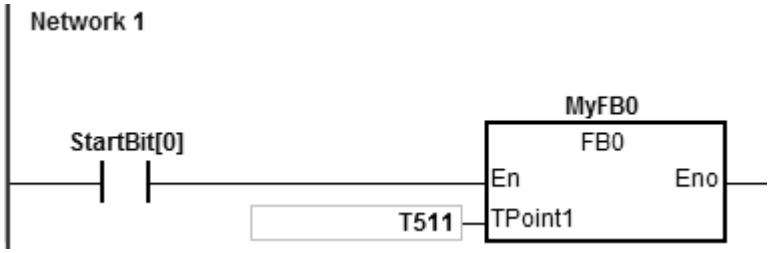
The data type of CVar1 should be TIMER.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
	VAR	CVar1	T0	TIMER	N/A	
▶	VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).

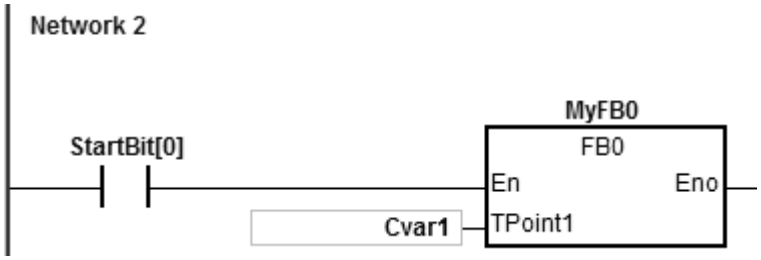
7. The program in the program organization unit (POU) operates as shown below.

Network 1: When StartBit[0] is ON, the address of T511 is transmitted to TPoint1 in FB0.



When VarBit1 in the FB0 is ON, the TMR instruction is executed, and TPoint1 (T511) starts counting. When the value of TPoint1 matches the TPoint1 setting, VarOut is ON.

Network 2: When StartBit[1] is ON, the address of CVar1[0] is transmitted to TPoint1 in FB0.



When VarBit1 in FB0 is ON, the TMR instruction is executed, and TPoint (CVar1) starts counting. When the value of TPoint1 matches the TPoint1 setting, VarOut is ON.

4.7 Pointer Registers for 16-bit Counters

- ISPSoft supports function blocks. If you want to use a 16-bit counter in a function block, you must declare a 16-bit counter pointer register in the function block. The address of the 16-bit counter is transmitted to the 16-bit counter pointer register when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is C_POINTE, the variable is the 16-bit counter pointer register. The value in the 16-bit counter pointer register can refer directly to the value stored in the device T, or in the variable which is the counter in ISPSoft.
- You can declare up to eight 16-bit counter pointer registers in every function block. The range is CR0–CR7.
- If you want to use an instruction in the function block, and the counter is supported by the operands, you have to use a 16-bit counter pointer register.

Example: using a 16-bit counter in a function block.

1. Create a program organization unit (POU) in ISPSoft.
2. Create a function block which is called FB0.

3. Declare the variable in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, CPoint1 as the identifier, C_POINTER as the data type. The variable is a 16-bit counter pointer register.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
▶	VAR_IN_OUT	CPoint1	N/A [Auto]	C_POINTER	N/A	

4. The program in the function block FB0 appears as shown below:

5. Declare the variable in the program organization unit (POU).

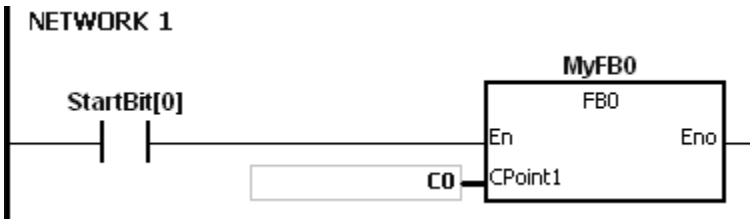
   The data type of CVar1 should be COUNTER.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
	VAR	CVar1	C1	COUNTER	N/A	
▶	VAR	MyFB0	N/A [Auto]	FB0	N/A	

6. Call the function block FB0 in the program organization unit (POU).

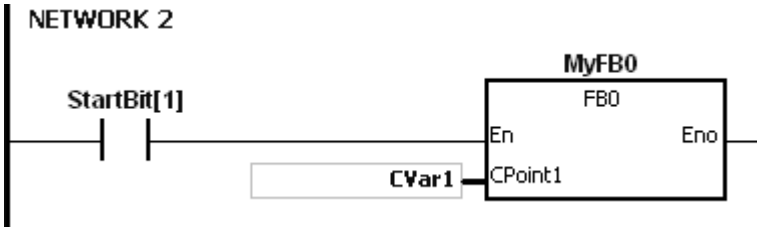
7. The program in the program organization unit (POU) operates as shown below.

   Network 1: When StartBit[0] is ON, the address of C0 is transmitted to CPoint1 in FB0.



   When VarBit1 in FB0 is ON, CPoint1 (C0) is ON.

   Network 2: When StartBit[1] is ON, the address of CVar1 is transmitted to CPoint1 in FB0.



   When VarBit1 in FB0 is ON, CPoint1 (CVar1) is ON.

4.8 Pointer Registers for 32-bit Counters

- ISPSoft supports function blocks. If you want to use a 32-bit counter in the function block, you must declare a 32-bit counter pointer register in the function block. The address of the 32-bit counter is transmitted to the 32-bit counter pointer register when the function block is called.
- When the variable declaration type is VAR_IN_OUT, and the data type is HC_POINTER, the variable is a 32-bit counter pointer register. The value in a 32-bit counter pointer register can refer directly to the value stored in the device HC or in the variable which is the counter in ISPSoft.
- You can declare up to eight 32-bit counter pointer registers in every function block. The range is HCR0–HCR7.
- If you want to use an instruction in the function block, and the 32-bit counter is supported by the operands, you must use the 32-bit counter pointer register.

Example: using a 32-bit counter in a function block.

1. Create a function block called FB0.

2. Declare the variable in the function block FB0.

   Choose VAR_IN_OUT as the declaration type, HCPoint1 as the identifier, HC_POINTER as the data type. The variable is a 32-bit counter pointer register.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	VarBit1	N/A [Auto]	BOOL	FALSE	
▶	VAR_IN_OUT	HCPoint1	N/A [Auto]	HC_POINTER	N/A	

3. The program in the function block FB0 appears as follows:

4. Declare the variable in the program organization unit (POU).

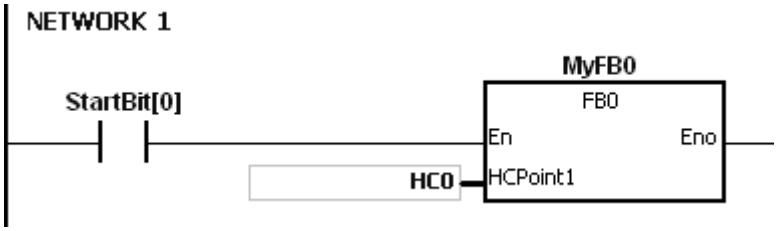
   The data type of CVar1 should be COUNTER, and you must fill in the address column with a valid address of the 32-bit counter.

	Local Symbols					
	Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
	VAR	StartBit	N/A [Auto]	ARRAY [2] OF BOOL	[2(FALSE)]	
	VAR	CVar1	HC1	COUNTER	N/A	
▶	VAR	MyFB0	N/A [Auto]	FB0	N/A	

5. Call the function block FB0 in the program organization unit (POU).

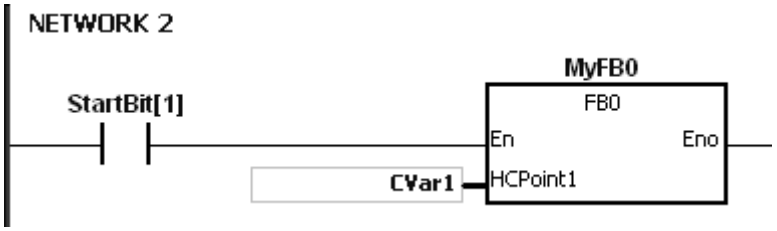
6. The program in the program organization unit (POU) operates as follows:

   Network 1: When StartBit[0] is ON, the address of HC0 is transmitted to HCPoint1 in FB0.



   When VarBit1 in FB0 is ON, HCPoint1 (HC0) is ON.

   Network: When StartBit[1] is ON, the address of CVar1 is transmitted to HCPoint1 in FB0.



   When VarBit1 in FB0 is ON, HCPoint1 (CVar1) is ON.

4.9 File Register

- DVP-ES3 Series PLC provides File registers (FR) for storing larger numbers of parameters.
- You can edit, upload, and download the parameters in the file registers with ISPSoft.
- The values in file registers can be read while the PLC is running. Refer to the MEMW instruction (API 2303) in the DVP-ES3 Series Programming Manual for more information about how to read and write parameters to file registers.

5

Chapter 5 Basic Instructions

Table of Contents

5.1 List of Basic Instructions

The following table lists the Basic instructions covered in this chapter.

Instruction code	Function	Operand	Operation time (μs)
LD/AND/OR	Loading contact A/Connecting contact A in series/Connecting contact A in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.025
LDI/ANI/ORI	Loading contact B/Connecting contact B in series/Connecting contact B in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.03
OUT	Driving the coil	DY, Y, M, SM, S, T, C, HC, D	0.04
SET	Keeping the device on	DY, Y, M, SM, S, T, C, HC, D	0.04
MC/MCR	Setting/Resetting the master control	N	0.24
LDP/ANDP/ORP	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.22
LDF/ANDF/ORF	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	DX, X, Y, M, SM, S, T, C, HC, D	0.22
PED/APED/OPED	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel	X, Y, M, SM, S, T, C, HC, D	0.22
NED/ANED/ONED	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	X, Y, M, SM, S, T, C, HC, D	0.22
PLS	Rising-edge output	Y, M, SM, S, D	0.22
PLF	Falling-edge output	Y, M, SM, S, D	0.22
INV	Inverting the logical operation result	–	0.22
NP	The circuit is rising edge-triggered.	–	0.24
PN	The circuit is falling edge-triggered.	–	0.24
FB_NP	The circuit is rising edge-triggered.	Y, M, S, D	0.24
FB_PN	The circuit is falling edge-triggered.	Y, M, S, D	0.24

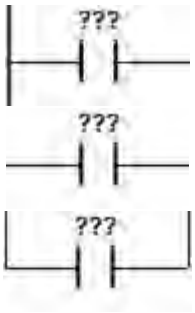
5.2 Basic Instructions

Instruction code	Operand	Function
LD/AND/OR	**S**	Loading contact A/Connecting contact A in series/Connecting contact A in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol



LD

AND

OR

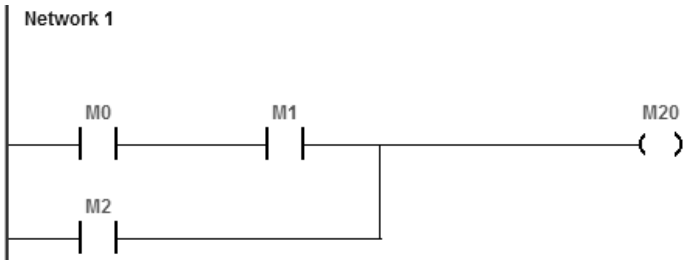
S : Specified device

Explanation

1. LD applies to contact A that starts from the main line or contact A which is the start of a contact circuit. Use it to save the current contents, and store the contact state in the accumulative register.
2. AND connects contact A in series. It reads the state of the contact that is specified as connected in series, and performs the AND operation with the previous logical operation result. It stores the final result in the accumulative register.
3. OR connects contact A in parallel. It reads the state of the contact that is specified as connected in parallel, and performs the OR operation with the previous logical operation result. It stores the final result in the accumulative register.

Example

1. Contact A of M0 is loaded, contact A of M1 is connected in series, contact A of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 are ON, or when M2 is ON, M20 is ON.

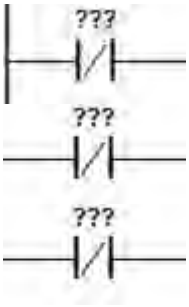


Instruction code	Operand	Function
LDI/ANI/ORI	**S**	Loading contact B/Connecting contact B in series/Connecting contact B in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol



LDI

ANI

ORI

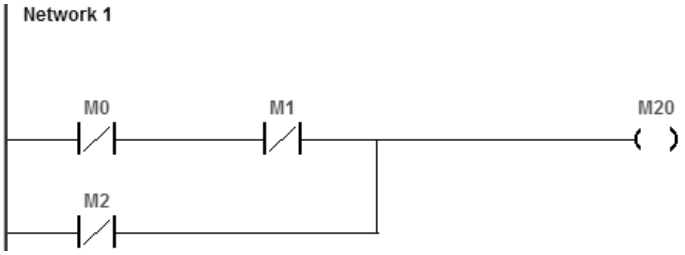
S : Specified device

Explanation

1. LDI applies to contact B that starts from the main line or contact B that is the start of a contact circuit. Use it to save the current contents, and store the contact state in the accumulative register.
2. ANI connects contact B in series. It reads the state of the contact that is specified as connected in series, and performs the AND operation with the previous logical operation result. It stores the final result in the accumulative register.
3. ORI connects contact B in parallel. It reads the state of the contact that is specified as connected in parallel, and performs the OR operation with the previous logical operation result. It stores the final result in the accumulative register.

Example

1. Contact B of M0 is loaded, contact B of M1 is connected in series, contact B of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 are ON, or when M2 is ON, M20 is ON.

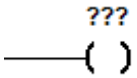


Instruction code	Operand	Function
OUT	**D**	Driving the coil

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D		●		●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol



D : Specified device

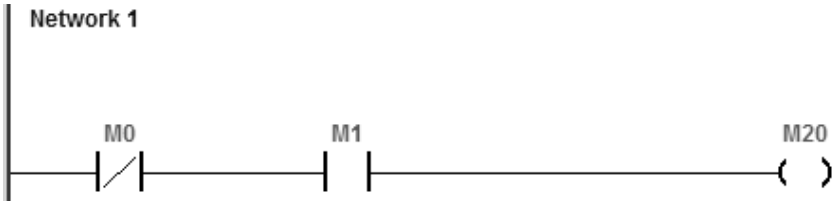
Explanation

1. The logical operation result prior to the application of the OUT instruction is output to the specified device.
2. The following table describes the action of the coil contact.

Operation result	OUT		
	Coil	Contact	
		Contact A (normally open)	Contact B (normally closed)
False	OFF	OFF	ON
True	ON	ON	OFF

Example

1. Contact B of M0 is loaded, contact A of M1 is connected in series, and the coil M20 is driven.
2. When M0 is OFF, and M1 is ON, M20 is ON.

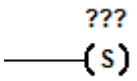


Instruction code	Operand	Function
SET	**D**	Keeping the device on

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D		●		●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol



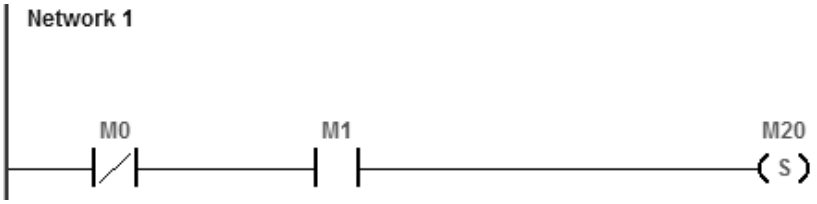
D : Specified device

Explanation

When the instruction SET is driven, the specified device is set to ON. It does not matter if the SET instruction is still driven, the specified device stays ON. You can set the specified device to OFF with the RST instruction.

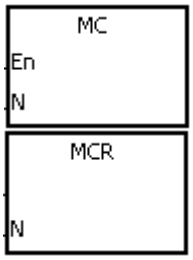
Example

1. Contact B of M0 is loaded, contact A of M1 is connected in series, and M20 stays ON.
2. When M0 is OFF, and M1 is ON, M20 is ON. Even if the operation result changes, M20 still stays ON.



Instruction code	Operand	Function
MC/MCR	**N**	Setting/Resetting the master control

Symbol



MC

MCR

N : Level of the nested program structure N0~N31

Explanation

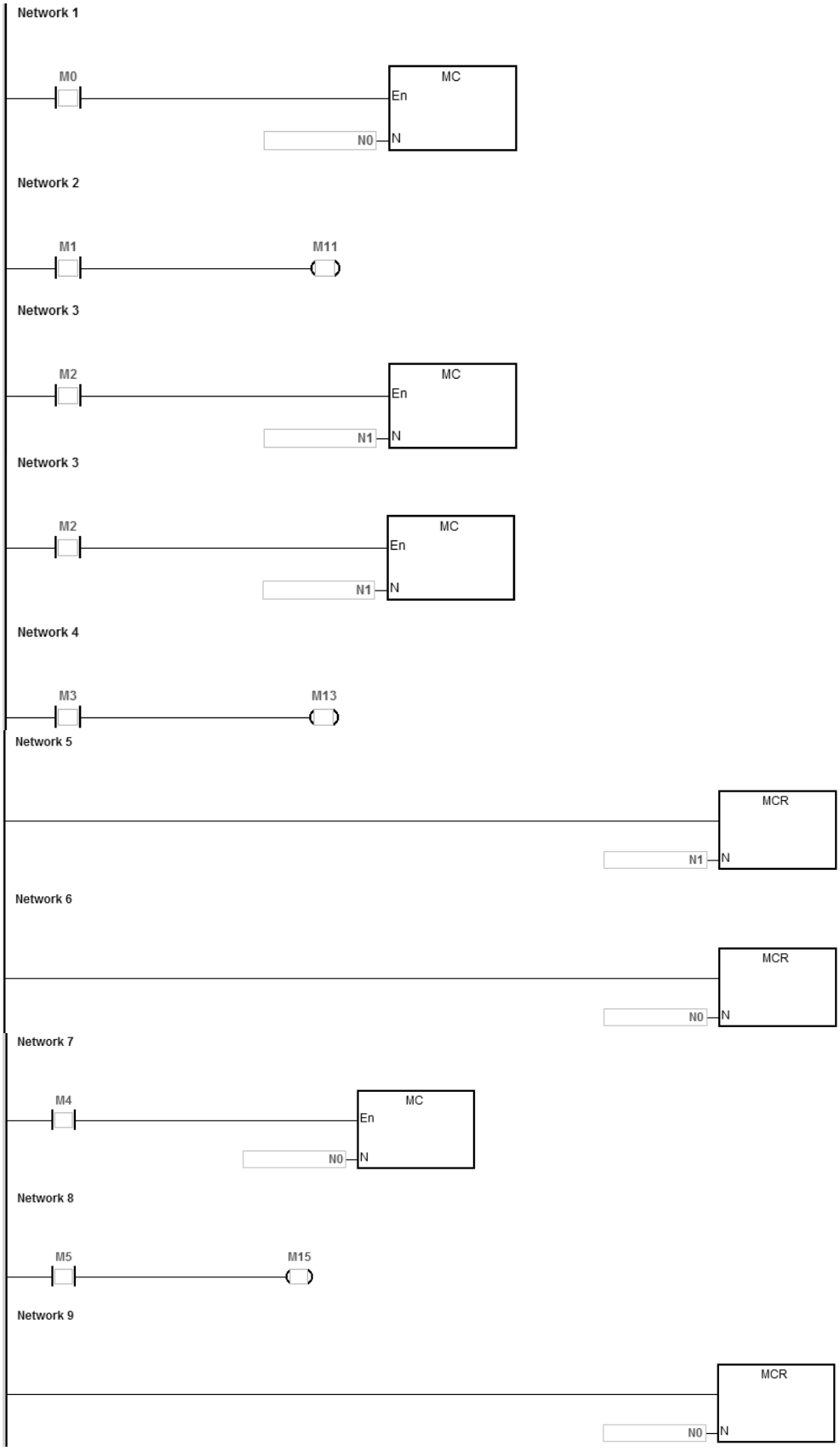
1. MC sets the master control. When the MC instruction is executed, the instructions between MC and MCR are executed as usual. When the MC instruction is OFF, the actions of the instructions between MC and MCR are as described in the following table.

Instruction type	Description
General-purpose timer	The timer value is reset to zero. The coil and the contact are OFF.
Timer in the function block	The timer value is reset to zero. The coil and the contact are OFF.
Accumulative timer	The coil is OFF. The timer value and the state of the contact remain the same.
Counter	The coil is OFF. The timer value and the state of the contact remain the same.
Coils driven by OUT	All coils are OFF.
Devices driven by SET and RST	The states of the devices remain the same.
Applied instruction	All applied instructions are not executed. The FOR/NEXT loop is still repeated N times, but the actions of the instructions inside the FOR/NEXT loop follow those of the instructions between MC and MR.

2. MCR resets the master control, and is placed at the end of the master control program. There should not be any contact instruction before MCR.

3. MC/MCR supports the nested program structure. There are at most 32 levels of nested program structures (N0–N31). Refer to the example below.

Example

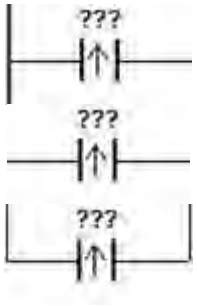

Network 1
M0
MC
En
N0
N
Network 2
M1
M11
Network 3
M2
MC
En
N1
N
Network 3
M2
MC
En
N1
N
Network 4
M3
M13
Network 5
MCR
N1
N
Network 6
MCR
N0
N
Network 7
M4
MC
En
N0
N
Network 8
M5
M15
Network 9
MCR
N0
N

Instruction code	Operand	Function
LDP/ANDP/ORP	**S**	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol



LDP

ANDP

ORP

S : Specified device

Explanation

1. LDP stores the current contents, and stores the rising-edge detection of the contact in the accumulative register.
2. ANDP connects the rising-edge detection of the contact in series.
3. ORP connects the rising-edge detection of the contact in parallel.
4. The system must scan LDP/ANDP/ORP to get the state of the device. Changes to the device state are not detected until LDP/ANDP/ORP is scanned the next time.
5. Use the corresponding PED, APED, and OPED instructions in subroutines.

Example

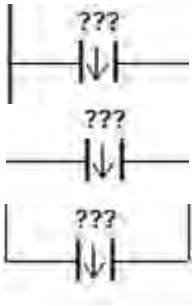
1. The rising-edge detection of M0 starts, the rising-edge detection of M1 is connected in series, the rising-edge detection of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 are switched from OFF to ON, or when M2 is switched from OFF to ON, M20 is ON for a scan cycle.

Instruction code	Operand	Function
LDF/ANDF/ORF	**S**	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S	●		●	●	●	○	●	●	●	●	●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol



LDF

ANDF

ORF

S : Specified device

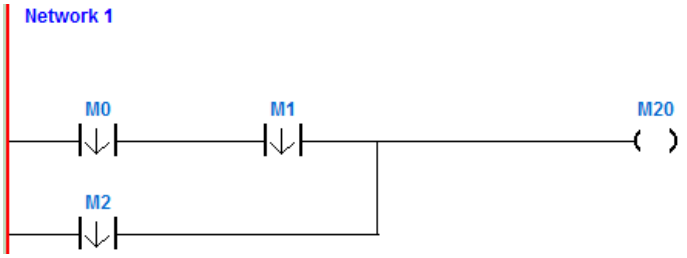
5

Explanation

1. LDF saves the current contents, and stores the contact falling-edge detection in the accumulative register.
2. ANDF connects the falling-edge detection of the contact in series.
3. ORP connects the falling-edge detection of the contact in parallel.
4. The system must scan LDF/ANDF/ORF to get the state of the device. Changes to the device state are not detected until LDF/ANDF/ORF is scanned the next time.
5. Use the corresponding NED, ANED, and ONED instructions in subroutines.

Example

1. The falling-edge detection of M0 starts, the falling-edge detection of M1 is connected in series, the falling-edge detection of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 switches from OFF to ON, or when M2 switches from OFF to ON, M20 is ON for a scan cycle.

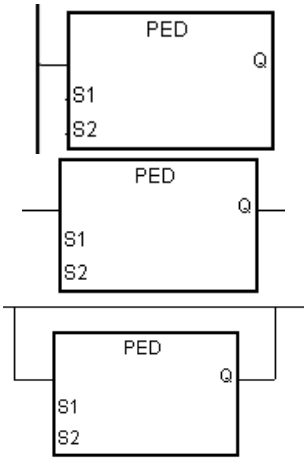


Instruction code	Operand	Function
PED/APED/OPED	S_1 · S_2	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S_1			●	●	●	○	●	●	●	●	●
S_2				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2	●												

Symbol



PED
APED
OPED

S_1 : Specified device

S_2 : For internal use

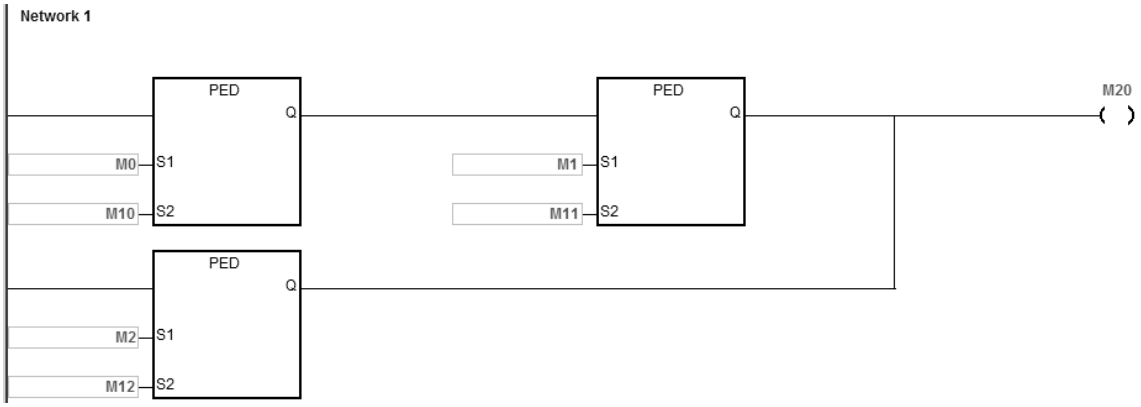
Q : Output the state of the operation result

Explanation

1. PED/APED/OPED correspond to LDP/ANDP/ORP. The only difference between PED/APED/OPED and LDP/ANDP/ORP is that you must specify the bit device **S_2** in which to store the previous state of the contact when PED/APED/OPED is executed. Do not use the device **S_2** repeatedly in the program. Otherwise, the wrong execution result appears.
2. APED connects the rising-edge detection of the contact in series.
3. OPED connects the rising-edge detection of the contact in parallel.
4. The system must scan PED/APED/OPED to get the state of the device. Changes to the device state are not detected until PED/APED/OPED is scanned the next time
5. You can use PED/APED/OPED only in function blocks.
6. The state of the operation result is automatically output after the instruction is executed. You do not need to use an input device for this.

Example

1. The rising-edge detection of M0 starts, the rising-edge detection of M1 is connected in series, the rising-edge detection of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 switch from OFF to ON, or when M2 switches from OFF to ON, M20 is ON for a scan cycle.

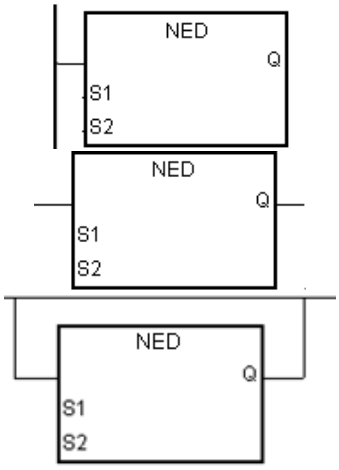


Instruction code	Operand	Function
NED/ANED/ONED	**S_1**、**S_2**	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S_1			●	●	●	○	●	●	●	●	●
S_2				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2	●												

Symbol



NED
ANED
ONED

S_1 : Specified device

S_2 : For internal use

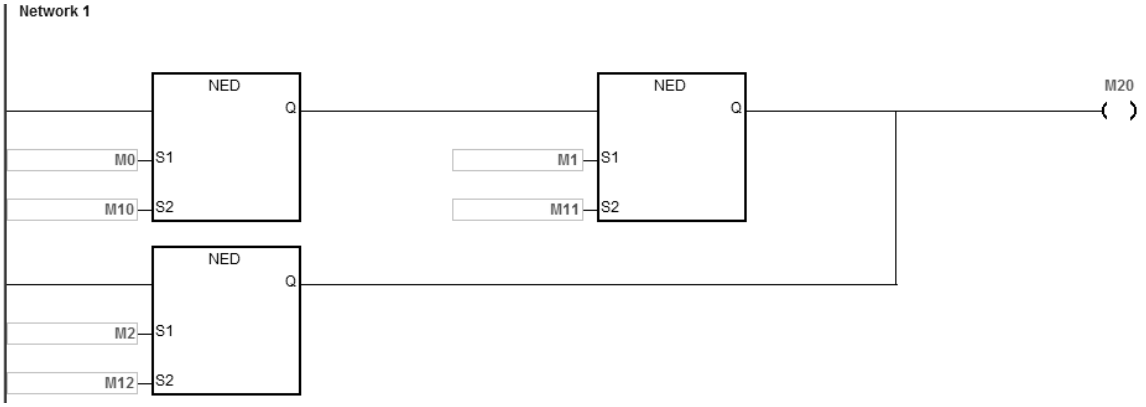
Q : Output the state of the operation result

Explanation

1. NED/ANED/ONED correspond to LDF/ANDF/ORF. The only difference between NED/ANED/ONED and LDF/ANDF/ORF is that you must specify the bit device **S_2** in which to store the previous state of the contact when NED/ANED/ONED is executed. Do not use the device **S_2** repeatedly in the program. Otherwise, the wrong execution result appears.
2. ANED connects the falling-edge detection of the contact in series.
3. ONED connects the falling-edge detection of the contact in parallel.
4. The system must scan NED/ANED/ONED to get the state of the device. Changes to the device state are not detected until NED/ANED/ONED is scanned the next time
5. You can use NED/ANED/ONED only in function blocks.
6. The state of the operation result is automatically output after the instruction is executed. You do not need to use input device for this.

Example

1. The falling -edge detection of M0 starts, the falling -edge detection of M1 is connected in series, the falling -edge detection of M2 is connected in parallel, and the coil M20 is driven.
2. When both M0 and M1 switch from OFF to ON, or when M2 switches from OFF to ON, M20 is ON for a scan cycle.

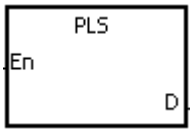


Instruction code	Operand	Function
PLS	**D**	Rising-edge output

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D				●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol



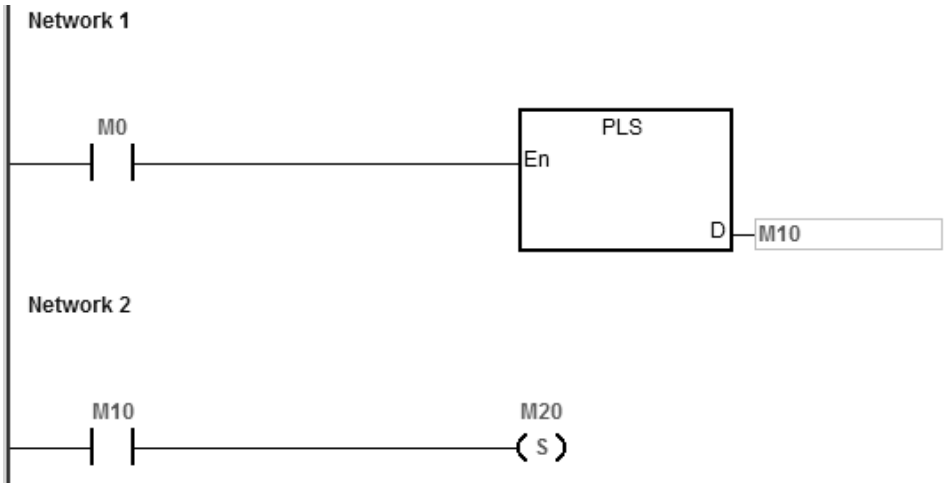
PLS | **D** : Specified device

Explanation

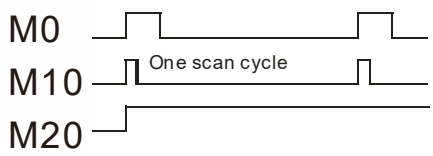
1. When the conditional contact switches from OFF to ON, the PLS instruction is executed, and the device D sends out a pulse for a scan cycle.
2. Do not use the PLS instruction in function blocks.
3. ISPSoft V3.06.01 supports data type D and the BOOL type variable for declaration.

Example

When M0 is ON, M10 is ON for a pulse time. When M10 is ON, M20 is set to ON.



Timing diagram



Instruction code	Operand	Function
PLF	**D**	Falling-edge output

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
D				●	●	○	●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Symbol

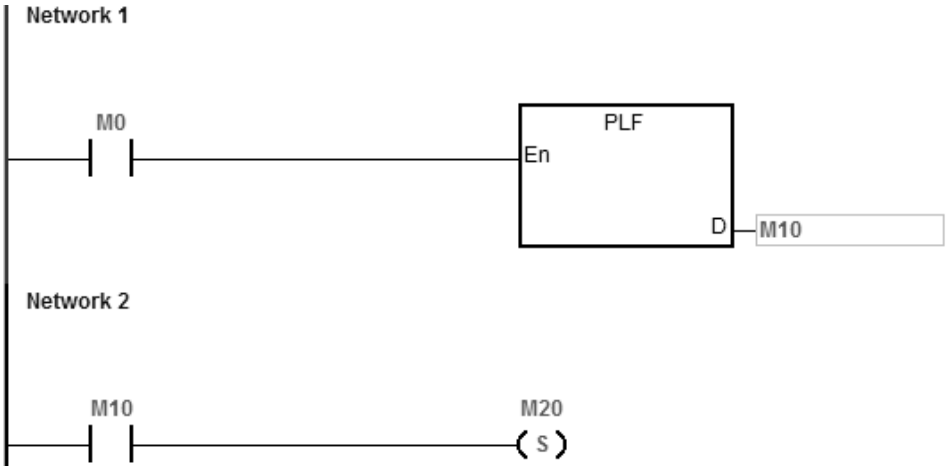
PLF | **D** : Specified device

Explanation

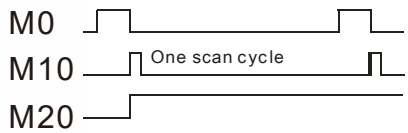
1. When the conditional contact switches from ON to OFF, the instruction PLF is executed, and the device D sends out a pulse for a scan cycle.
2. Do not use the instruction PLS in function blocks.
3. ISPSoft V3.06.01 supports data type D and the BOOL type variable for declaration.

Example

When M0 is ON, M10 is ON for a pulse time. When M10 is ON, M20 is set to ON.

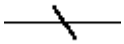


Timing chart



Instruction code	Operand	Function
INV	-	Inverting the logical operation result

Symbol

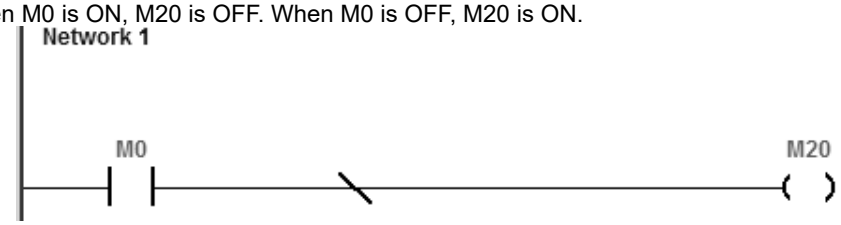


Explanation

The logical operation result preceding the INV instruction is inverted, and the inversion result stored in the accumulative register.

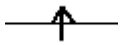
Example

When M0 is ON, M20 is OFF. When M0 is OFF, M20 is ON.



Instruction code	Operand	Function
NP	-	Triggering the circuit on the rising edge.

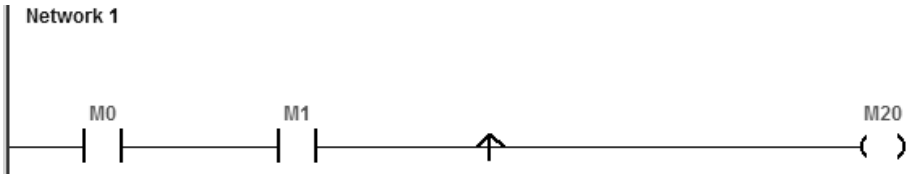
Symbol



Explanation

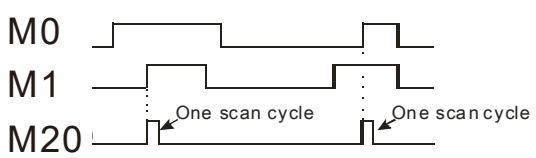
1. When the value in the accumulative register switches from 0 to 1, the NP instruction keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Use the FB_NP instruction in function blocks.

Example



Instruction		Operation
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
NP		The circuit is rising edge-triggered.
OUT	M20	The coil M20 is driven.

Timing diagram



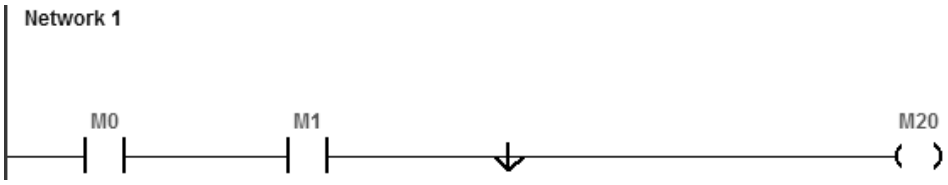
Instruction code	Operand	Function
PN	-	Triggering the circuit on the falling edge.

Symbol

Explanation

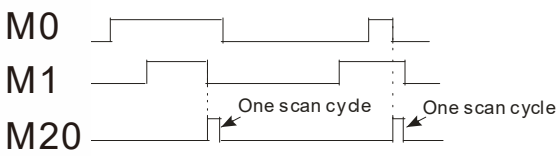
1. When the value in the accumulative register switches from 1 to 0, the PN instruction keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Use the FB_ PN instruction in function blocks.

Example



Instruction		Operation
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
PN		The circuit is falling edge-triggered.
OUT	M20	The coil M20 is driven.

Timing diagram

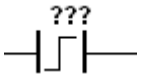


Instruction code	Operand	Function
FB_NP	**S**	Triggering the circuit on the rising edge.

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol

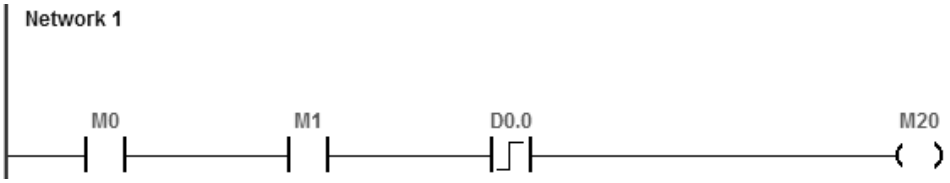


S ：For internal use

Explanation

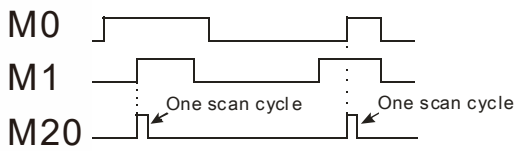
1. When the value in the accumulative register turns from 0 to 1, the FB_NP instruction keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Do not use **S** repeatedly in the program. Otherwise, the wrong execution result appears.
3. Use FB_NP only in function blocks.

Example



Instruction		Operation
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
FB_NP	D0.0	The circuit is rising edge-triggered.
OUT	M20	The coil M20 is driven.

Timing diagram

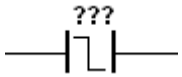


Instruction code	Operand	Function
FB_PN	**S**	Triggering the circuit on the falling edge.

Device	DX	DY	X	Y	M	SM	S	T	C	HC	D
S				●	●		●				●

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												

Symbol

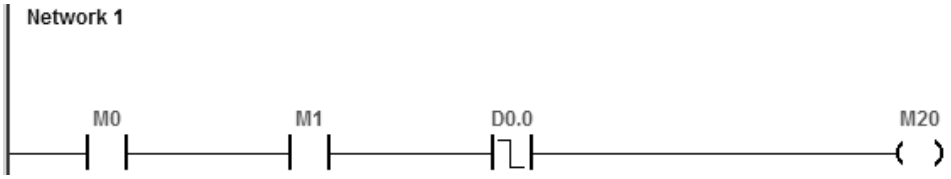


S ：For internal use

Explanation

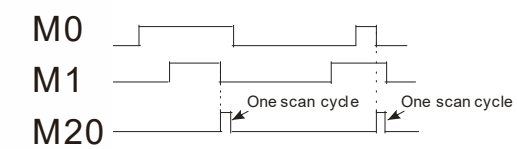
1. When the value in the accumulative register switches from 1 to 0, the FB_PN instruction keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Do not use **S** repeatedly in the program. Otherwise, the wrong execution result appears.
3. Use FB_PN only in function blocks.

Example



Instruction		Operation
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
FB_PN	D0.0	The circuit is falling edge-triggered.
OUT	M20	The coil M20 is driven.

Timing diagram



MEMO

6

Chapter 6 Applied Instructions

Table of Contents

6.1 Comparison Instructions

6.1.1 List of Comparison Instructions

The following table lists the Comparison instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0000	LD=	DLD=	–	S1 = S2
0001	LD<>	DLD<>	–	S1≠S2
0002	LD>	DLD>	–	S1 > S2
0003	LD>=	DLD>=	–	S1≧S2
0004	LD<	DLD<	–	S1 < S2
0005	LD<=	DLD<=	–	S1≦S2
0006	AND=	DAND=	–	S1 = S2
0007	AND<>	DAND<>	–	S1≠S2
0008	AND>	DAND>	–	S1 > S2
0009	AND>=	DAND>=	–	S1≧S2
0010	AND<	DAND<	–	S1 < S2
0011	AND<=	DAND<=	–	S1≦S2
0012	OR=	DOR=	–	S1 = S2
0013	OR<>	DOR<>	–	S1 ≠ S2
0014	OR>	DOR>	–	S1 > S2
0015	OR>=	DOR>=	–	S1≧S2
0016	OR<	DOR<	–	S1 < S2
0017	OR<=	DOR<=	–	S1≦S2
0018	–	FLD=	–	S1 = S2
0019	–	FLD<>	–	S1≠S2
0020	–	FLD>	–	S1 > S2
0021	–	FLD>=	–	S1≧S2
0022	–	FLD<	–	S1 < S2
0023	–	FLD<=	–	S1≦S2
0024	–	FAND=	–	S1 = S2
0025	–	FAND<>	–	S1≠S2
0026	–	FAND>	–	S1 > S2
0027	–	FAND>=	–	S1≧S2
0028	–	FAND<	–	S1 < S2
0029	–	FAND<=	–	S1≦S2
0030	–	FOR=	–	S1 = S2
0031	–	FOR<>	–	S1≠S2

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0032	–	FOR>	–	S1 > S2
0033	–	FOR>=	–	S1≧S2
0034	–	FOR<	–	S1 < S2
0035	–	FOR<=	–	S1≦S2
0036	LD$=	–	–	S1 = S2
0037	LD$<>	–	–	S1≠S2
0042	AND$=	–	–	S1 = S2
0043	AND$<>	–	–	S1≠S2
0048	OR$=	–	–	S1 = S2
0049	OR$<>	–	–	S1≠S2
0054	CMP	DCMP	✓	Comparing values
0055	ZCP	DZCP	✓	Zone comparison
0056	–	FCMP	✓	Comparing the floating-point numbers
0057	–	FZCP	✓	Floating-point zone comparison
0058	MCMP	–	✓	Matrix comparison
0059	CMPT=	–	✓	Comparing tables ON: =
0060	CMPT<>	–	✓	Comparing tables ON: <>
0061	CMPT>	–	✓	Comparing tables ON: >
0062	CMPT>=	–	✓	Comparing tables ON: ≧
0063	CMPT<	–	✓	Comparing tables ON: <
0064	CMPT<=	–	✓	Comparing tables ON: ≦
0065	CHKADR	–	–	Checking the address in a pointer register
0066	LDZ=	DLDZ=	–	|S1-S2|=|S3|
0067	LDZ<>	DLDZ<>	–	|S1-S2|≠|S3|
0068	LDZ>	DLDZ>	–	|S1-S2| > |S3|
0069	LDZ>=	DLDZ>=	–	|S1-S2|≧|S3|
0070	LDZ<	DLDZ<	–	|S1-S2| < |S3|
0071	LDZ<=	DLDZ<=	–	|S1-S2|≦|S3|
0072	ANDZ=	DANDZ=	–	|S1-S2|=|S3|
0073	ANDZ<>	DANDZ<>	–	|S1-S2|≠|S3|
0074	ANDZ>	DANDZ>	–	|S1-S2| > |S3|
0075	ANDZ>=	DANDZ>=	–	|S1-S2|≧|S3|

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0076	ANDZ<	DANDZ<	–	|S1-S2| < |S3|
0077	ANDZ<=	DANDZ<=	–	|S1-S2|≦|S3|
0078	ORZ=	DORZ=	–	|S1-S2|=|S3|
0079	ORZ<>	DORZ<>	–	|S1-S2|≠|S3|
0080	ORZ>	DORZ>	–	|S1-S2| > |S3|
0081	ORZ>=	DORZ>=	–	|S1-S2|≧|S3|
0082	ORZ<	DORZ<	–	|S1-S2| < |S3|
0083	ORZ<=	DORZ<=	–	|S1-S2|≦|S3|

6.1.2 Explanation of Comparison Instructions

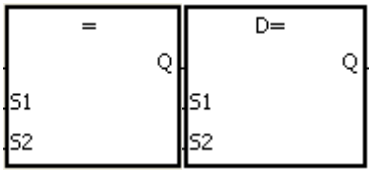
API	Instruction code			Operand	Function
0000-0005	D	LD※		S_1 · S_2	Comparing values LD※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking LD= and DLD= for example

S_1 : Data source 1

S_2 : Data source 1

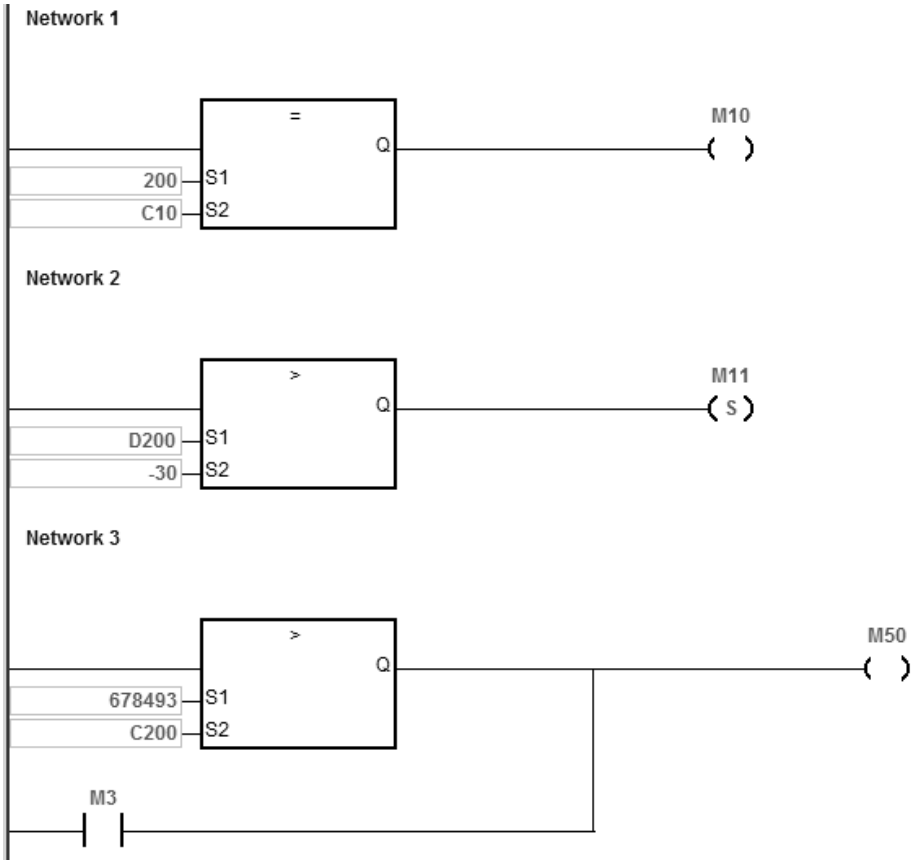
Explanation

1. These instructions compare the values in S_1 and S_2. Take the LD= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.
2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0000	LD =	**D**LD =	$S_1 = S_2$	$S_1 \neq S_2$
0001	LD < >	**D**LD < >	$S_1 \neq S_2$	$S_1 = S_2$
0002	LD >	**D**LD >	$S_1 > S_2$	$S_1 \leqq S_2$
0003	LD > =	**D**LD > =	$S_1 \geqq S_2$	$S_1 < S_2$
0004	LD <	**D**LD <	$S_1 < S_2$	$S_1 \geqq S_2$
0005	LD < =	**D**LD < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

1. When the value in C10 is equal to 200, M10 is ON.
2. When the value in D200 is greater than -30, M11 stays ON.
3. When the value in (C201, C200) is less than 678,493, or when M3 is ON, M50 is ON.


Network 1
=
Q
M10
200
S1
C10
S2
Network 2
>
Q
M11
S
D200
S1
-30
S2
Network 3
>
Q
M50
678493
S1
C200
S2
M3

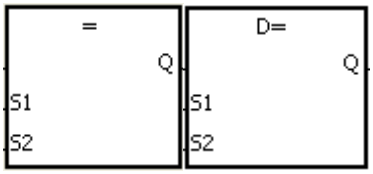
API	Instruction code			Operand	Function
0006-0011	D	AND※		S_1 · S_2	Comparing values AND※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking AND= and DAND= for example

S_1 : Data source 1

S_2 : Data source 2

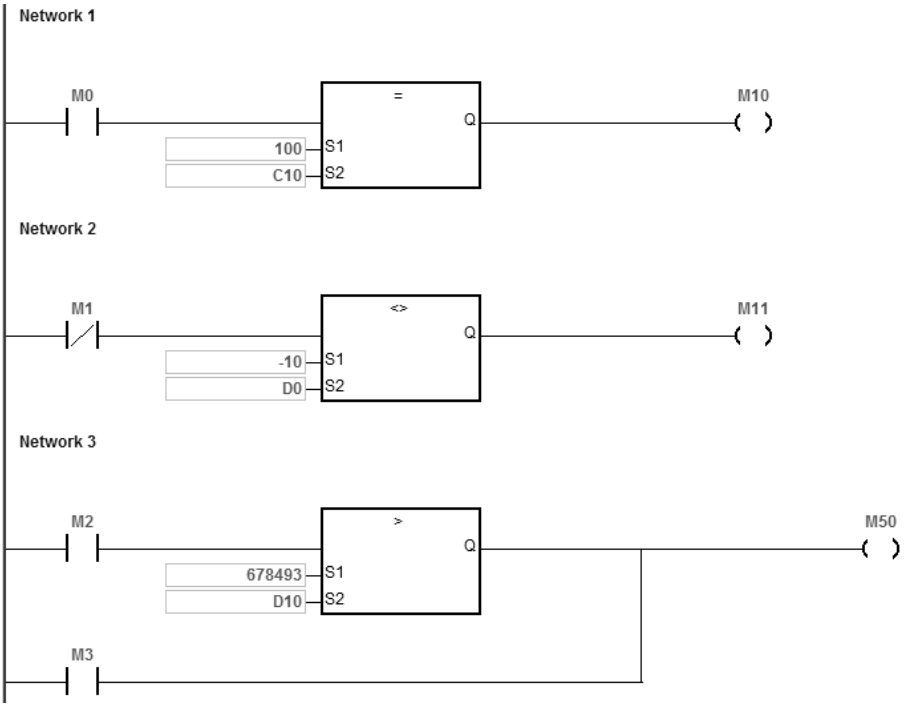
Explanation

1. These instructions compare the values in S_1 and S_2. Take the AND= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.
2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0006	AND =	**D**AND =	$S_1 = S_2$	$S_1 \neq S_2$
0007	AND < >	**D**AND < >	$S_1 \neq S_2$	$S_1 = S_2$
0008	AND >	**D**AND >	$S_1 > S_2$	$S_1 \leqq S_2$
0009	AND > =	**D**AND > =	$S_1 \geqq S_2$	$S_1 < S_2$
0010	AND <	**D**AND <	$S_1 < S_2$	$S_1 \geqq S_2$
0011	AND < =	**D**AND < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

1. When M0 is ON and the current value in C10 is equal to 100, M10 is ON.
2. When M1 is OFF and the value in D0 is not equal to -10, M11 stays ON.
3. When M2 is ON and the value in (D11, D10) is less than 678,493, or when M3 is ON, M50 is ON.


Network 1
M0
=
Q
M10
100
S1
C10
S2
Network 2
M1
<>
Q
M11
-10
S1
D0
S2
Network 3
M2
>
Q
M50
678493
S1
D10
S2
M3

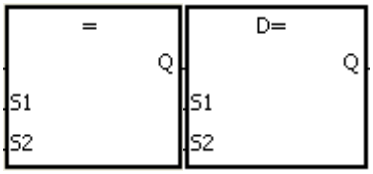
API	Instruction code			Operand	Function
0012-0017	D	OR※		S_1 · S_2	Comparing values OR※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking OR= and DOR= for example

S_1 : Data source 1

S_2 : Data source 2

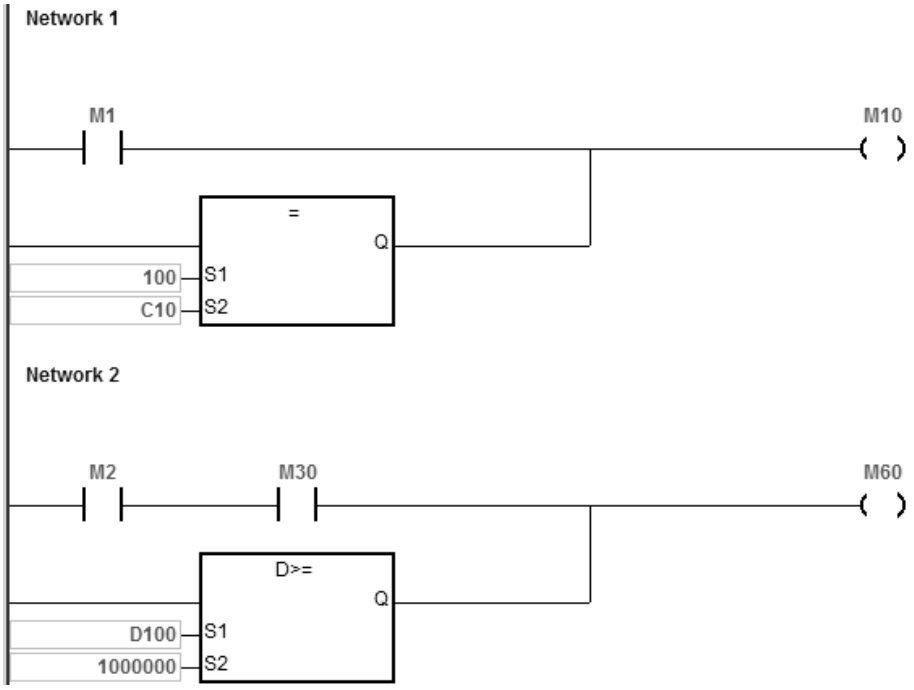
Explanation

1. These instructions compare the values in S_1 and S_2. Take the OR= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.
2. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0012	OR =	**D**OR =	$S_1 = S_2$	$S_1 \neq S_2$
0013	OR < >	**D**OR < >	$S_1 \neq S_2$	$S_1 = S_2$
0014	OR >	**D**OR >	$S_1 > S_2$	$S_1 \leqq S_2$
0015	OR > =	**D**OR > =	$S_1 \geqq S_2$	$S_1 < S_2$
0016	OR <	**D**OR <	$S_1 < S_2$	$S_1 \geqq S_2$
0017	OR < =	**D**OR < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

1. When M1 is ON, or when the current value in C10 is equal to 100, M10 is ON.
2. When both M2 and M30 are ON, or when the value in (D101, D100) is greater than or equal to 1000,000, M60 is ON.


Network 1
M1
M10
=
Q
100
S1
C10
S2
Network 2
M2
M30
M60
D>=
Q
D100
S1
1000000
S2

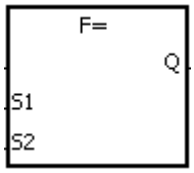
API	Instruction code	Operand	Function
0018-0023	FLD※	S_1 · S_2	Comparing floating-point numbers LD ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



Taking FLD= for example

S_1 : Data source 1

S_2 : Data source 2

Explanation

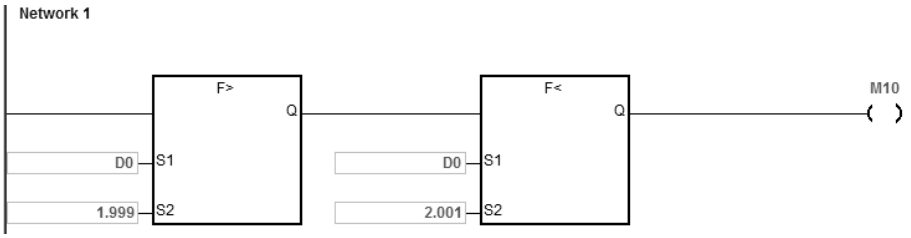
1. These instructions compare the 32-bit single precision floating point number

2. Compare values in S_1 and S_2. Take the FLD= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met. Refer to Additional Remarks point 2 below for more information on the instruction of FLD=.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0018	FLD =	$S_1 = S_2$	$S_1 \neq S_2$
0019	FLD < >	$S_1 \neq S_2$	$S_1 = S_2$
0020	FLD >	$S_1 > S_2$	$S_1 \leqq S_2$
0021	FLD > =	$S_1 \geqq S_2$	$S_1 < S_2$
0022	FLD <	$S_1 < S_2$	$S_1 \geqq S_2$
0023	FLD < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

Take the FLD = instruction for example. When the value in D0 is larger than 1.999 and is smaller than 2.001, M10 is ON.



Additional remarks

1. If the value in S_1 or S_2 exceeds the range of values that can be represented by floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.
2. If the floating-point value in S_1 or S_2 is from external input, or from ISPSoft, chances are a minor value difference may occur and the result of the “FLD=“ instruciton may not be dependable. Use instructions “FLD<” and” FAND>” or “FLD>” and “FAND<” instead when the floating-point value in S_1 or S_2 is from external input, or from ISPSoft.

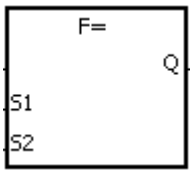
API	Instruction code	Operand	Function
0024-0029	FAND※	S_1 · S_2	Comparing floating-point numbers AND※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



Taking FAND= and DFAND= for example

S_1 : Data source 1

S_2 : Data source 2

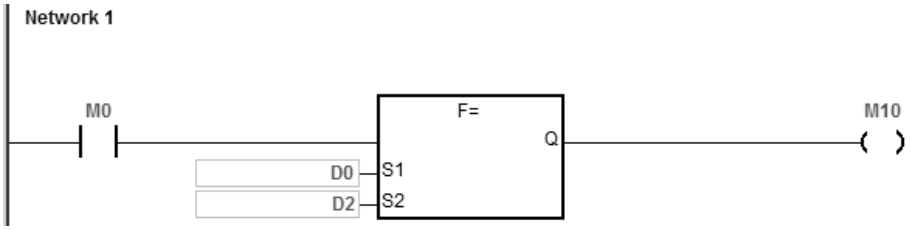
Explanation

1. These instructions compare the 32-bit single precision floating point numbers
2. Compare values in S_1 and S_2. Take the FAND= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0024	FAND =	$S_1 = S_2$	$S_1 \neq S_2$
0025	FAND < >	$S_1 \neq S_2$	$S_1 = S_2$
0026	FAND >	$S_1 > S_2$	$S_1 \leqq S_2$
0027	FAND > =	$S_1 \geqq S_2$	$S_1 < S_2$
0028	FAND <	$S_1 < S_2$	$S_1 \geqq S_2$
0029	FAND < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

Take the instruction FAND = for example. When M0 is ON and the value in D0 is equal to that in D2, M10 is ON.



Additional remarks

1. If the value in S_1 or S_2 exceeds the range of values that can be represented by floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

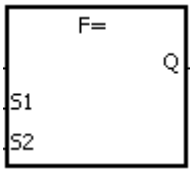
API	Instruction code	Operand	Function
0030-0035	FOR※	S_1 · S_2	Comparing floating-point numbers OR※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				

Pulse Instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Data source1

S_2 : Data source2

Taking FOR= and DFOR= for example

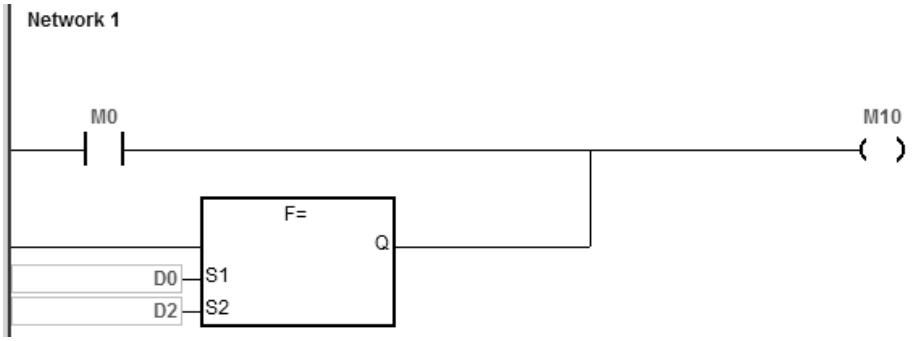
Explanation

1. These instructions compare the 32-bit single precision floating point numbers.
2. Compare values in S_1 and S_2. Take the FOR= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

API number	32-bit instruction	Continuity condition	Discontinuity condition
0030	FOR =	$S_1 = S_2$	$S_1 \neq S_2$
0031	FOR < >	$S_1 \neq S_2$	$S_1 = S_2$
0032	FOR >	$S_1 > S_2$	$S_1 \leqq S_2$
0033	FOR > =	$S_1 \geqq S_2$	$S_1 < S_2$
0034	FOR <	$S_1 < S_2$	$S_1 \geqq S_2$
0035	FOR < =	$S_1 \leqq S_2$	$S_1 > S_2$

Example

When M0 is ON, or when the value in D0 is equal to that in D2, M10 is ON.



Additional remarks

1. If the value in S_1 or S_2 exceeds the range of values that can be represented by floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

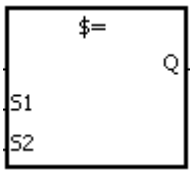
API	Instruction code	Operand	Function
0036-0037	LD$※	S_1 · S_2	Comparing strings LD$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Taking LD$= for example

S_1 : Data source1

S_2 : Data source2

Explanation

1. These instructions compare the strings in S_1 and S_2.
2. S_1 and S_2 can contain strings up to 256 characters(16#00 the end symbol is included).
3. Take the instruction LD$= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

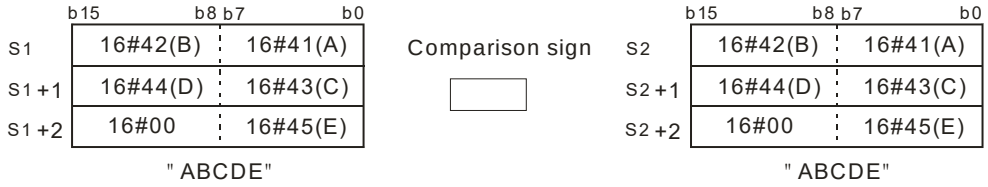
API number	16-bit instruction	Continuity condition	Discontinuity condition
0036	LD$ =	$S_1 = S_2$	$S_1 \neq S_2$
0037	LD$ < >	$S_1 \neq S_2$	$S_1 = S_2$

4. Only when the data in **S**–**S**+n (n indicates the n^{th} device, up to 256 characters in each string) includes 16#00 can the data be compared complete strings. For example:

	b15 ~ b8	b7 ~ b0
S	16#32(2)	16#31(1)
S+1	16#34(4)	16#33(3)
S+2	16#00	16#35(5)

"12345"

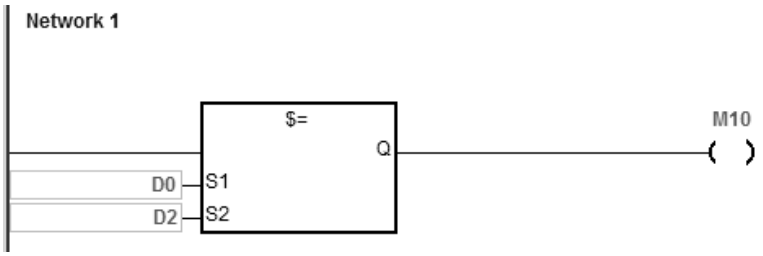
5. When the two strings are the same, the corresponding comparison results of the instructions are listed below. For example:



Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example

When the string starting with the data in D0-16#00 is equal to the string staring with D2-16#00, M10 is ON.



Additional remarks

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction is not executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, the string ends when the end symbol 16#00 is found. The symbol 16#00 determines the length of the string.

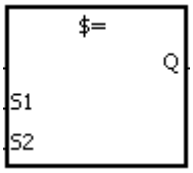
API	Instruction code	Operand	Function
0042-0043	AND$※	S_1 · S_2	Comparing strings AND$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Taking AND$= for example

S_1 : Data source1

S_2 : Data source2

Explanation

1. These instructions compare the strings in S_1 and S_2.
2. S_1 and S_2 can contain string up to 256 characters (16#00 the end symbol is included).
3. Take the AND$= instruction for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

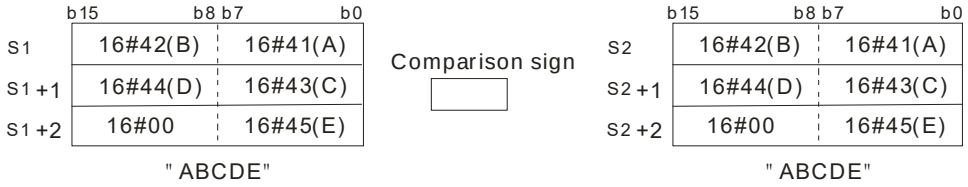
API number	16-bit instruction	Continuity condition	Discontinuity condition
0042	AND$ =	$S_1 = S_2$	$S_1 \neq S_2$
0043	AND$ < >	$S_1 \neq S_2$	$S_1 = S_2$

Only when the data in **S**–**S**+n (n indicates the n^{th} device, up to 256 characters in each string) includes 16#00 can the instruction compare the data as complete strings. For example:

	b15 – b8	b7 – b0
S	16#32(2)	16#31(1)
S+1	16#34(4)	16#33(3)
S+2	16#00	16#35(5)

"12345"

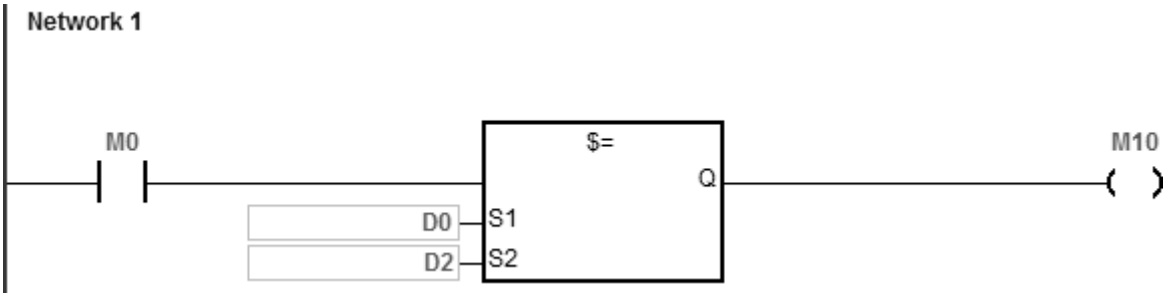
4. When two strings are the same, the corresponding comparison operation results of the instructions are listed below. For example:



Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example

When the string starting with the data in D0-16#00 is equal to the string staring with D2-16#00, M10 is ON.



Additional remarks

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction is not executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, the string ends when the end symbol 16#00 is found. The symbol 16#00 determines the length of the string.

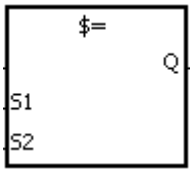
API	Instruction code	Operand	Function
0048-0049	OR$※	S_1 · S_2	Comparing strings OR$※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Taking OR$= for example

S_1 : Data source1

S_2 : Data source2

Explanation

1. These instructions compare the strings in S_1 and S_2.
2. S_1 and S_2 can contain up to 256 characters (16#00 the end symbol is included).
3. Take the instruction OR$= for example. When the comparison result is that the value in S_1 is equal to that in S_2, the continuity condition of the instruction is met. When the comparison result is that the value in S_1 is not equal to that in S_2, the discontinuity condition of the instruction is met.

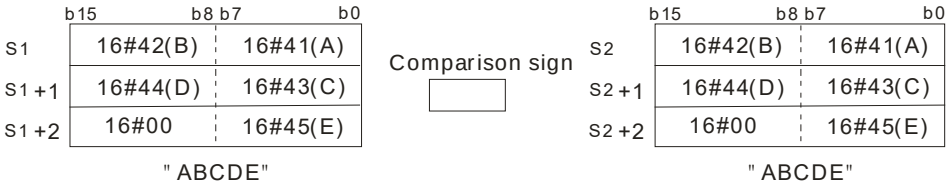
API number	16-bit instruction	Continuity condition	Discontinuity condition
0048	OR$ =	$S_1 = S_2$	$S_1 \neq S_2$
0049	OR$ < >	$S_1 \neq S_2$	$S_1 = S_2$

Only when the data in **S**–**S**+n (n indicates the n^{th} device, up to 256 characters in each string) includes 16#00 can the instruction compare the data as complete strings. For example:

	b15 b8	b7 b0
S	16#32(2)	16#31(1)
S+1	16#34(4)	16#33(3)
S+2	16#00	16#35(5)

"12345"

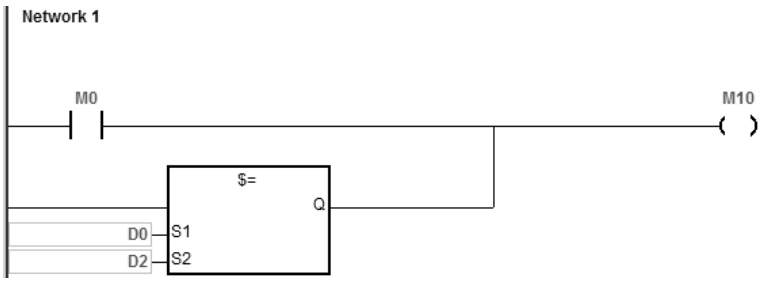
4. When two strings are the same, the corresponding comparison operation results of the instructions are listed below. For example:



Comparison symbol	Comparison operation result
$ =	Continuity
$ < >	Discontinuity

Example

When the string starting with the data in D0-16#00 is equal to the string staring with D2-16#00, M10 is ON.



Additional remarks

1. If the string contains more than 256 characters or the string does not end with 16#00, the instruction is not executed, SM is ON, and the error code in SR0 is 16#200E.
2. During the string comparison, the string ends when the end symbol 16#00 is found. The symbol 16#00 determines the length of the string.

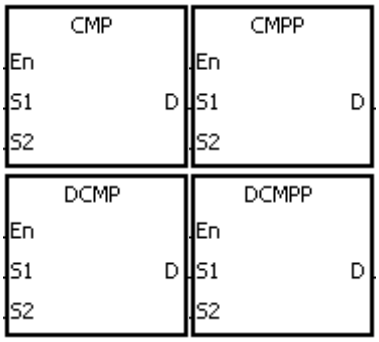
API	Instruction code			Operand	Function
0054	D	CMP	P	$S_1 \cdot S_2 \cdot D$	Comparing values

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S1 : Comparison value1

S2 : Comparison value2

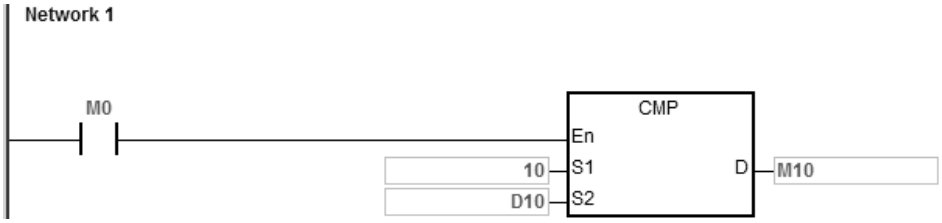
D : Comparison result

Explanation

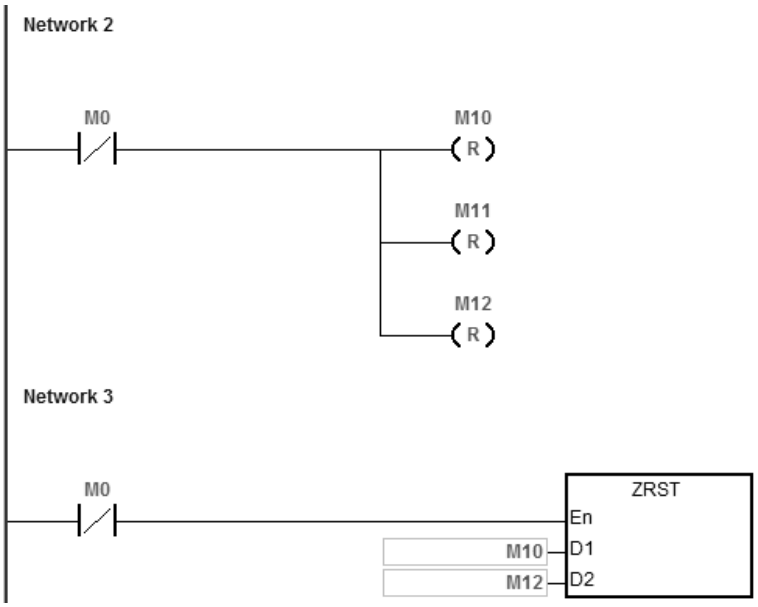
1. These instructions compare the single decimal numbers in S_1 and S_2 and store the comparison results in **D**.
2. The operand **D** occupies 3 consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the value in S_1 is greater than the value in S_2, **D** is ON. If the value in S_1 is equal to the value in S_2, **D**+1 is ON. If the value in S_1 is less than the value in S_2, **D**+2 is ON.
3. Only the DCMP and DCMPP instructions can use the 32-bit counter, but not the device E.

Example

1. If the operand **D** is M0, the comparison results are stored in M0, M1 and M2, as shown below.
2. When M0 is ON, the CMP instruction is executed. M10, M11, or M12 is ON. When M0 is OFF, the execution of the CMP instruction stops and the state of M10, the state of M11, and the state of M12 remain unchanged.



3. If you need to clear the comparison result, use the RST or ZRST instruction.



Additional remarks

1. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.
2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

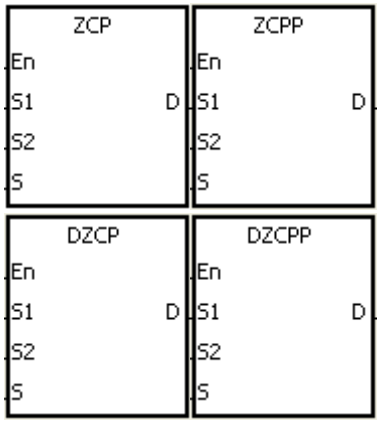
API	Instruction code			Operand	Function
0055	D	ZCP	P	S_1 · S_2 · **S** · **D**	Zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S					●	●	●	●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S		●	●		●	●	●				●	●	
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S1 : Minimum value of the zone comparison

S2 : Maximum value of the zone comparison

S : Comparison value

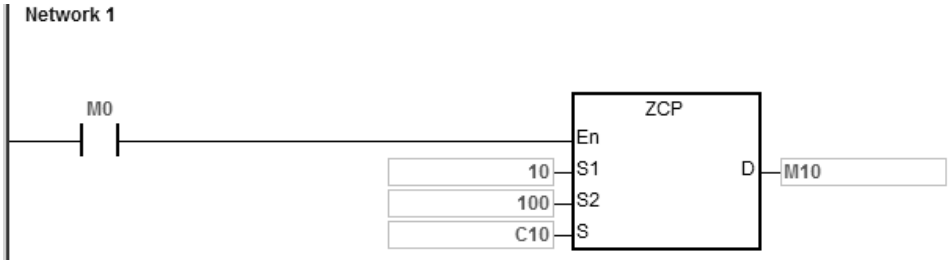
D : Comparison result

Explanation

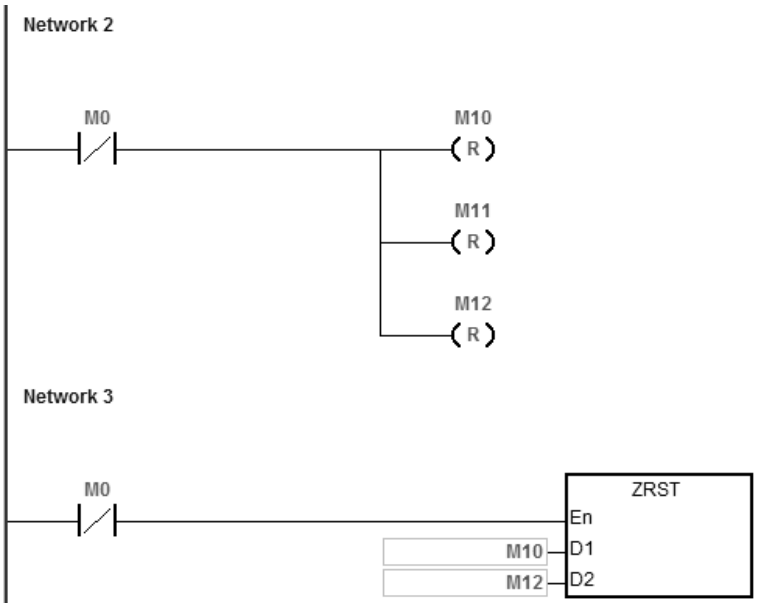
1. These instructions compare signed decimal numbers in **S** and S_1, and compare the signed decimal numbers in **S** and S_2, and stores the comparison results in **D**.
2. The value in S_1 must be less than that in S_2. If the value in S_1 is larger than that in S_2, S_1 is the maximum/minimum value during the execution of the ZCP instruction.
3. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2.
   If the value in S_1 is less than the value in **S**, **D** is ON.
   If the value in **S** is between the values in S_1 and S_2, **D**+1 is ON.
   If the value in **S** is greater than the value in S_2, **D**+2 is ON.
4. Only the DZCP and DZCPP instructions can use the 32-bit counter, but not the device E.

Example

1. If the operand **D** is M0, the comparison results are stored in M0, M1 and M2, as shown below.
2. When M0 is ON, the ZCP instruction is executed. M10, M11, or M12 is ON. When M0 is OFF, the ZCP instruction is not executed, and the state of M10, the state of M11, and the state of M12 remain unchanged.



3. If you need to clear the comparison result, use the RST or ZRST instruction.



Additional remarks

1. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.
2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

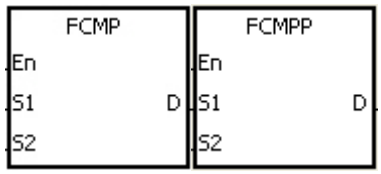
API	Instruction code			Operand	Function
0056		FCMP	P	S_1 · S_2 · D	Comparing floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Floating-point comparison value1

S_2 : Floating-point comparison value2

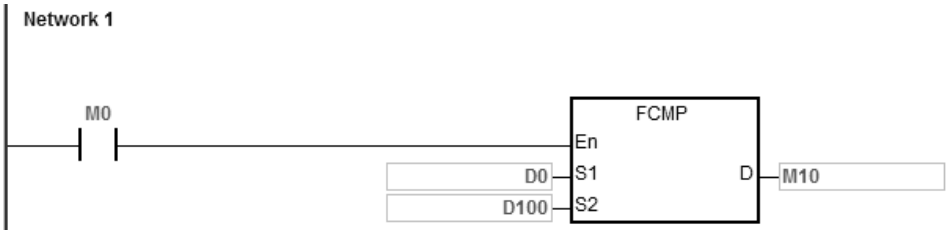
D : Comparison result

Explanation

1. This instruction compares the floating-point numbers in **S_1** and **S_2**, and stores the comparison results (> · = · <) in **D**.
2. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the value in **S_1** is greater than the value in **S_1**, **D** is ON. If the value in **S_1** is equal to the value in **S_2**, **D**+1 is ON. If the value in **S_1** is less than the value in **S_2**, **D**+2 is ON.

Example

1. If the operand **D** is M10, the comparison results is stored in M10, M11 and M12, as shown below.
2. When M0 is ON, the FCMP instruction is executed. M10, M11, or M12 is ON. When M0 is OFF, the FCMP instruction is not executed and the state of M10, the state of M11, and the state of M12 remain unchanged.
3. If you want to get the comparison result ≧, ≦, or ≠, connect M10–M12 is series or in parallel.
4. If you want to clear the comparison result, use the RST or ZRST instruction.



Additional remarks

1. If the value in S_1 or S_2 exceeds the range of values that can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.
2. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.
3. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

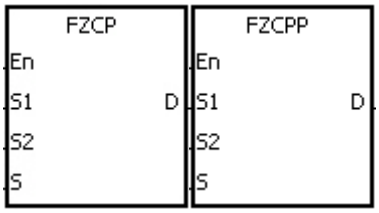
API	Instruction code			Operand	Function
0057		FZCP	P	**S_1 · S_2 · S · D**	Floating-point zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○
S					●	●	●	●	●		○					○
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
S									●				
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Minimum value of the zone comparison

S_2 : Maximum value of the zone comparison

S : Comparison value

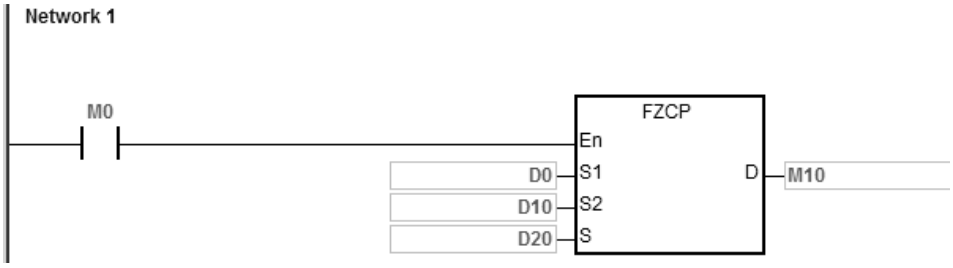
D : Comparison result

Explanation

1. This instruction compares the floating-point numbers in **S** and **S_1**, and compare the floating-point numbers in **S** with that in **S_2**, and then stores the results in **D**.
2. The value in **S_1** must be less than that in **S_2**. If the value in **S_1** is larger than that in **S_2**, **S_1** is the maximum/minimum value during the execution of the FZCP instruction.
3. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D**+1, and **D**+2. If the value in **S_1** is greater than the value in **S**, **D** is ON. If the value in **S** is between the value in **S_1** and the value in **S_2**, **D**+1 is ON. If the compared value in **S_2** is less than the value in **S**, **D**+2 is ON.

Example

1. If the operand **D** is M0, the comparison results are stored in M0, M1 and M2.
2. When M0 is ON, the FZCP instruction is executed. M10, M11, or M12 is ON. When M0 is OFF, the FZCP instruction is not executed, and the state of M10, the state of M11, and the state of M12 remain unchanged.
3. If you want to clear the comparison result, use the RST or ZRST instruction.



Additional remarks

1. If the value in S_1 or S_2 or **S** exceeds the range of values that can be represented by the floating-point numbers, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.
2. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.
3. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

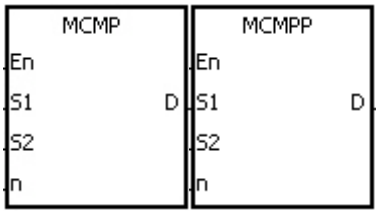
API	Instruction code			Operand	Function
0058		MCMP	P	S_1 · S_2 · **n** · **D**	Matrix comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○					
S_2					●	●		●	●		○					
n					●	●		●	●		○		○	○		
D					●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Matrix source device 1

S_2 : Matrix source device 2

n : Length of the array

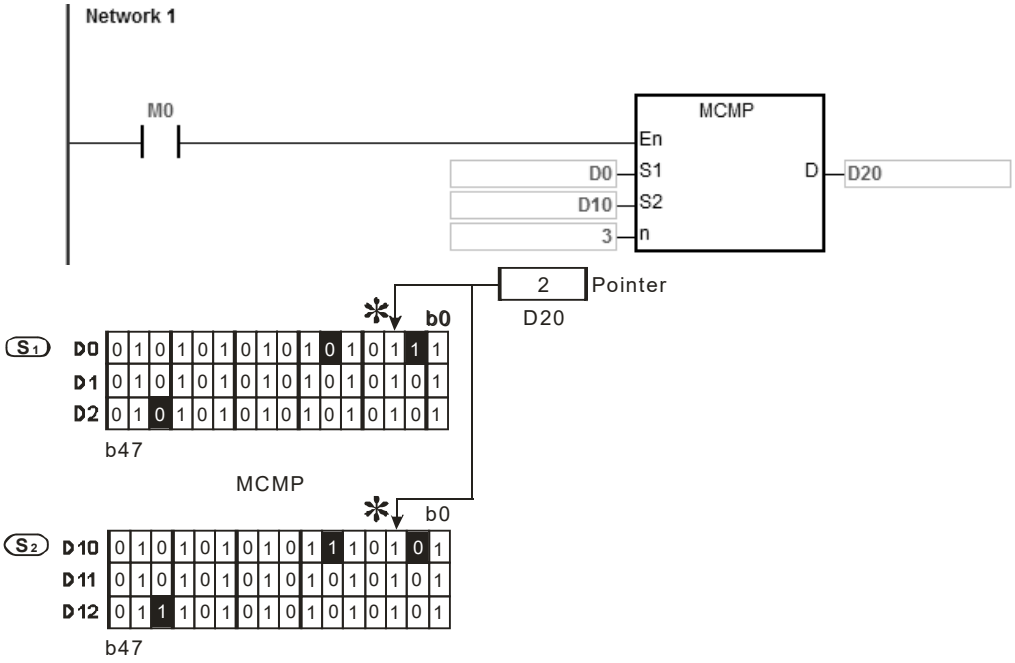
D : Pointer

Explanation

1. This instruction searches for the bits with different states, starting from the bits specified by adding one to the current value in **D**. After finding the bits with different states, the instruction stores the bit number in **D**, and the comparison is finished.
2. The operand **n** must be between 1–256.
3. When SM607 is ON, the equivalent values are compared. When SM607 is OFF, the different values are compared. When the matching bits are found, the comparison stops immediately, and SM610 is ON. When the last bits are compared, SM608 is ON, and the bit number is stored in **D**. The comparison starts from the 0^{th} bits in the next scan cycle, and SM609 is ON. When the value in **D** exceeds the range, SM611 is ON.
4. When the MCMP instruction is executed, you need a 16-bit register to specify a certain bit among the 16**n** bits in the matrix for the operation. The register is called the pointer, and is specified by you. The value in the register is between 0–16**n**-1, and corresponds to the bit between b0 to b16**n**-1. During the operation, you are prevented from altering the value of the pointer in case the search for the matching bits is affected. If the value of the pointer exceeds the range, SM611 is ON, and the MCMP instruction is not executed.
5. If SM608 and SM610 occur simultaneously, they are ON simultaneously.

Example

1. When M0 is switched from OFF to ON, SM609 is OFF. The search for the bits with different states (SM607 is OFF) starts from the bits specified by the adding one to the current value of the pointer.
2. Suppose the current value in D20 is 2. When M0 is switched from OFF to ON four times, you get the following execution results.
   - The value in D20 is 5, SM610 is ON, and SM608 is OFF.
   - The value in D20 is 45, SM610 is ON, and SM608 is OFF.
   - The value in D20 is 47, SM610 is OFF, and SM608 is ON.
   - The value in D20 is 1, SM610 is ON, and SM608 is OFF.



Additional remarks

1. Operation error codes: If the devices **S_1**+n-1 and **S_2**+n-1 exceed the range, the MCMP instruction is not executed, SM is ON, and the error code in SR0 is 16#2003. If the value in the operand **n** is not between 1 and 256, the MCMP instruction is not executed, SM is ON, and the error code in SR0 is 16#200B.
2. Operation flags:

	Matrix comparison flag.
SM607:	ON: comparing equivalent values
	OFF: comparing different values
SM608:	The matrix comparison ends. When the last bits are compared, SM608 is ON.
SM609:	ON: the comparison starts from bit 0.
SM610:	Matrix bit search flag. When the matching bits are found, the comparison stops immediately, and SM610 is ON.
SM611:	Matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

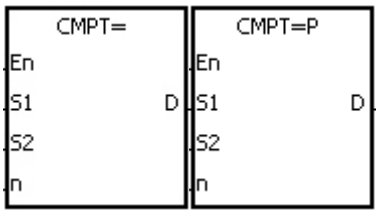
API	Instruction code			Operand	Function
0059-0064		CMPT※	P	S_1 · S_2 · n · D	Comparing tables

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○		○	○		
S_2					●	●		●	●		○					
n					●	●		●	●		○	○	○	○		
D		●	●	●				●		●						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Source device 1

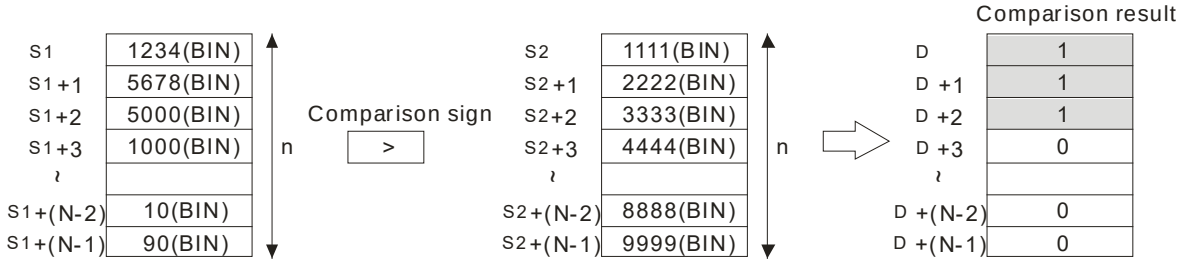
S_2 : Source device 2

n : Data length

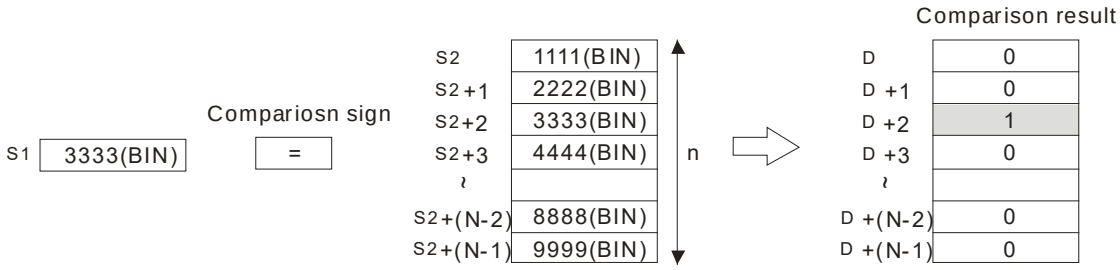
D : Comparison result

Explanation

1. This instruction compares **n** signed decimal numbers in devices starting at **S_1** with those in devices starting at **S_2**, and then stores the comparison results in **D**.
2. The operand **n** must be between 1–256.
3. The value that is written into the operand **D** is a one-bit value.
4. When the results of the comparison using the CMPT# instruction are that all devices are ON, SM620 is ON. Otherwise, SM620 is OFF.
5. If the operand **S_1** is a device, the comparison is as shown below.



6. If the operand **S_1** is a constant between -32768 to 32767, the comparison is as shown below.

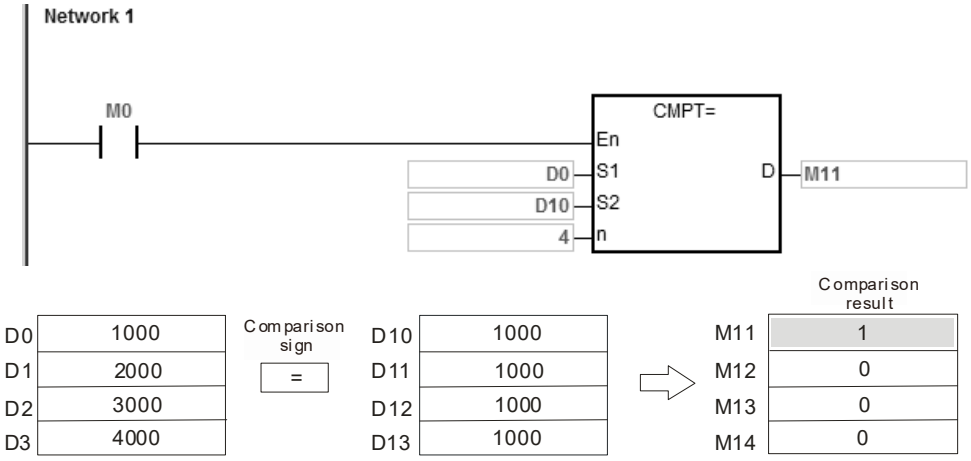


7. The corresponding comparison operation results of the instructions are listed in the following table.

API number	16-bit instruction	Comparison operation result	
		ON	OFF
0059	CMPT =	**$S_1 = S_2$**	**$S_1 \neq S_2$**
0060	CMPT < >	**$S_1 \neq S_2$**	**$S_1 = S_2$**
0061	CMPT >	**$S_1 > S_2$**	**$S_1 \leqq S_2$**
0062	CMPT > =	**$S_1 \geqq S_2$**	**$S_1 < S_2$**
0063	CMPT <	**$S_1 < S_2$**	**$S_1 \geqq S_2$**
0064	CMPT < =	**$S_1 \leqq S_2$**	**$S_1 > S_2$**

Example

The data in D0–D3 are compared with that in D10–D13. If the comparison result is that the data in D0–D3 are the same as that in D10–D13, M11–M14 are ON.



Additional remarks

1. If the value in the operand **n** is not between 1–256, the instruction is not executed, SM is ON, and the error code in SR0 is 16#200B.
2. If the number of devices specified by **S_1–S_1+n**, **S_2–S_2+n**, or **D** is insufficient, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

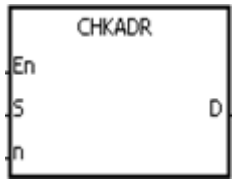
API	Instruction code	Operand	Function
0065	CHKADR	**S** · **n** · **D**	Checking the addresses in a pointer register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S																
n					●	●		●	●		○	○	○	○		
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													
n		●			●	●							
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : Pointer register

n : Number of devices

D : Check result

Explanation

1. This instruction checks whether the value in **S** and (the value in **S**)+**n**-1 exceed the device range. If the result is that the value in **S** and (the value in **S**)+**n**-1 do not exceed the device range, the device **D** is ON. Otherwise, it is OFF.
2. **S** supports the pointer registers D, T, C, HC (POINTER/T_POINTER/C_POINTER/HC_POINTER).
3. The operand **n** must be between 1–1024.
4. You can use the CHKADR instruction only in a function block. Use CHKADR during the initial program development phase or when you are not sure if the device range will be exceeded. After the program is written and debugged, you can delete this instruction.

Example

1. Create a program (Prog0) and a function block (FB0) in ISPSoft.

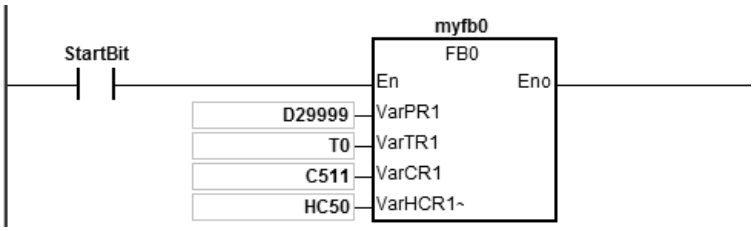
Declare two variables in the program.

Local Symbols				
Declaration Type	Identifiers	Address	Type...	Initial Value
VAR	myfb0	N/A [Auto]	FB0	N/A
VAR	StartBit	N/A [Auto]	BOOL	FALSE

2. Declare VarPR1, VarTR1, VarCR1, and VarHCR1 in the function block, and assign the data types POINTER, T_POINTER, C_POINTER, and HC_POINTER to them respectively.

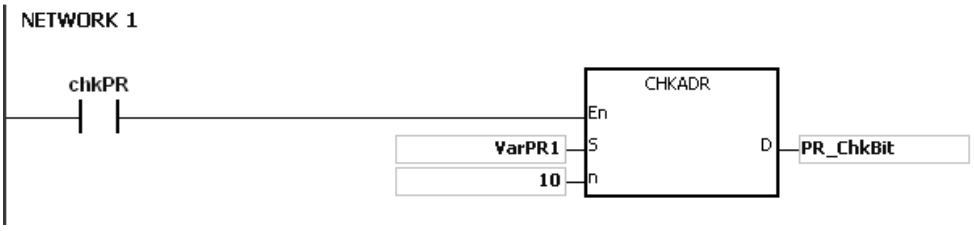
Local Symbols				
Declaration Type	Identifiers	Address	Type...	Initial Value
VAR_IN_OUT	VarPR1	N/A [Auto]	POINTER	N/A
VAR_IN_OUT	VarTR1	N/A [Auto]	T_POINTER	N/A
VAR_IN_OUT	VarCR1	N/A [Auto]	C_POINTER	N/A
VAR_IN_OUT	VarHCR1	N/A [Auto]	HC_POINTER	N/A
VAR	PR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	TR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	CR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	HCR_ChkBit	N/A [Auto]	BOOL	FALSE
VAR	chkPR	N/A [Auto]	BOOL	N/A
VAR	chkTR	N/A [Auto]	BOOL	N/A
VAR	chkCR	N/A [Auto]	BOOL	N/A
VAR	chkHCR	N/A [Auto]	BOOL	N/A

3. Call the function block FB0 in the program, and assign D29999, T0, C511, and HC50 to VarPR1, VarTR1, VarCR1, and VarHCR1 in FB0 respectively.

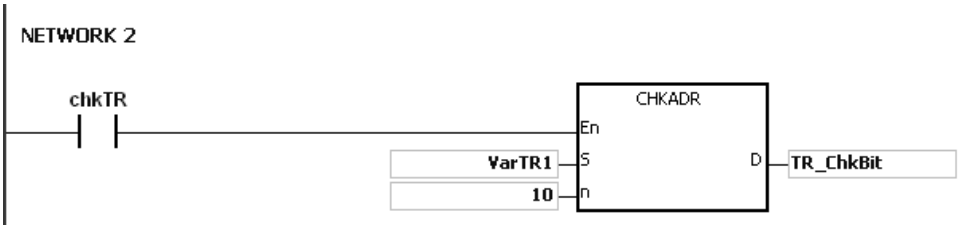


4. Use the CHKADR instruction to check whether VarPR1, VarTR1, VarCR1, and VarHCR1 exceed the range.

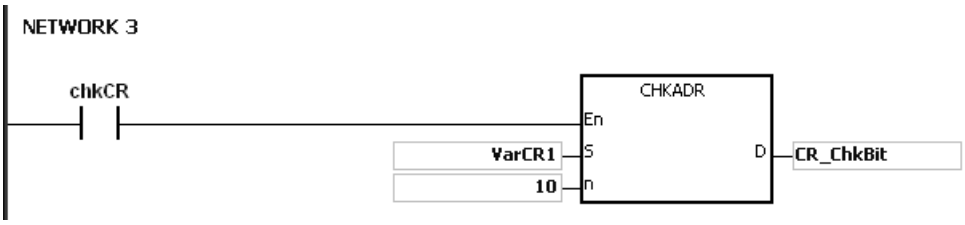
5. When chkPR is ON, the actual device represented by VarPR1 is D29999. Since the legal range of devices is from D0 to D29999, and D29999+10-1=D30008 which exceeds the range, PR_ChkBit is OFF.



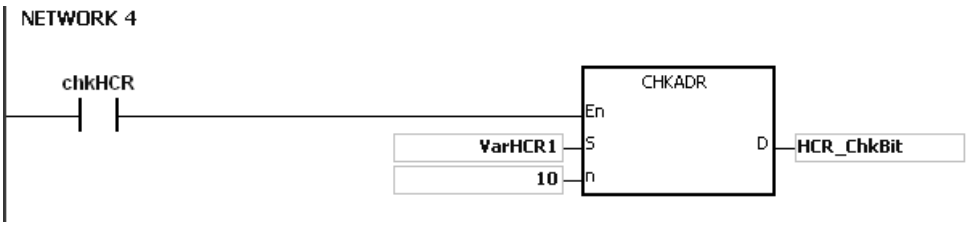
6. When chkTR is ON, the actual device represented by VarTR1 is T0. Since the legal range of devices is from T0 to T511, and T0+10-1=T9 which does not exceed the range, TR_ChkBit is ON.



7. When chkCR is ON, the actual device represented by C511. Since the legal range of devices is from C0 to C511, and C511+10-1=C520 which exceeds the range, CR_ChkBit is OFF.



8. When chkHCR is ON, the actual device represented by HC50 is VarHCR1. Since the legal range of deices is from HC0 to HC255, and HC50+10-1=HC59 which does not exceed the range, HCR_ChkBit is ON.



Additional remarks

1. If the value (the actual device address) in **S** exceeds the device range, the CHKADR instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.
2. If the value in the operand **n** is not between 1–1024, the CHKADR instruction is not executed, SM is ON, and the error code in SR0 is 16#200B.

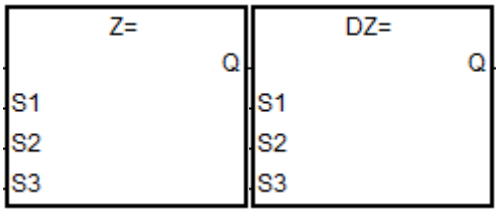
API	Instruction code			Operand	Function
0066- 0071	D	LDZ※		S_1 · S_2 · S_3	Comparing contact type absolute values LDZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



S_1 : Data source1

S_2 : Data source2

S_3 : Comparison result

Taking LDZ= and DLDZ= for example

6

Explanation

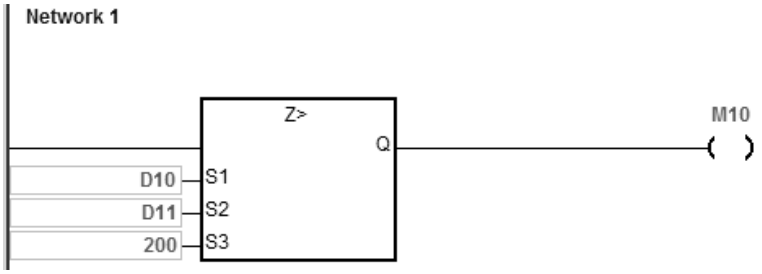
1. These instructions compare the absolute value of the difference between S_1 and S_2 with the absolute value of S_3. Take the LDZ= instruction for example. If the comparison result is that the absolute value of the difference between S_1 and S_2 is equal to the absolute value of S_3, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between S_1 and S_2 is not equal to the absolute value of S_3, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0066	LDZ =	**D**LDZ =	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0067	LDZ < >	**D**LDZ < >	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0068	LDZ >	**D**LDZ >	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0069	LDZ > =	**D**LDZ > =	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0070	LDZ <	**D**LDZ <	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0071	LDZ < =	**D**LDZ < =	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

Example

1. When the absolute difference of D10 and D11 is greater than 200, M10 is ON. While the absolute difference is less than 200, M10 is OFF.



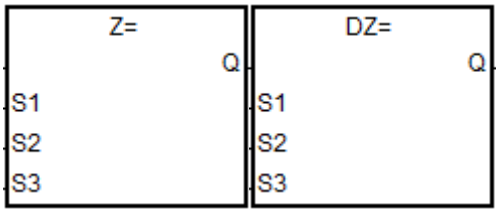
API	Instruction code			Operand	Function
0072-0077	D	ANDZ※		**S_1 · S_2 · S_3**	Comparing contact type absolute values ANDZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



S_1 : Data source1

S_2 : Data source2

S_3 : Comparison result

Taking ANDZ= and DANDZ= for example

6

Explanation

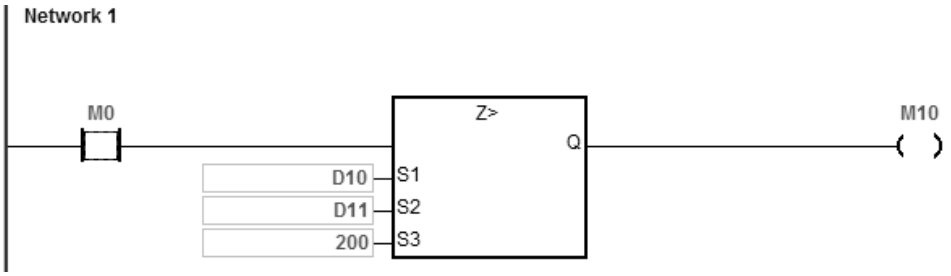
1. These instructions compare the absolute value of the difference between **S_1** and **S_2** with the absolute value of **S_3**. Take the ANDZ= instruction for example. If the comparison result is that the absolute value of the difference between **S_1** and **S_2** is equal to the absolute value of **S_3**, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between **S_1** and **S_2** is not equal to the absolute value of **S_3**, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0072	ANDZ =	**D**ANDZ =	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$
0073	ANDZ < >	**D**ANDZ < >	$\lvert S_1 - S_2 \rvert \neq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert = \lvert S_3 \rvert$
0074	ANDZ >	**D**ANDZ >	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$
0075	ANDZ > =	**D**ANDZ > =	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$
0076	ANDZ <	**D**ANDZ <	$\lvert S_1 - S_2 \rvert < \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert \geqq \lvert S_3 \rvert$
0077	ANDZ < =	**D**ANDZ < =	$\lvert S_1 - S_2 \rvert \leqq \lvert S_3 \rvert$	$\lvert S_1 - S_2 \rvert > \lvert S_3 \rvert$

Example

1. When M0 is ON and the absolute difference of D10 and D11 is greater than 200, M10 is ON. While the absolute difference is less than 200, M10 is OFF.



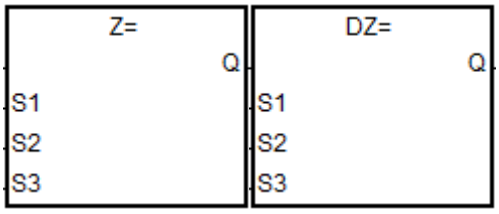
API	Instruction code			Operand	Function
0078-0083	D	ORZ※		S_1 · S_2 · S_3	Comparing contact type absolute values ORZ※

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
S_3		●	●		●	●	●				●	●	

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking ORZ= and DORZ= for example

S_1 : Data source1

S_2 : Data source2

S_3 : Comparison result

6

Explanation

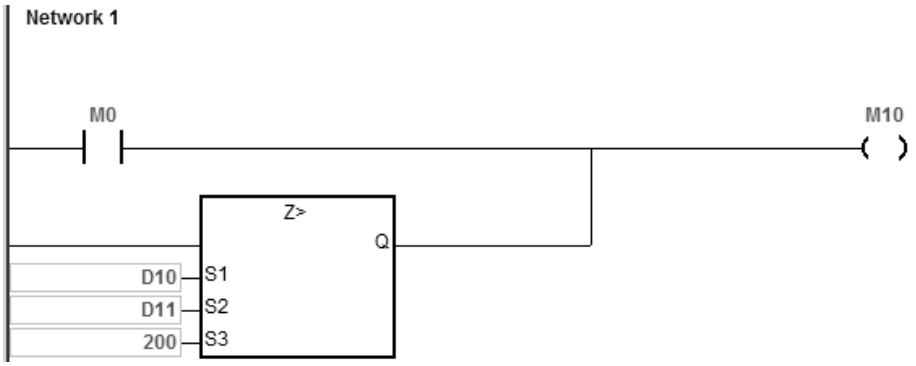
1. These instructions compare the absolute value of the difference between S_1 and S_2 with the absolute value of S_3. Take the ORZ= instruction for example. If the comparison result is that the absolute value of the difference between S_1 and S_2 is equal to the absolute value of S_3, the continuity condition of the instruction is met. If the comparison result is that the absolute value of the difference between S_1 and S_2 is not equal to the absolute value of S_3, the discontinuity condition of the instruction is met.

2. Only the 32-bit instruction can use the 32-bit HC device, but not the device E.

API number	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
0078	ORZ =	DORZ =	$\|S_1-S_2\|=\|S_3\|$	$\|S_1-S_2\|\neq\|S_3\|$
0079	ORZ < >	DORZ < >	$\|S_1-S_2\|\neq\|S_3\|$	$\|S_1-S_2\|=\|S_3\|$
0080	ORZ >	DORZ >	$\|S_1-S_2\|>\|S_3\|$	$\|S_1-S_2\|\leqq\|S_3\|$
0081	ORZ > =	DORZ > =	$\|S_1-S_2\|\geqq\|S_3\|$	$\|S_1-S_2\|<\|S_3\|$
0082	ORZ <	DORZ <	$\|S_1-S_2\|<\|S_3\|$	$\|S_1-S_2\|\geqq\|S_3\|$
0083	ORZ < =	DORZ < =	$\|S_1-S_2\|\leqq\|S_3\|$	$\|S_1-S_2\|>\|S_3\|$

Example

1. When M0 is ON and the absolute difference of D10 and D11 is greater than 200, M10 is ON. While the absolute difference is less than 200, M10 is OFF.



6.2 Arithmetic Instructions

6.2.1 List of Arithmetic Instructions

The following table lists the Arithmetic instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0100	+	D+	✓	Adding binary numbers
0101	-	D-	✓	Subtracting binary numbers
0102	*	D*	✓	Multiplying binary numbers
0103	/	D/	✓	Dividing binary numbers
0104	–	F+	✓	Adding floating-point numbers
0105	–	F-	✓	Subtracting floating-point numbers
0106	–	F*	✓	Multiplying floating-point numbers
0107	–	F/	✓	Dividing floating-point numbers
0112	BK+	DBK+	✓	Adding binary numbers in blocks
0113	BK-	DBK-	✓	Subtracting binary numbers in blocks
0114	$+	–	✓	Linking strings
0115	INC	DINC	✓	Adding one to a binary number
0116	DEC	DDEC	✓	Subtracting one from a binary number
0117	MUL16	MUL32	✓	MUL16: Multiplying binary numbers for 16-bit instructions MUL32: Multiplying binary numbers for 32-bit instructions
0118	DIV16	DIV32	✓	DIV16: Dividing binary numbers for 16-bit instructions DIV32: Dividing binary numbers for 32-bit instructions

6.2.2 Explanation of Arithmetic Instructions

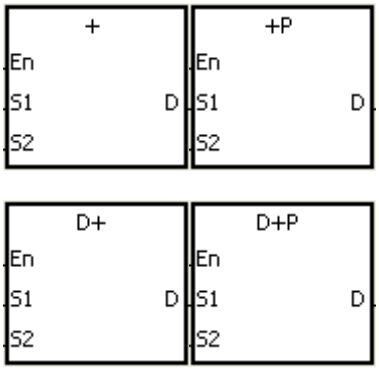
API	Instruction code			Operand	Function
0100	D	+	P	**S_1 · S_2 · D**	Adding binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Augend

S_2 : Addend

D : Sum

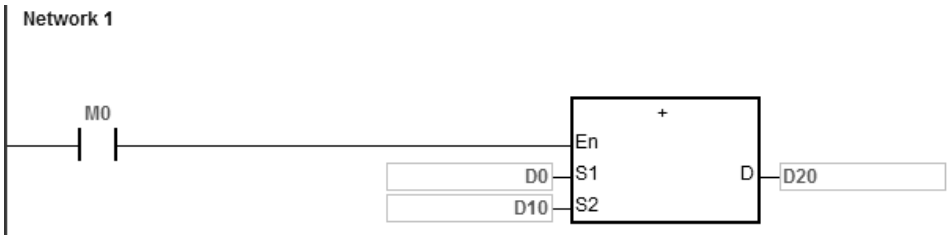
Explanation

1. This instruction adds the binary value in **S_2** to the binary value in **S_1**, and stores the sum in **D**.
2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.
3. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)
4. When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
5. For 16-bit binary values, when the operation result exceeds the range of 16-bit binary values, SM602 is ON. Otherwise, it is OFF.

6. For 32-bit binary values, when the operation result exceeds the range of 32-bit binary values, SM602 is ON. Otherwise, it is OFF.

Example 1

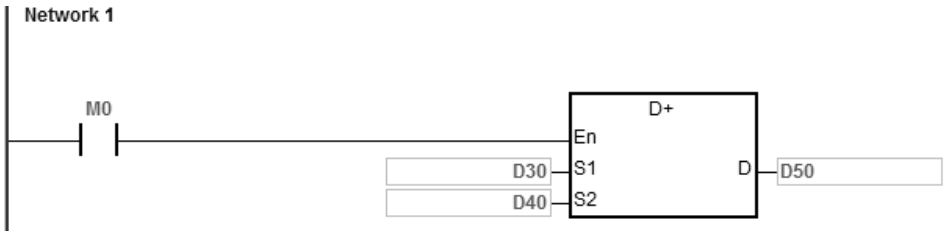
Adding 16-bit binary values: when M0 is ON, the instruction adds the addend in D10 to the augend in D0, and stores the sum in D20.



- When the values in D0 and D10 are 100 and 10 respectively, D0 plus D10 equals 110, and 110 is stored in D20.
- When the values in D0 and D10 are 16#7FFF and 16#1 respectively, D0 plus D10 equals 16#8000, and 16#8000 is stored in D20.
- When the values in D0 and D10 are 16#FFFF and 16#1 respectively, D0 plus D10 equals 16#10000. Since the operation result exceeds the range of 16-bit binary values, SM602 is ON, and the value stored in D20 is 16#0. Since the operation result is 16#0, SM600 is ON.

Example 2

Adding 32-bit binary values: when M0 is ON, the instruction adds the addend in (D41, D40) to the augend in (D31, D30), and stores the sum in (D51, D50). The data in D30, D40, and D50 is the lower 16-bit data, whereas the data in D31, D41, and D51 is the higher 16-bit data.



- When the values in (D31, D30) and (D41, D40) are 11111111 and 44444444 respectively, (D31, D30) plus (D41, D40) equals 55555555, and 55555555 is stored in (D51, D50).
- When the values in (D31, D30) and (D41, D40) are 16#80000000 and 16#FFFFFFFF respectively, (D31, D30) plus (D41, D40) equals 16#17FFFFFFF. Since the operation result exceeds the range of 32-bit binary values, SM602 is ON, and the value stored in (D51, D50) is 16#7FFFFFFF.

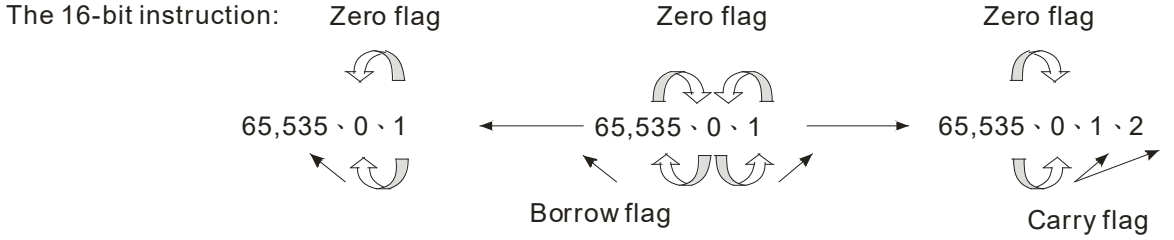
Flags

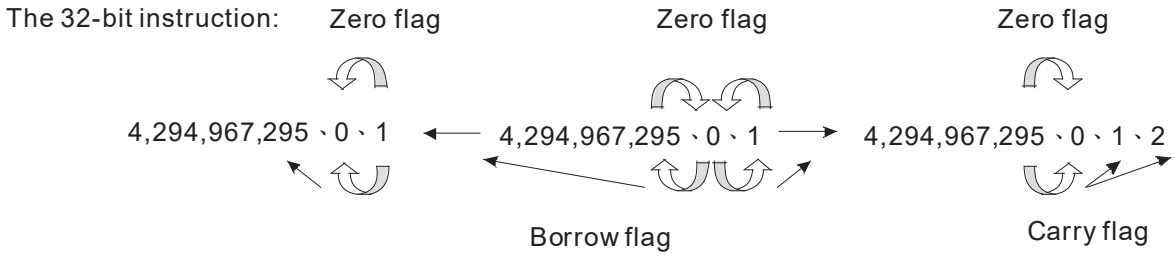
For 16-bit binary values:

1. If the operation result is zero, SM600 is ON.
2. If the operation result exceeds 65,535, SM602 is ON.

For 32-bit values:

1. If the operation result is zero, SM600 is set to ON.
2. If the operation result exceeds 4,294,967,295, SM602 is set to ON.





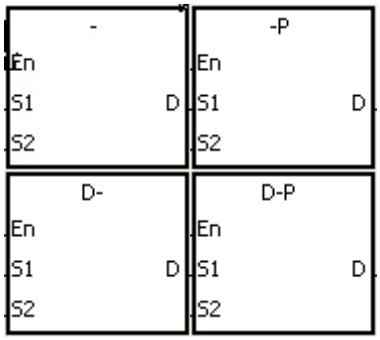
API	Instruction code			Operand	Function
0101	D	-	P	$S_1 \cdot S_2 \cdot D$	Subtracting binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Minuend

S_2 : Subtrahend

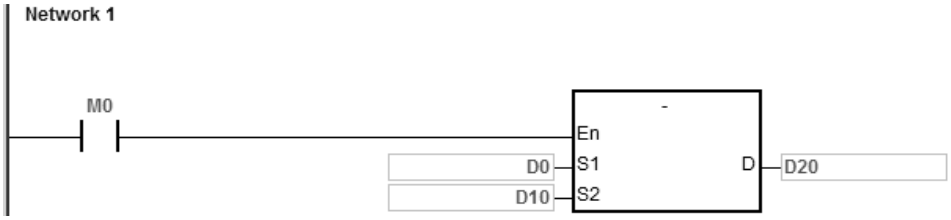
D : Difference

Explanation

1. This instruction subtracts the binary value in **S_2** from the binary value in **S_1**, and stores the difference in **D**.
2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.
3. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)
4. When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
5. When borrowing occurs during the arithmetic, SM601 is ON. Otherwise, it is OFF.

Example 1

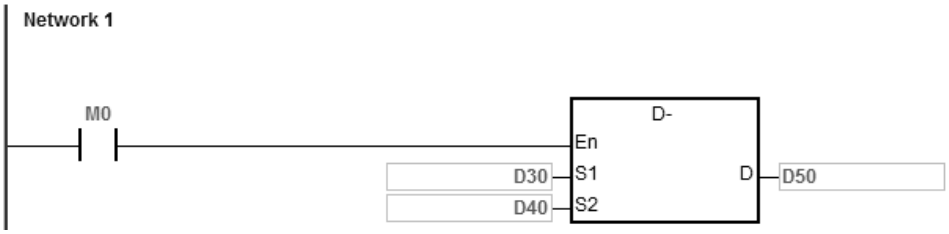
Subtracting 16-bit binary values: when M0 is ON, the instruction subtracts the subtrahend in D10 from the minuend in D0, and stores the difference in D20.



- When the values in D0 and D10 are 100 and 10 respectively, D0 minus D10 leaves 90, and 90 is stored in D20.
- When the values in D0 and D10 are 16#8000 and 16#1 respectively, D0 minus D10 leaves 16#7FFF, and 16#7FFF is stored in D20.
- When the values in D0 and D10 are 16#1 and 16#2 respectively, D0 minus D10 leaves 16#FFFF. Since borrowing occurs during the operation, SM601 is ON, and the value stored in D20 is 16#FFFF.
- When the values in D0 and D10 are 16#0 and 16#FFFF respectively, D0 minus D10 leaves 16#F0001. Since borrowing occurs during the operation, SM601 is ON, and the value stored in D20 is 16#1.

Example 2

Adding 32-bit binary values: when M0 is ON, the instruction subtracts the subtrahend in (D41, D40) from the minuend in (D31, D30), and stores the sum in (D51, D50). The data in D30, D40, and D50 is the lower 16-bit data, whereas the data in D31, D41, and D51 is the higher 16-bit data.



- When the values in (D31, D30) and (D41, D40) are 55555555 and 11111111 respectively, (D31, D30) minus (D41, D40) D10 leaves 44444444, and 44444444 is stored in (D51, D50).
- When the values in (D31, D30) and (D41, D40) are 16#80000000 and 16#FFFFFFFF respectively, (D31, D30) minus (D41, D40) leaves 16#F80000001. Since borrowing occurs during the operation, SM601 is ON, and the value stored in (D51, D50) is 16#80000001.

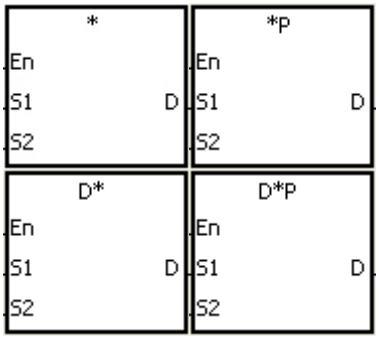
API	Instruction code			Operand	Function
0102	D	*	P	S_1 · S_2 · D	Multiplying binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol:



S_1 : Multiplicand

S_2 : Multiplier

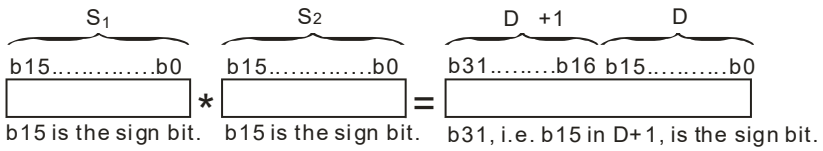
D : Product

Explanation

1. This instruction multiplies the signed binary value in S_1 by the signed binary value in S_2, and stores the product in **D**.

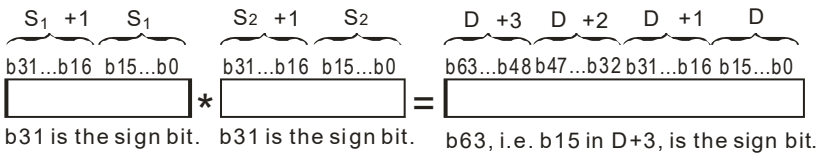
2. Only the instruction D* can use the 32-bit counter.

3. Multiplying 16-bit binary values:



The product is a 32-bit value, and is stored in the register (D+1, D), which is composed of 32 bits. When the sign bit b31 is 0, the product is a positive value. When the sign bit b31 is 1, the product is a negative value.

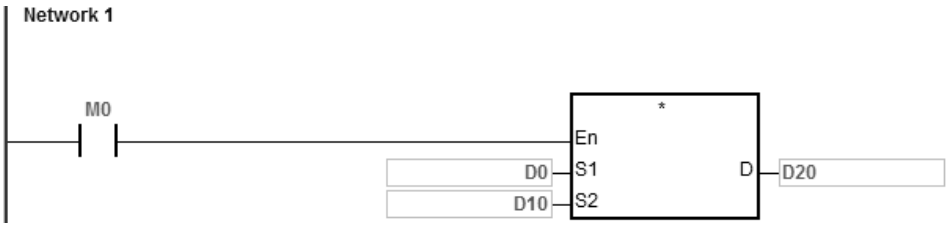
4. Multiplying 32-bit binary values:



The product is a 64-bit value, and is stored in the register (D+3, D+2, D+1, D0), which is composed of 64 bits. When the sign bit b63 is 0, the product is a positive value. When the sign bit b63 is 1, the product is a negative value.

Example

The instruction multiplies the 16-bit value in D0 by the 16-bit value in D10, and stores the 32-bit product in (D21, D20). The data in D21 is the higher 16-bit data, whereas the data in D20 is the lower 16-bit data. Whether the result is a positive value or a negative value depends on the state of the highest bit b31. When b31 is OFF, the result is a positive value. When b31 is ON, the result is a negative value.



D0×D10=(D21, D20)

16-bit value×16-bit value=32-bit value

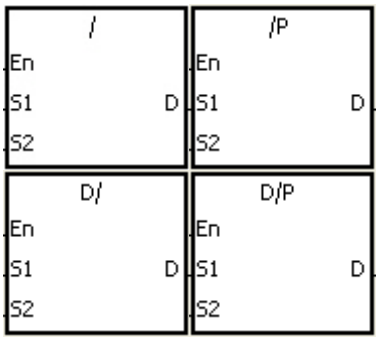
API	Instruction code			Operand	Function
0103	D	/	P	**S_1 · S_2 · D**	Dividing binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



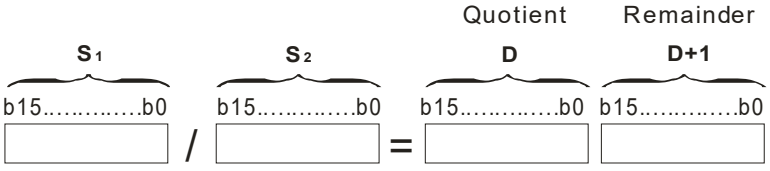
S_1 : Dividend

S_2 : Divisor

D : Quotient; remainder

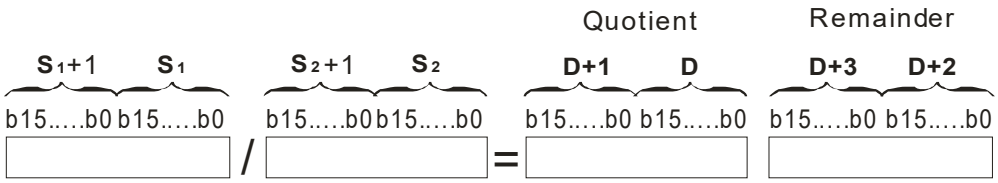
Explanation

1. This instruction divides the signed binary value in **S_1** by the signed binary value in **S_2**, and stores the quotient and the remainder in **D**.
2. Only the 32-bit instructions can use the 32-bit counter.
3. When the sign bit is 0, the value is a positive one. When the sign bit is 1, the value is a negative one.
4. Dividing 16-bit values:



The operand **D** occupies two consecutive devices. The quotient is stored in **D**, and the remainder is stored in **D**+1.

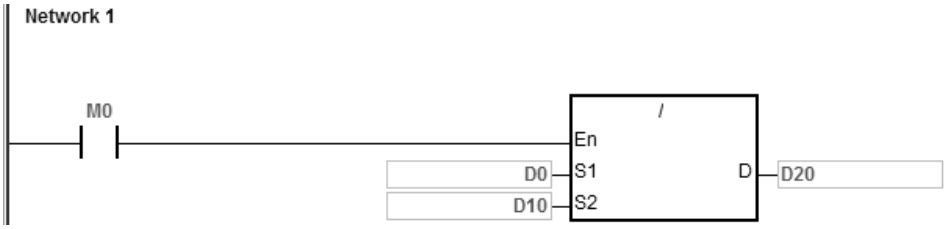
5. Dividing 32-bit values:



The operand **D** occupies two devices. The quotient is stored in (**D**+1, **D)**, and the remainder is stored in (**D**+3, **D**+2).

Example

When M0 is ON, the instruction divides the dividend in D0 by the divisor in D10, and stores the quotient in D20, and stores the remainder in D21. Whether the result is a positive value or a negative value depends on the state of the highest bit.



Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the divisor is 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2012.
3. If you declare the operand **D** used during the execution of the 16-bit instruction in ISPSoft, the data type is ARRAY [2] of WORD/INT.
4. If you declare the operand **D** used during the execution of the 32-bit instruction in ISPSoft, the data type is ARRAY [2] of DWORD/DINT.

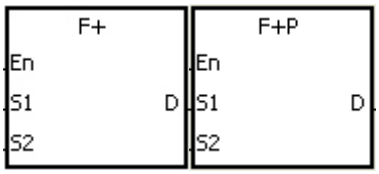
API	Instruction code			Operand	Function
0104		F+	P	S_1 · S_2 · **D**	Adding floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							○
S_2					●	●	●	●	●							○
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Augend

S_2 : Addend

D : Sum

Explanation

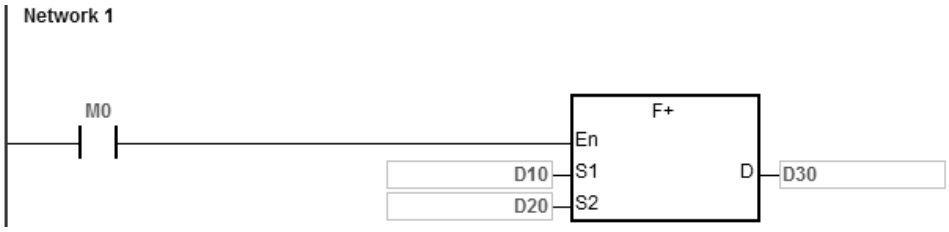
1. This instruction adds the 32-bit single-precision floating-point numbers in S_2 and S_1, and stores the sum in **D**.

2. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
   - When the absolute value of the operation result is less than the value that can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value that can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.

Example

Adding single-precision floating-point numbers: when M0 is ON, the instruction adds the addend 16#4046B852 in (D21, D20) to the augend 16#3FB9999A in (D11, D10), and stores the sum 16#4091C28F in (D31, D30). 16#4046B852, 16#3FB9999A, and 16#4091C28F represent the floating point numbers 3.105, 1.450, and 4.555 respectively.



Additional remark

If the value in S_1 or the value in S_2 exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

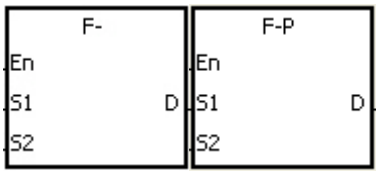
API	Instruction code			Operand	Function
0105		F-	P	**S_1 · S_2 · D**	Subtracting floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							○
S_2					●	●	●	●	●							○
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Minuend

S_2 : Subtrahend

D : Difference

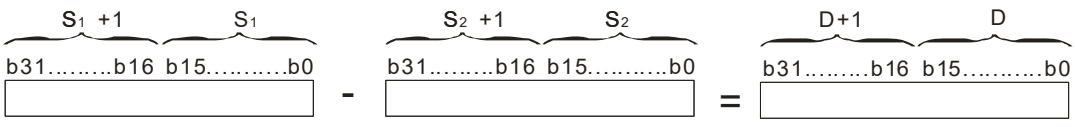
6

Explanation

1. This instruction subtracts the 32-bit single-precision floating-point number in **S_2** from the 32-bit single-precision floating-point numbers number in **S_1**, and stores the difference in **D**.

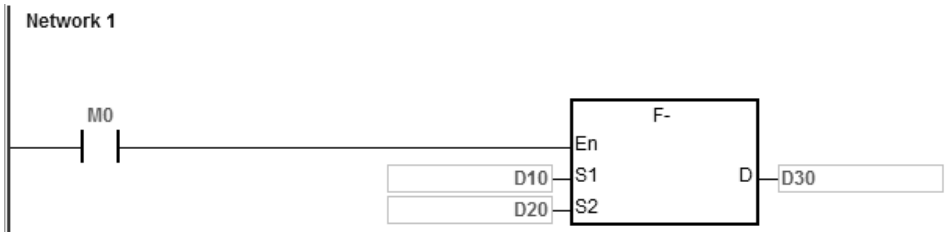
2. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value that can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value that can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.



Example

Subtracting 32-bit single-precision floating-point numbers: when M0 is ON, the instruction subtracts the subtrahend in (D21, D20) from the minuend in (D21, D20), and stores the difference in (D31, D30).



Additional remarks

If the value in S_1 or the value in S_2 exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

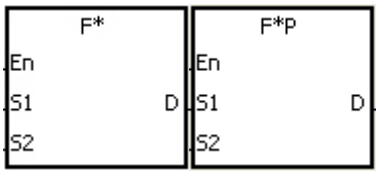
API	Instruction code			Operand	Function
0106		F*	P	**S_1 · S_2 · D**	Multiplying floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							○
S_2					●	●	●	●	●							○
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Multiplicand

S_2 : Multiplier

D : Product

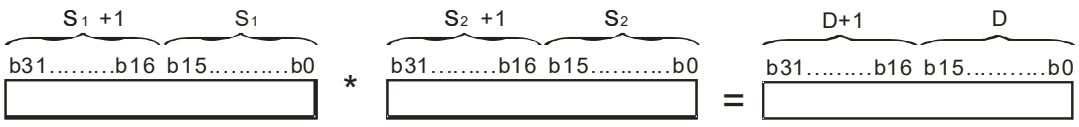
6

Explanation

1. This instruction multiplies the 32-bit single-precision floating-point number in **S_1** by the 32-bit single-precision floating-point number in **S_2**, and stores the product in **D**.

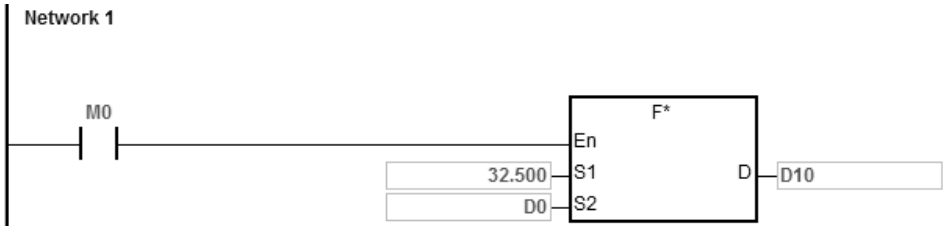
2. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value that can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value that can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.



Example

Multiplying 32-bit single-precision floating-point numbers: when M0 is ON, the instruction multiplies the multiplicand 32.5 by the multiplier in (D1, D0), and stores the product in (D11, D10).



Additional remarks

If the value in S_1 or the value in S_2 exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

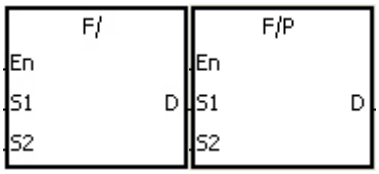
API	Instruction code			Operand	Function
0107		F/	P	**S_1 · S_2 · D**	Dividing floating-point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							○
S_2					●	●	●	●	●							○
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Dividend

S_2 : Divisor

D : Quotient

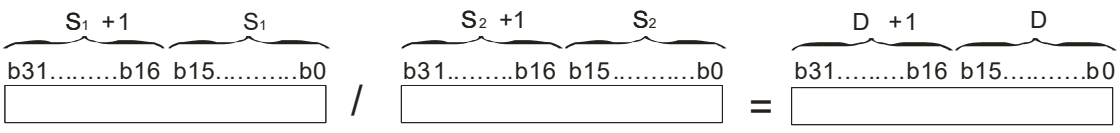
6

Explanation

1. This instruction is divides the 32-bit single-precision floating-point number in **S_1** by the 32-bit single-precision floating-point number in **S_2**, and stores the quotient in **D**.

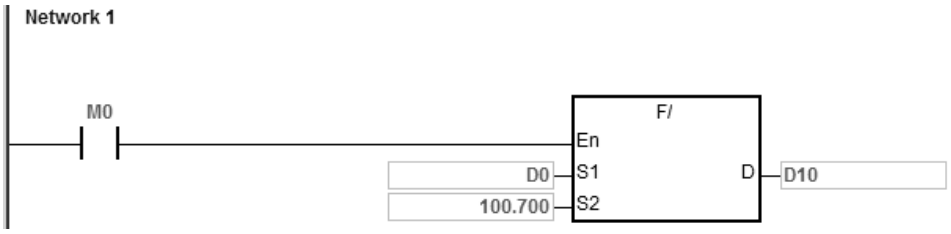
2. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

   - When the operation result is zero, SM600 is ON.
   - When the absolute value of the operation result is less than the value that can be represented by the minimum floating-point number, the value in **D** is 16#FF800000 and SM601 is ON.
   - When the absolute value of the operation result is larger than the value that can be represented by the maximum floating-point number, the value in **D** is 16#7F800000 and SM602 is ON.



Example

Dividing 32-bit single-precision floating-point numbers: when M0 is ON, the instruction divides the dividend in (D1, D0) by the divisor 100.7, and stores the quotient in (D11, D10).



Additional remarks

1. If the divisor is 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2012.
2. If the value in S_1 or the value in S_2 exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

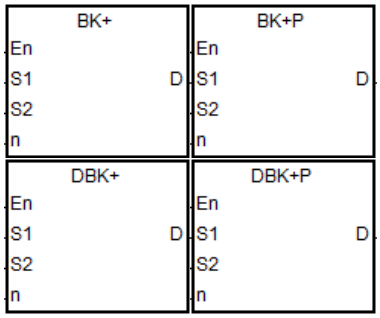
API	Instruction code			Operand	Function
0112	D	BK+	P	**S_1 · S_2 · n · D**	Adding binary values in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							
S_2					●	●	●	●	●				○	○		
n					●	●	●	●	●				○	○		
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●					●	●	
S_2		●			●	●					●	●	
n		●			●	●					●	●	
D		●			●	●					●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Augend

S_2 : Addend

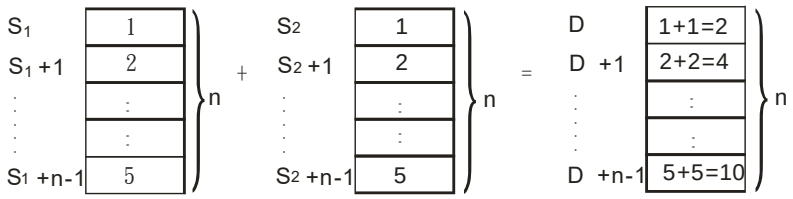
n : Data length

D : Sum

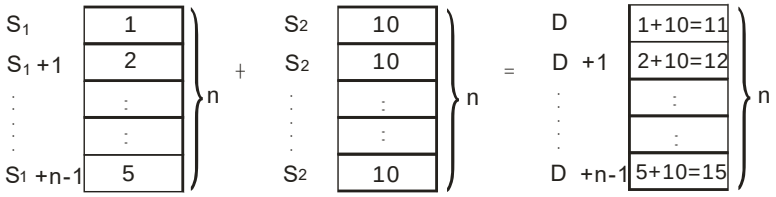
Explanation

1. This instruction adds **n** pieces of data in devices starting from **S_2** to those in devices starting from **S_1**. The augends and the addends are binary values, and the instruction stores the sums in **D**.
2. The operand n must be between 1–256.
3. Only the 32-bit instructions can use the 32-bit counter.
4. When the operation result is zero, SM600 is ON.
5. For the 16-bit instructions, when the operation result is less than –32,768, SM601 is ON.
6. For the 16-bit instructions, when the operation result is larger than 32,767, SM602 is ON.
7. For the 32-bit instructions, when the operation result is less than–21,474,836,488, SM601 is ON.
8. For the 32-bit instructions, when the operation result is larger than 2,147,483,647, SM602 is ON.

9. 16-bit instruction example: when the operand **S_2** is a device (not a constant or a hexadecimal value)

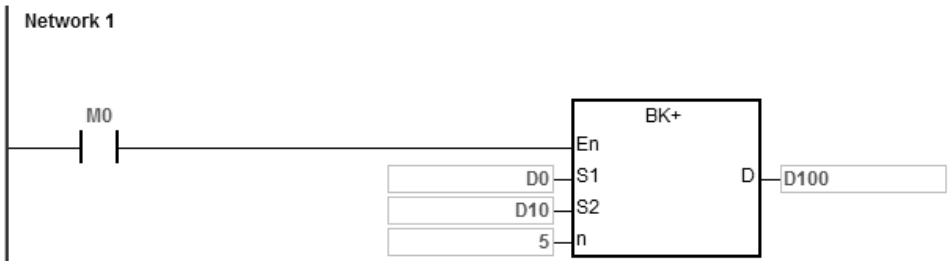


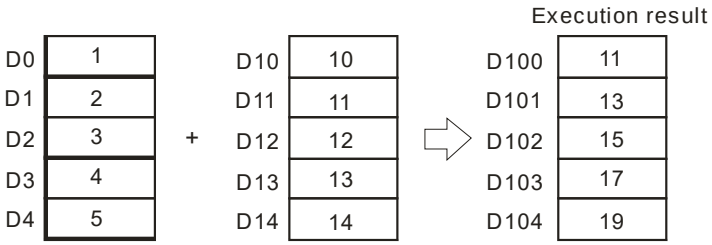
10. 16-bit instruction example: when the operand **S_2** is a constant or a hexadecimal value



Example 1

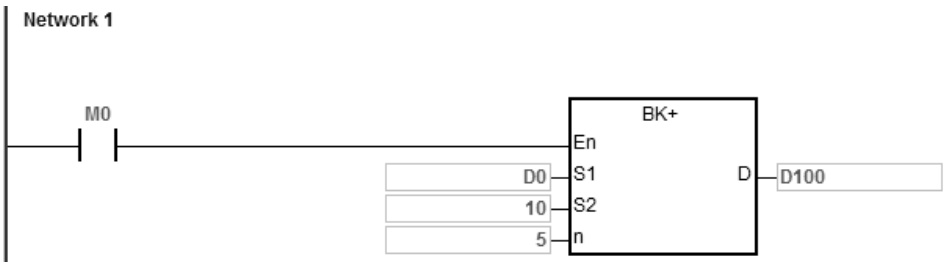
When M0 is ON, the instruction adds the binary values in D10–D14 to the binary values in D0–D4, and stores the sums.



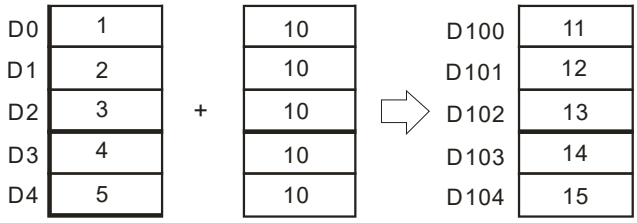


Example 2

When M0 is ON, the instruction adds the addend 10 to the binary values in D0–D4, and stores the sums in D100–D104.



Execution result



Additional remarks

1. For 16-bit instructions, if the devices **S**$_1$–**S**$_1$+**n**-1, **S**$_2$–**S**$_2$+**n**-1, or **D**–**D**+**n**-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

2. For 32-bit instructions, if the devices **S**$_1$–**S**$_1$+2*__n__-1, **S**$_2$–**S**$_2$+2*__n__-1, or **D**–**D**+2*__n__-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

3. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. For 16-bit instructions, if **S**$_1$–**S**$_1$+**n**-1 overlap **D**–**D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. For 32-bit instructions, if **S**$_1$–**S**$_1$+2*__n__-1 overlap **D**–**D**+2*__n__-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

6. For 16-bit instructions, if **S**$_2$–**S**$_2$+**n**-1 overlap **D**–**D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

7. For 32-bit instructions, if **S**$_2$–**S**$_2$+2*__n__-1 overlap **D**–**D**+2*__n__-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

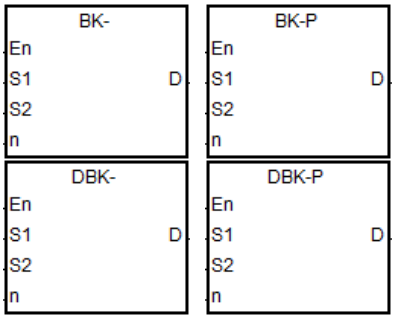
API	Instruction code			Operand	Function
0113	D	BK-	P	S_1 · S_2 · **n** · **D**	Subtracting binary values in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							
S_2					●	●	●	●	●				○	○		
n					●	●	●	●	●				○	○		
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●					●	●	
S_2		●			●	●					●	●	
n		●			●	●					●	●	
D		●			●	●					●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Minuend

S_2 : Subtrahend

n : Data length

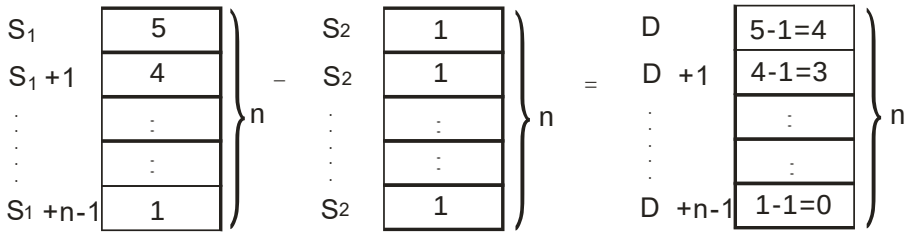
D : Difference

Explanation

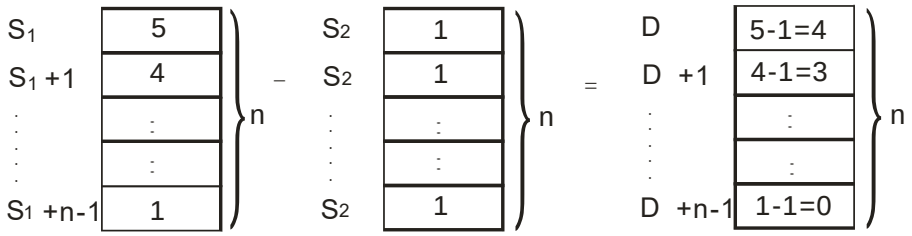
1. This instruction subtracts **n** pieces of data in devices starting from S_2 from those in devices starting from S_1. The minuends and the subtrahends are binary values, and the instruction stores the differences in **D**.
2. The operand n must be between 1–256.
3. Only the 32-bit instructions can use the 32-bit counter.
4. When the operation result is zero, SM600 is ON.
5. For 16-bit instructions, when the operation result is less than –32,768, SM601 is ON.
6. For 16-bit instructions, when the operation result is larger than 32,767, SM602 is ON.
7. For 32-bit instructions, when the operation result is less than –2,147,483,648, SM601 is ON.

8. For 32-bit instructions, when the operation result is larger than 2,147,483,647, SM602 is ON.

9. 16-bit instruction example: when the operand **S_2** is a device (not a constant or a hexadecimal value)

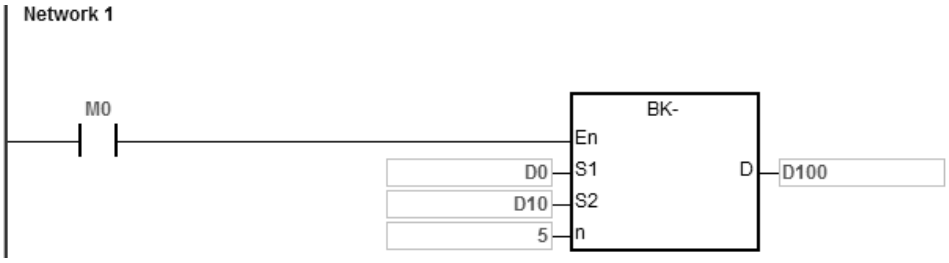


10. 16-bit instruction example: when the operand **S_2** is a constant or a hexadecimal value



Example 1

When M0 is ON, the instruction subtracts the binary values in D10–D14 from the binary values in D0–D4, and stores the differences in D100–D104.

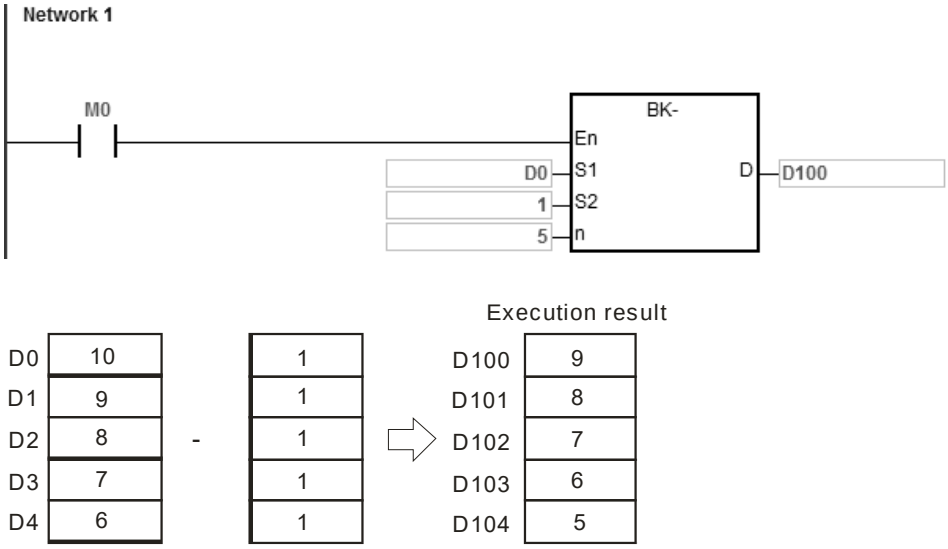


Execution result

D0	5	D10	1	D100	4	1	
D1	4	D11	2	D101	2	SM600	
D2	3	- D12	3	D102	0		
D3	2	D13	4	D103	−2		
D4	1	D14	5	D104	−4		

Example 2

When M0 is ON, the instruction subtracts the subtrahend 1 from the binary values in D0–D4, and stores the differences in D100–D104.



Additional remarks

1. For 16-bit instructions, if the devices **S_1**–**S_1+n**-1, **S_2**–**S_2+n**-1, or **D–D+n**-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

2. For 32-bit instructions, if the devices **S_1**–**S_1**+2*n-1, **S_2**–**S_2**+2*n-1, or **D–D**+2*n-1 exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

3. If **n** Is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. For 16-bit instructions, if **S_1**–**S_1+n**-1 overlap **D–D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. For 32-bit instructions, if **S_1**–**S_1**+2*n-1 overlap **D**–**D**+2*n-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

6. For 16-bit instructions, if **S_2**–**S_2+n**-1 overlap **D–D+n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

7. For 32-bit instructions, if **S_2**–**S_2**+2*n-1 overlap **D**–**D**+2*n-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

API	Instruction code			Operand	Function
0114		$+	P	$S_1 \cdot S_2 \cdot D$	Linking strings

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													●
S_2													●
D													●

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S_1 : String 1

S_2 : String 2

D : Device in which the string is stored

Explanation

1. This instruction links the string starting with the data in the device specified by **S_1** (exclusive of 16#00), and the string starting with the data in the device specified by **S_2** (exclusive of 16#00) and stores the result in **D**. In addition, the instruction adds the code 16#00 to the end of the linked string in **D**. When the instruction is not executed, the data in **D** is unchanged.

2. When **S_1, S_2** or **D** is not a string ($), the content of the data source can be up to 256 characters (including the ending code 16#00).

3. If successful, the string in **S_1** and the string in **S_2** are linked and stored in **D**, as shown below.

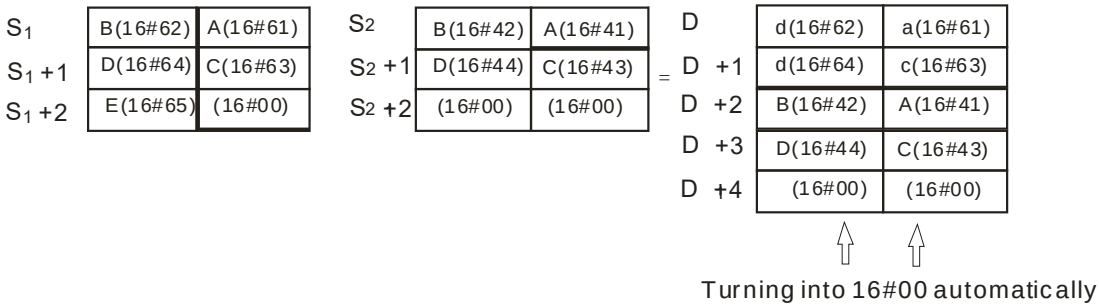
S_1	B(16#62)	A(16#61)
S_1 +1	D(16#64)	C(16#63)
S_1 +2	(16#00)	E(16#65)

S_2	B(16#42)	A(16#41)
S_2 +1	D(16#44)	C(16#43)
S_2 +2	(16#00)	(16#00)

=

D	B(16#62)	A(16#61)
D +1	D(16#64)	C(16#63)
D +2	A(16#41)	E(16#65)
D +3	C(16#43)	B(16#42)
D +4	(16#00)	D(16#44)

Turning into 16#00 automatically



4. When **S1, S2** or **D** is not a string ($),the ending code 16#00 is added to the end of the data that is moved.

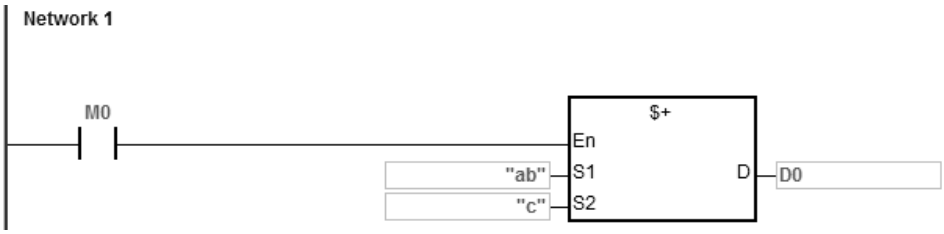
5. If **S1** or **S2** is not a string, then when the instruction is executed and the first character is the code 16#00, 16#00 is still linked and moved.

6. The string “abcde” in **S1** is shown as below.

S1	b(16#62)	a(16#61)
S1 +1	d(16#64)	c(16#63)
S1 +2	(16#00)	e(16#65)

Example

Suppose **S1** is the string “ab” and **S2** is the string “c”. After the conditional contact M0 is enabled, the data in D0 is 16#6261 and the data in D65535 is16#0063.



Additional remarks

1. If **S1** or **S2** is a string, at most 31 characters can be moved.

2. If D is not sufficient to contain the string composed of the strings in **S1** and **S2**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the string of **S1**+**S2** is more than 256 characters (the ending code 16#00 included), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **S1** or **S2** overlaps **D**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

5. If the string in **S1** or **S2** does not end with 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200E.

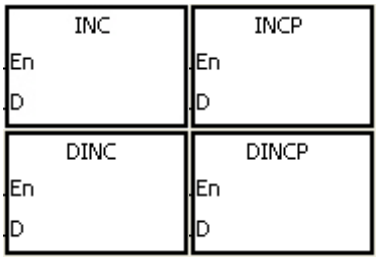
API	Instruction code			Operand	Function
0115	D	INC	P	**D**	Adding one to a binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Destination device

Explanation

1. This instruction adds 1 to the value in **D**.
2. Only the DINC instruction can use the 32-bit counter.
3. For the 16-bit operation, 32,767 plus 1 equals -32,768. For the 32-bit operation, 2,147,483,647 plus 1 equals -2,147,483,648.

Example

When M0 switches from OFF to ON, the value in D0 increases by one.

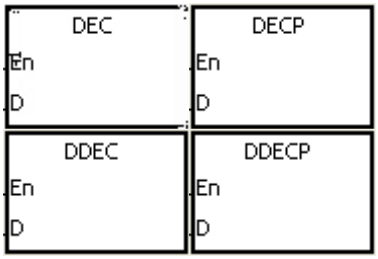
API	Instruction code			Operand	Function
0116	D	DEC	P	**D**	Subtracting one from a binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Destination device

Explanation

1. This instruction subtracts 1 from the value in **D**.
2. Only the DDEC instruction can use the 32-bit counter.
3. For the 16-bit operation, -32,768 minus 1 leaves 32,767. For the 32-bit operation, -2,147,483,648 minus 1 leaves 2,147,483,647.

Example

When M0 switches from OFF to ON, the value in D0 decreases by one.

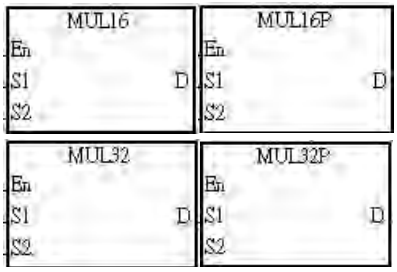
API	Instruction code			Operand	Function
0117		MUL16 MUL32	P	S_1 · S_2 · D	Multiplying 16-bit binary numbers Multiplying 32-bit binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Multiplicand

S_2 : Multiplier

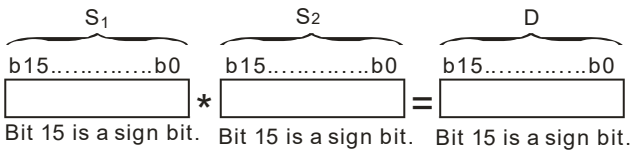
D : Product

Explanation

1. This instruction multiplies the signed binary value in **S_1** by the signed binary value in **S_2**, and stores the product in **D**.

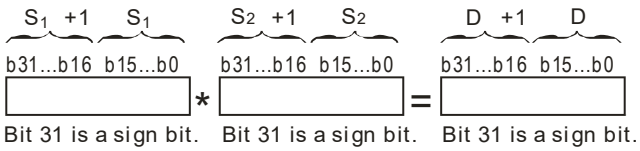
2. Only the MUL32 instruction can use an HC device.

3. 16-bit binary multiplication:



The product is a 16-bit value stored in **D** which is a 16-bit register. If b15 in **D** is 0, the product stored in **D** is a positive value. If b15 in **D** is 1, the product stored in **D** is a negative value.

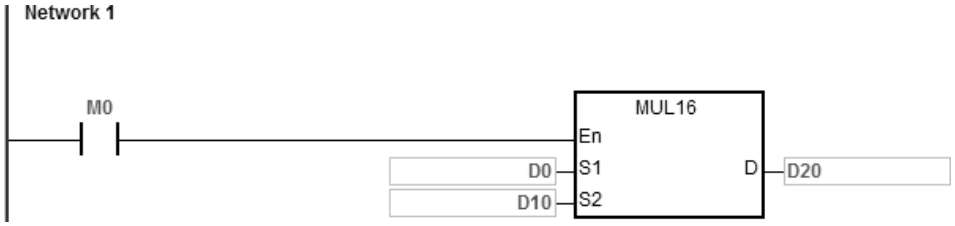
4. 32-bit binary multiplication:



The product is a 32-bit value stored in (**D**, **D**+1**)** which is a 32-bit register. If b31 in **D** is 0, the product stored in (**D**, **D**+1**)** is a positive value. If b31 in **D** is 1, the product stored in (**D**, **D**+1**)** is a negative value.

Example

The instruction multiplies the 16-bit value in D0 by the 16-bit value in D10, and stores the product in D20. The sign of the product (positive or negative) depends on the leftmost bit (bit 15) in D20. If bit 15 in D20 is 0, the product stored in D20 is a positive value. If bit 15 in D20 is 1, the product stored in D20 is a negative value.



D0×D10=D20

16-bit value×16-bit value=16-bit value

Additional remarks

1. If the product of a 16-bit multiplication is not a 16-bit signed value available, and is greater than the maximum 16-bit positive number K32767, or less than the minimum negative number K-32768, the carry flag SM602 is ON, and only the low 16 bits are written.

2. If you need the complete result of a 16-bit multiplication (a 32-bit value), use the */*P instruction (API 0102). Refer to the explanation for the * instruction (API 0102) for more information.

3. If the product of a 32-bit multiplication is not a 32-bit signed value available, and is greater than the maximum 32-bit positive number K2147483647, or less than the minimum negative number K-2147483648, the carry flag SM602 is ON, and only the low 32 bits are written.

4. If you need the complete result of a 32-bit multiplication (a 64-bit value), use API 0102 D*/D*P. Refer to the explanation for the * instruction (API 0102) for more information.

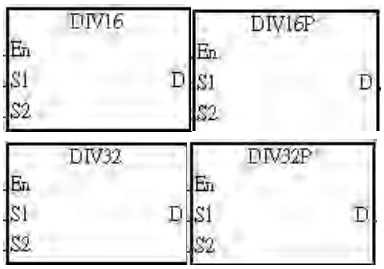
API	Instruction code			Operand	Function
0118		DIV16 DIV32	P	**S_1 · S_2 · D**	Dividing 16-bit binary numbers Dividing 32-bit binary numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	●	●		
S_2					●	●	●	●	●		○	○	●	●		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Dividend

S_2 : Divisor

D : Quotient; remainder

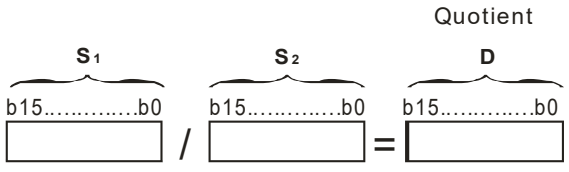
Explanation

1. This instruction divides the signed binary value in **S_1** by the signed binary value in **S_2**, and stores the quotient in **D**.

2. Only the 32-bit instruction can use an HC device.

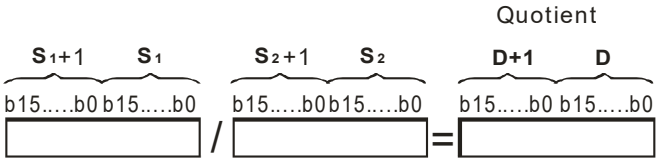
3. Sign bit=0 (Positive number); sign bit =1 (Negative number)

4. 16-bit binary division:



The quotient is stored in **D**.

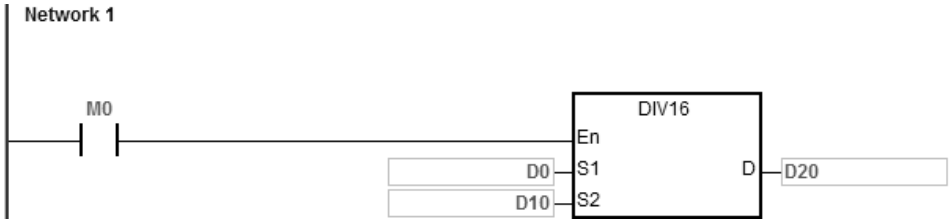
5. 32-bit binary division:



D occupies two consecutive devices. The quotient is stored in (**D**+1, **D**).

Example

When M0 is ON, the instruction divides the dividend in D0 by the divisor in D10, and stores the quotient D20. Whether the quotient is a positive value or a negative value depends on the leftmost bit in D20.



Additional remarks

1. If the device is not available, the instruction is not executed, SM0 will be ON, and the error code stored in SR0 is 16#2003.

2. If the divisor is 0, the instruction is not executed, SM0 will be ON, and the error code stored in SR0 is 16#2012.

3. If you want to store the remainder, use the "/" instruction (Dividing binary values). Refer to the explanation for the "/" instruction (API 0103) for more information.

6.3 Data Conversion Instructions

6.3.1 List of Data Conversion Instructions

The following table lists the Data Conversion instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0200	BCD	DBCD	✓	Converting a binary number into a binary-coded decimal number
0201	BIN	DBIN	✓	Converting a binary-coded decimal number into a binary number
0202	FLT	DFLT	✓	Converting a binary integer into a binary floating-point number
0204	INT	DINT	✓	Converting a 32-bit floating-point number into a binary integer
0206	MMOV	–	✓	Converting a 16-bit value into a 32-bit value
0207	RMOV	–	✓	Converting a 32-bit value into a 16-bit value
0208	GRY	DGRY	✓	Converting a binary number into a Gray code
0209	GBIN	DGBIN	✓	Converting a Gray code into a binary number
0210	NEG	DNEG	✓	Two's complement
0211	–	FNEG	✓	Reversing the sign of a 32-bit floating-point number
0212	–	FBCD	✓	Converting a binary floating-point number into a decimal floating-point number
0213	–	FBIN	✓	Converting a decimal floating-point number into a binary floating-point number
0214	BKBCD	–	✓	Converting a binary numbers in blocks into a binary-coded decimal numbers in blocks
0215	BKBIN	–	✓	Converting a binary numbers in blocks into a binary-coded decimal numbers in blocks
0216	SCAL	DSCAL	✓	Finding a scaled value (point-slope)
0217	SCLP	DSCLP	✓	Finding a scaled value (two points)
0222	SCLM	DSCLM	✓	Multi-point area ratio operation

6.3.2 Explanation of Data Conversion Instructions

API	Instruction code			Operand	Function
0200	D	BCD	P	**S, D**	Converting a binary number into a binary-coded decimal number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○				
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

S : Source device

D : Conversion result

Explanation

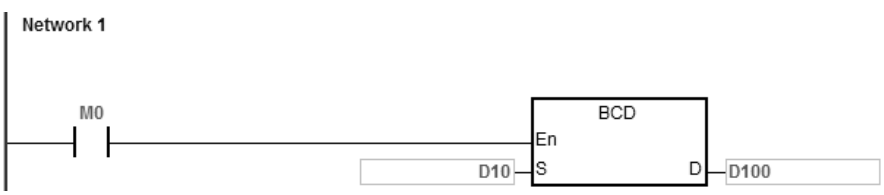
1. This instruction converts a binary value in **S** into a binary-coded decimal value, and stores the conversion result in **D**.

2. Only the DBCD instruction can use the 32-bit counter, but not the device E.

3. The four fundamental arithmetic operations in the PLC, the INC instruction, and the DEC instruction all operate on binary numbers. To show the decimal value on the display, use the BCD instruction to convert a binary value into a binary-coded decimal value

Example

1. When M0 is ON, the instruction converts a binary value in D10 into a binary-code decimal value, and stores the conversion result in D100.



2. If D10=16#04D2=1234, the conversion result is D100=16#1234.

Additional remarks

1. If the conversion result exceeds the range 0–9,999, the instruction BCD is not executed, SM0 is ON, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not between 0–9.

2. If the conversion result exceeds the range 0–99,999,999, the instruction DBCD is not executed, SM0 is ON, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not between 0–9.

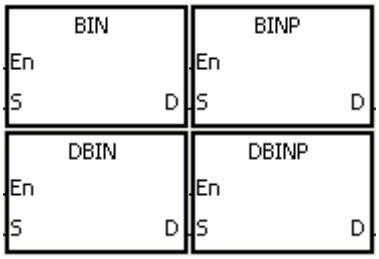
API	Instruction code			Operand	Function
0201	D	BIN	P	**S, D**	Converting a binary-coded decimal number into a binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○				
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Conversion result

Explanation

1. This instruction converts a binary-coded decimal value in **S** into a binary value, and stores the conversion result in **D**.

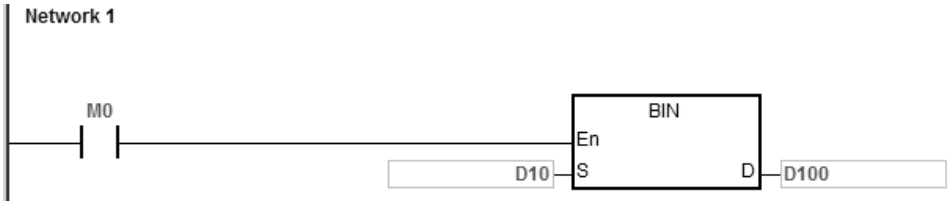
2. The 16-bit binary-coded decimal value in **S** must be between 0–9,999, and the 32-bit binary-coded decimal value in **S** must be between 0–99,999,999.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

4. The system converts constants and hexadecimal values into binary values automatically. Therefore, you do not need to use the instruction for that conversion.

Example

When M0 is ON, the instruction converts the binary-coded decimal value in D10 into the binary value, and stores the conversion result in D100.

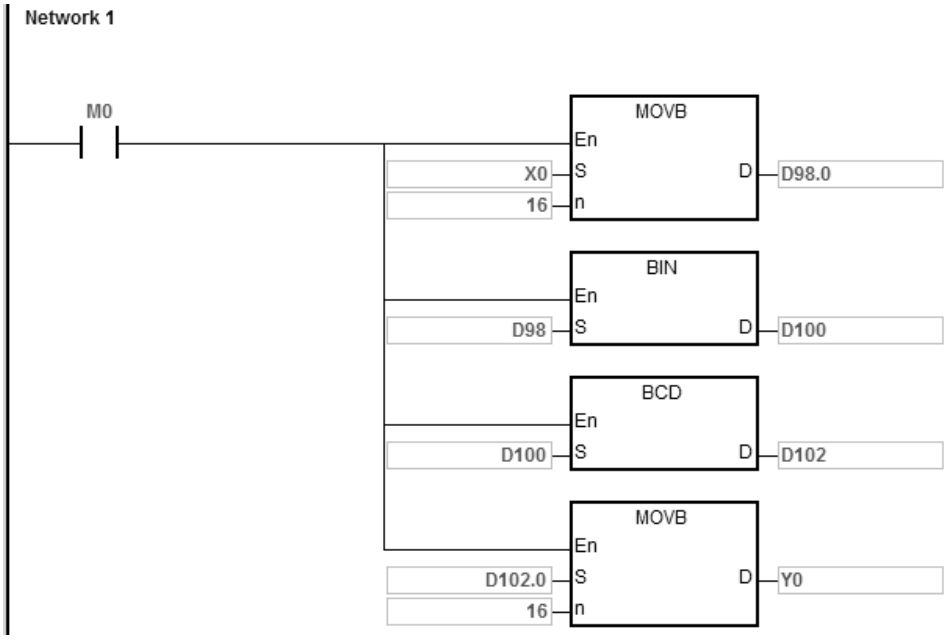


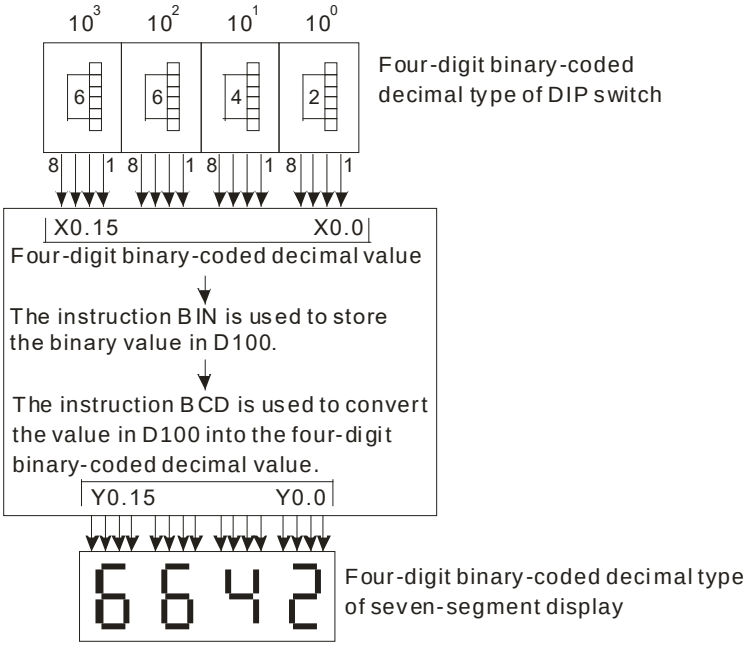
Additional remarks

1. If the value in **S** is not the binary-coded decimal value, an operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal value, but one of digits is not between 0–9.

2. Application of the BCD and BIN instructions:

   - Before the value of the binary-coded decimal type of DIP switch is read into the PLC, use the BIN instruction to convert the data into the binary value and store the conversion result in the PLC.
   - If you want to display the data stored inside the PLC in a seven-segment display of the binary-coded decimal type, use the BCD instruction to convert the data into the binary-coded decimal value before the data is sent to the seven-segment display.
   - When M0 is ON, move values in X0-X17 to D98. BIN instruction converts the binary-coded decimal value into the binary value, and stores the conversion result in D100. Subsequently, the BCD instruction converts the binary value in D100 into the binary-coded decimal value and stores the value in D102; after that move the conversion result to Y0–Y17.




10^3 10^2 10^1 10^0
6 6 4 2
Four-digit binary-coded decimal type of DIP switch
8 1 8 1 8 1 8 1
X0.15 X0.0
Four-digit binary-coded decimal value
The instruction BIN is used to store the binary value in D100.
The instruction BCD is used to convert the value in D100 into the four-digit binary-coded decimal value.
Y0.15 Y0.0
6 6 4 2
Four-digit binary-coded decimal type of seven-segment display

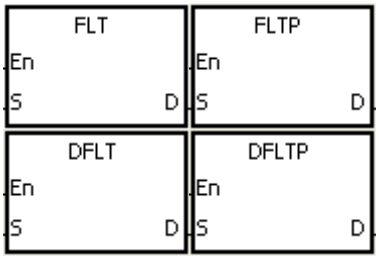
API	Instruction code			Operand	Function
0202	D	FLT	P	**S, D**	Converting a binary integer into a binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○				
D							●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Conversion result

Explanation

1. This instruction converts the binary integer in **S** into the single-precision floating-point number and stores the conversion result in **D**.

2. The operand **S** used in the instruction FLT cannot be the 32-bit counter, and not the device E.

3. The source device **S** used in the instruction FLT occupies one register, and **D** used in FLT occupies two registers.

4. The source device **S** used in the instruction DFLT occupies two registers, and **D** used in DFLT also occupies two registers.

   - When the absolute value of the conversion result is larger than the value that can be represented by the maximum floating-point number, SM602 is ON, and the maximum floating-point number is stored in **D**.
   - When the absolute value of the conversion result is less than the value that can be represented by the minimum floating-point number, SM601 is ON, and the minimum floating-point number is stored in **D**.
   - When the conversion result is zero, SM600 is ON.

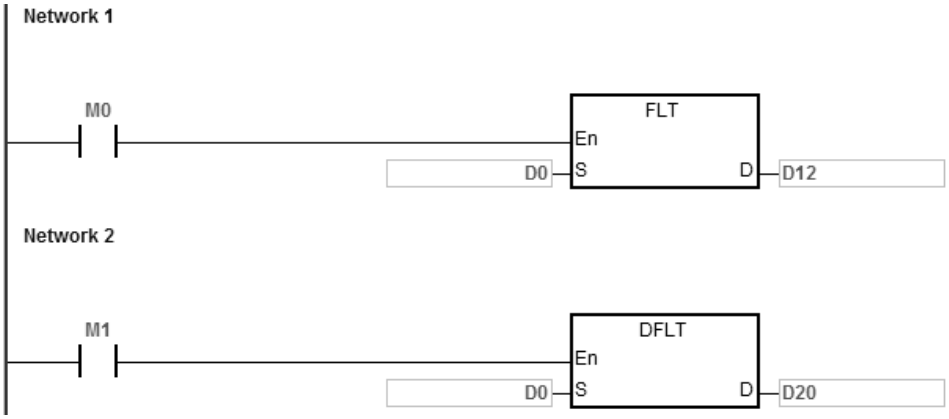
Example 1

1. When M0 is ON, the instruction converts the binary integer in D0 into a single-precision floating-point number, and stores the conversion result in (D13, D12).

2. When M1 is ON, the instruction converts the binary integer in (D1, D0) into a single-precision floating-point number, and stores the conversion result in (D21, D20).

3. Suppose the value in D0 is 10. When M0 is ON, the instruction converts 10 into the single-precision floating-point number 16#41200000, and then stores 16#41200000 in the 32-bit register (D13, D12).

4. Suppose the value in the 32-bit register (D1, D0) is 100,000. When M1 is ON, the instruction converts 100,000 into the single-precision floating-point number 16#47C35000, and stores 16#47C35000 in the 32-bit register (D21, D20).

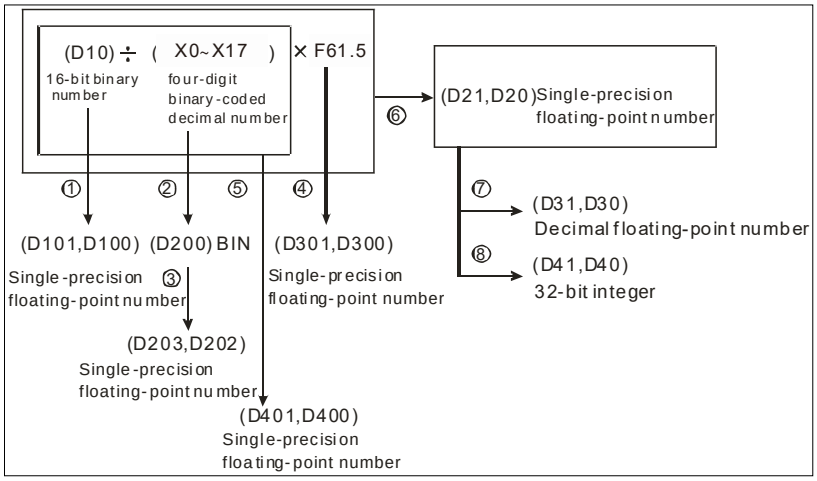


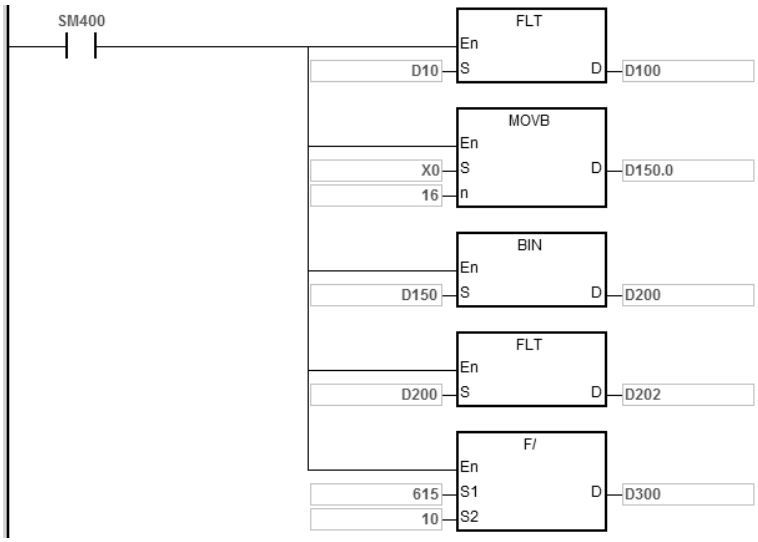
Example 2

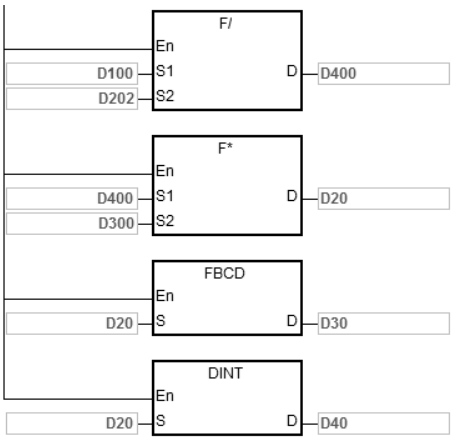
You can use the applied instructions to perform the following calculation.

- Convert the binary integer in D10 into the single-precision floating-point number, and store the conversion result in (D101, D100).
- Move the binary-coded decimal value in X0–X17 to D150.
- Convert the binary-coded decimal value in D150 into the binary value and store the conversion result in D150.
- Convert the binary integer in D200 into the single-precision floating-point number, and store the conversion result in (D203, D202).
- Divide the constant 615 by the constant 10, and store the quotient which is the single-precision floating-point number in (D301, D300).
- Divide the single-precision floating-point number in (D101, D100) by the single-precision floating-point number in (D203, D202), and store the quotient which is the single-precision floating-point number in (D401, D400).

- Multiply the single-precision floating-point number in (D401, D400) by the single-precision floating-point number in (D301, D300), and store the product which is the single-precision floating-point number in (D21, D20).
- Convert the single-precision floating-point number in (D21, D20) into the decimal floating-point number, and store the conversion result in (D31, D30).
- Convert the single-precision floating-point number in (D21, D20) into the binary integer, and store the conversion result in (D41, D40).






F/
En
D100
S1
D
D400
D202
S2
F*
En
D400
S1
D
D20
D300
S2
FBCD
En
D20
S
D
D30
DINT
En
D20
S
D
D40

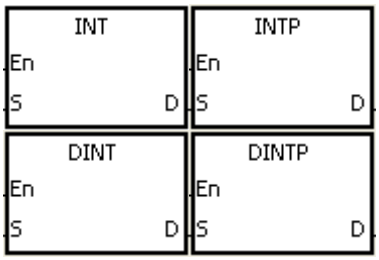
API	Instruction code			Operand	Function
0204	D	INT	P	**S, D**	Converting a 32-bit floating-point number into a binary integer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Conversion result

6

Explanation

1. This instruction converts a single-precision floating-point number in **S** into a binary integer, then rounds the binary floating-point number down to the nearest whole digit to becomes a binary integer, and then the instruction stores the binary integer in **D**.
2. The source device **S** used in the INT instruction occupies two registers, and **D** used in INT occupies one register.
3. The source device **S** used in the DINT instruction occupies two registers, and **D** used in DINT also occupies two registers.
4. The operand **D** used in the INT instruction cannot be the 32-bit counter, but not the device E.
5. The INT instruction is the opposite of the FLT instruction .
6. When the conversion result is zero, SM600 is ON.
7. During the conversion, if the floating-point number is rounded down to the nearest whole digit, SM601 is ON.
8. When the conversion result exceeds the range, SM602 is ON.

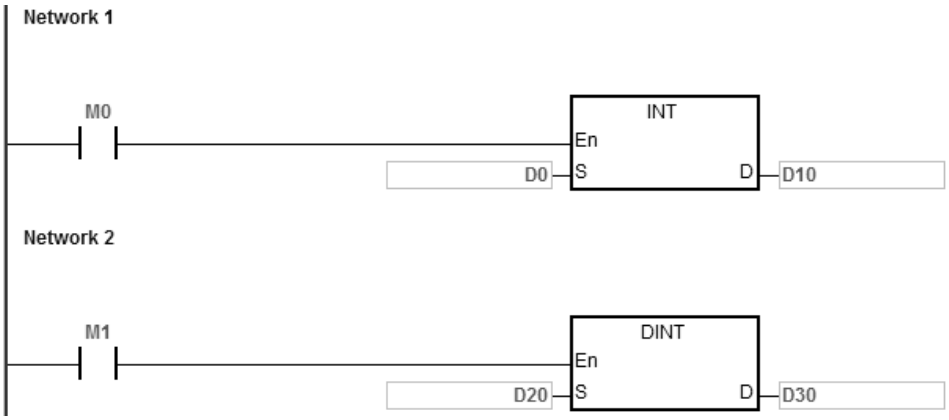
9. For the INT/IINTP instructions, the range of conversion result is between -32,768 and 32,767.

10. For the DINT/DINTP instructions, the range of conversion result is between -2,147,483,648 and 2,147,483,647.

Example

1. When M0 is ON, the instruction converts the single-precision floating-point number in (D1, D0) into a binary integer, and stores the conversion result in D10. The instruction rounds the binary floating-point number down to the nearest whole digit.

2. When M1 is ON, the instruction converts the single-precision floating-point number in (D21, D20) into a binary integer, and stores the conversion result in (D31, D30). The instruction rounds the binary floating-point number down to the nearest whole digit.



Additional remarks

If the value in **S** exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0206		MMOV	P	**S, D**	Converting a 16-bit value into a 32-bit value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●	●				●	●	
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Source device

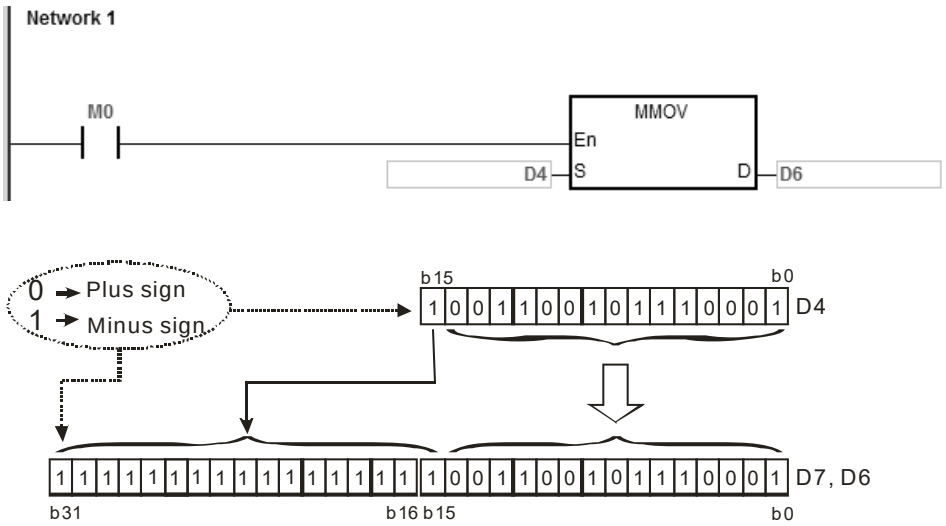
D : Conversion result

Explanation

This instruction copies the data in the 16-bit device **S** to the 32-bit device **D**, and copies the sign bit from **S** to **D**.

Example

When M0 is ON, the instruction copies the value of b15 in D4 to b15–b31 in (D7, D6), copies the values of b0–b14 to the corresponding bits in (D7, D6), and ignores the bits b15–b30. The data in (D7, D6) is a negative value (same as the source).



API	Instruction code			Operand	Function
0207		RMOV	P	**S, D**	Converting a 32-bit value into a 16-bit value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○		○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S			●				●						
D		●			●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Source device

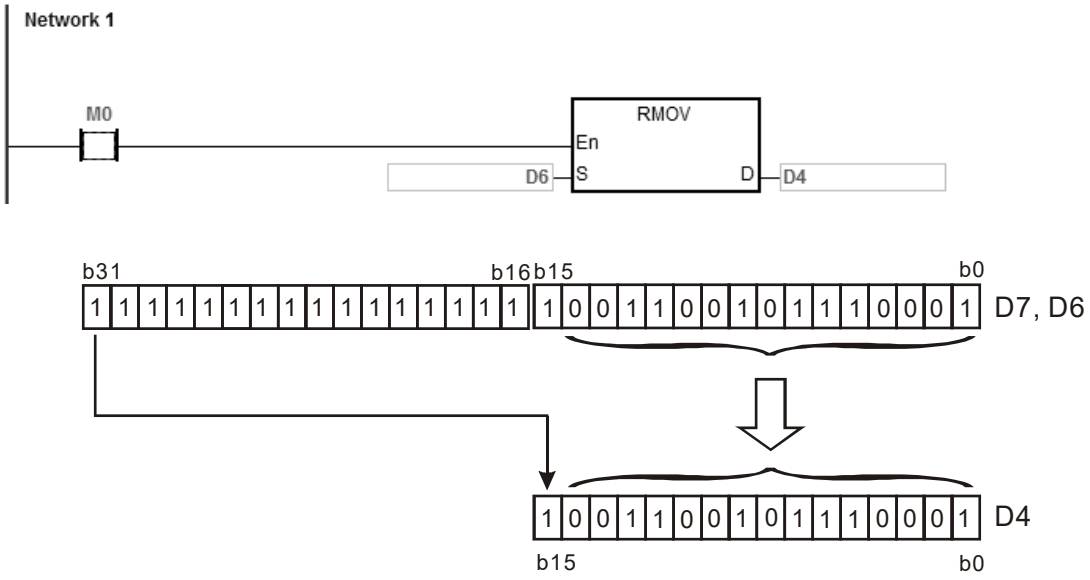
D : Conversion result

Explanation

This instruction copies the data in the 32-bit device **S** to the 16-bit device **D**, and copies the sign bit from b31 to b15 so that the value in D4 is negative (same as b31).

Example

When M0 is ON, the instruction copies the value of b31 in D7 to b15 in D4, copies the values of b0–b14 to the corresponding bits in D4, and ignores bits b15–b30. The data in (D7, D6) is a negative value (same as the source).



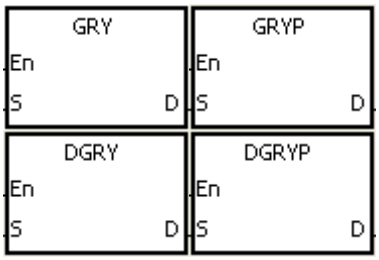
API	Instruction code			Operand	Function
0208	D	GRY	P	**S, D**	Converting a binary number into a Gray code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Conversion result

Explanation

1. This instruction converts the binary value in the device specified by **S** into a Gray code, and stores the conversion result in the device specified by **D**.

2. Only the DGRY instruction can use the 32-counter, but not the device E.

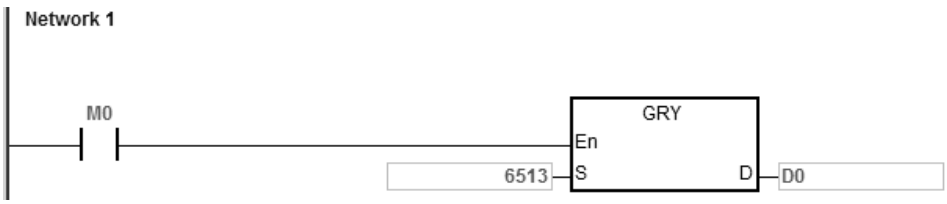
3. The value in **S** should be within the available range.

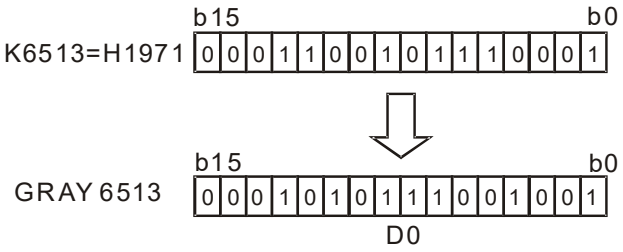
   The value in **S** in the 16-bit instruction must be between 0–32,767.

   The value in **S** in the 32-bit instruction must be between 0–2,147,483,647.

Example

When M0 is ON, the instruction converts the constant 6513 into a Gray code, and stores the conversion result in D0.





Additional remarks

If the value in **S** is less than 0, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

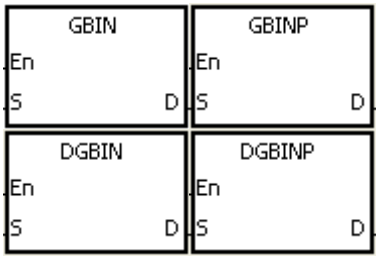
API	Instruction code			Operand	Function
0209	D	GBIN	P	**S, D**	Converting a Gray code into a binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Conversion result

Explanation

1. This instruction converts the Gray code in the device specified by **S** into the binary value, and stores the conversion result in the device specified by **D**.

2. Use this instruction to convert the Gray code in the absolute position encoder which is connected to the input terminal of the PLC to the binary value. The conversion result is stored in the specified register.

3. Only the DGBIN instruction can use the 32-counter, but not the device E.

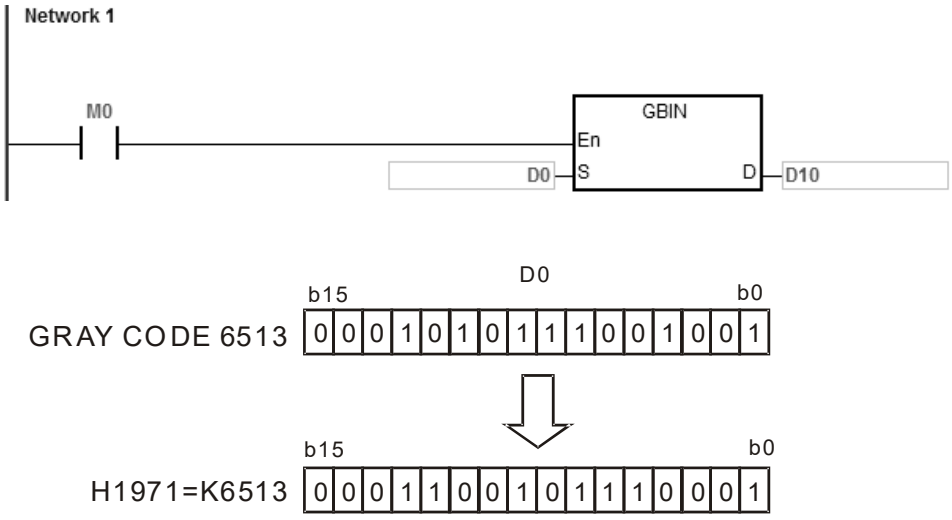
4. The value in the device **D** must be within the available range.

   The value in the device **D** in the 16-bit instruction must be between 0–32,767.

   The value in the device **D** in the 32-bit instruction must be between 0–2,147,483,647.

Example

When M0 is ON, the instruction converts the Gray code in D0 into the binary value, and stores the conversion result in D10.



Additional remarks

If the value in **S** is less than 0, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
0210	D	NEG	P	**D**	Finding the two's complement

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

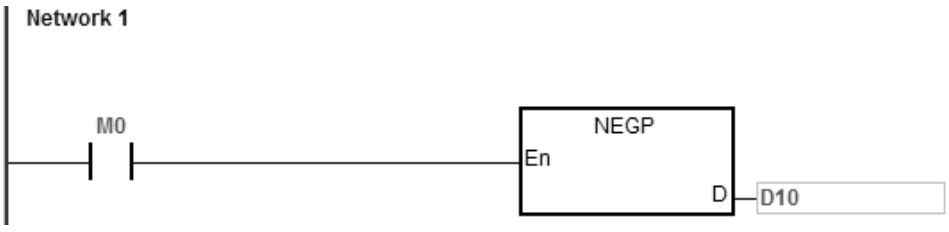
D : Device in which the two's complement is stored

Explanation

1. This instruction converts a negative binary value into the two's compliment.
2. Only the DNEG instruction can use the 32-bit counter.
3. Generally, you use the NEGP and DNEGP pulse instructions.

Example 1

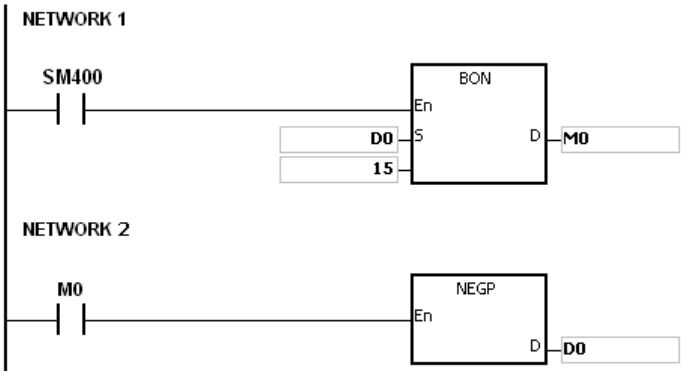
When M0 is switched from OFF to ON, this instruction inverts all bits in D0 (0 becomes 1, and 1 becomes 0), and 1 is added to the result, and then stores the final value in the original register D10.



Example 2

Finding the two's compliment of the negative value:

1. When the value of the 15th bit in D0 is 1, M0 is ON, and the value in D0 is a negative value.
2. When M0 is ON, the NEG instruction finds the two's complement of the negative value in D0 (the corresponding positive value).

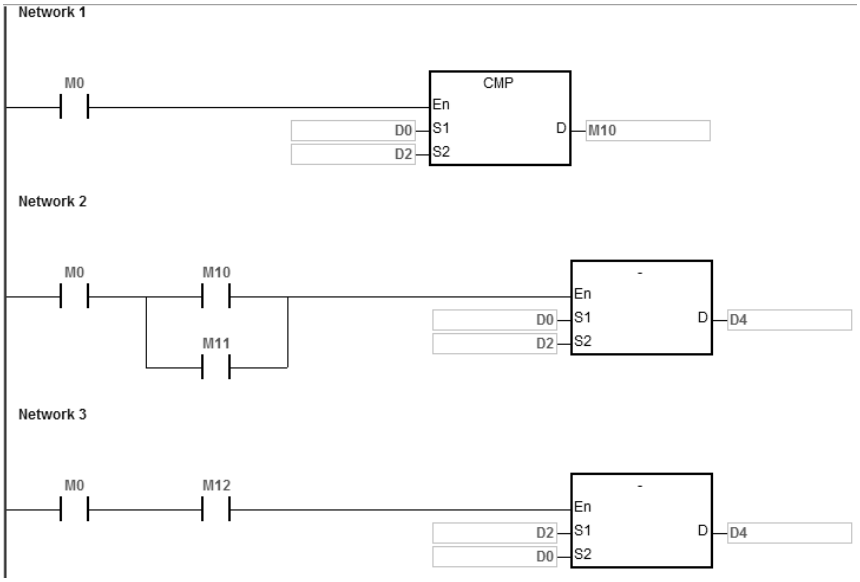


Example 3

Finding the two's compliment of the difference between two values:

When M0 is ON,

1. If the value in D0 is greater than that in D2, M10 is ON.
2. If the value in D0 is equal to that in D2, M11 is ON.
3. If the value in D0 is less than that in D2, M12 is ON.
4. The value in D4 is a positive value.



Additional remarks

Binary representation of the value and its absolute value:

1. Whether the data is a positive value or a negative value depends on the value of the highest bit in the register. If the highest bit in the register is 0, the data is a positive value. If the highest bit is 1, the data is a negative value.

2. You can convert the negative value into its absolute value with the instruction NEG.

(D0)=2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

(D0)=1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(D0)=0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

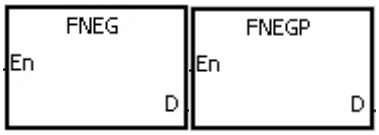
API	Instruction code			Operand	Function
0211		FNEG	P	**D**	Reversing the sign of a 32-bit floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



D : Device in which the sign of the value is reversed

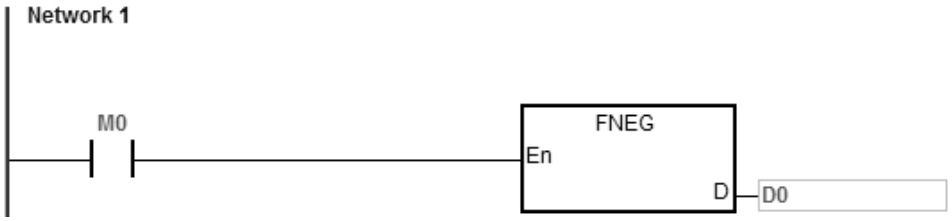
Explanation

This instruction reverses the sign of a single-precision floating-point number in **D**.

Example

Before the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000. When M0 switches from OFF to ON, the instruction reverses the sign of the single-precision floating-point number in (D1, D0). In other words, after the instruction is executed, the value in (D1, D0) is the positive value 16#2E0F9000.

Before the instruction is executed, the value in (D1, D0) is the positive value 16#2E0F9000. When M0 switches from OFF to ON, the instruction reverses the sign of the single-precision floating-point number in (D1, D0). In other words, after the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000.



API	Instruction code			Operand	Function
0212		FBCD	P	**S, D**	Converting a binary floating-point number into a decimal floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Data source

D : Conversion result

Explanation

1. This instruction converts thee single-precision floating-point number in the register specified by **S** into the decimal floating-point number, and stores the conversion result in the register specified by **D**.

2. The floating-point operation in the PLC is based on single-precision floating-point numbers. Use the FBCD instruction to convert a single-precision floating-point number into a decimal floating-point number.

3. Instruction flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

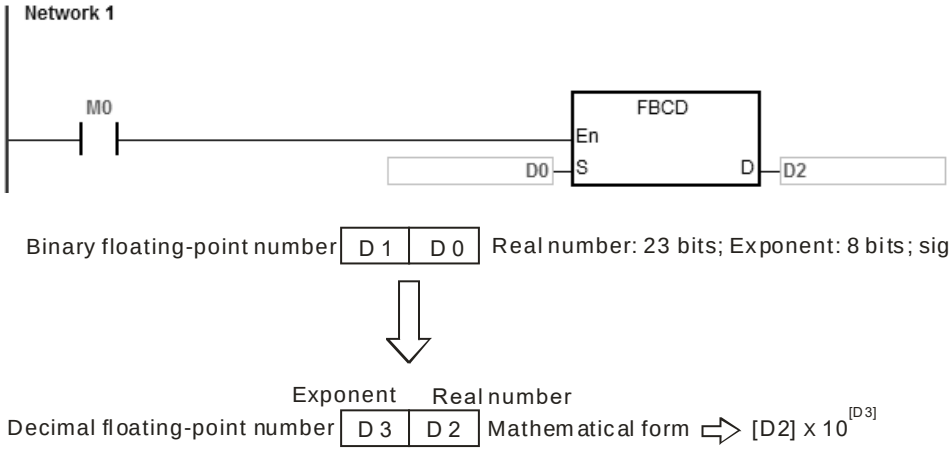
   When the absolute value of the conversion result is larger than the value that can be represented by the maximum floating-point number, SM602 is ON.

   When the absolute value of the conversion result is less than the value that can be represented by the minimum floating-point number, SM601 is ON.

   When the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the instruction converts the single-precision floating-point number in (D1, D0) into the decimal floating-point number, and stores the conversion result in (D3, D2).



Additional remarks

If the value in **S** exceeds the range of values that can be represented by the floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
0213		FBIN	P	**S, D**	Converting a decimal floating-point number into a binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		●					
D					●	●	●	●			●					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Data source

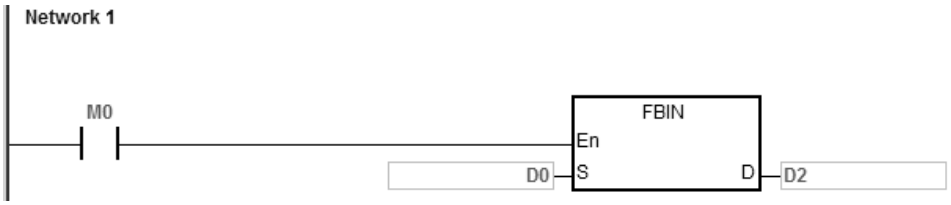
D : Conversion result

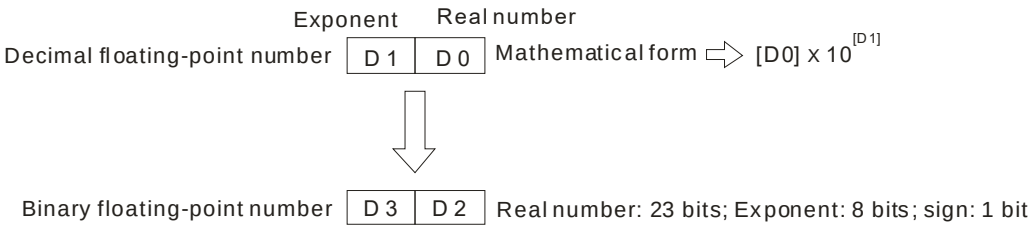
Explanation

1. This instruction converts the decimal floating-point number in the register specified by **S** into the single-precision floating-point number, and stores the conversion result in the register specified by **D**.
2. Suppose the value in **S** is 1234, and the value in **S**+1 is 3. The instruction converts the value in **S** into $1.234x10^6$.
3. The value in **D** should be a single-precision floating-point number, and the values in **S** and **S**+1 represent the decimal real number and the decimal exponent respectively.
4. Use the FBIN instruction to convert a decimal floating-point number into a single-precision floating-point number.
5. The real number of decimal floating-point numbers are from -9,999 to +9,999, the exponents of decimal floating-point numbers are from -41 to +35. The practical range of decimal floating-point numbers in the PLC is between ±1175×10-41 and ±3402×10+35. When the operation result is zero, SM600 is ON.

Example 1

1. When M0 is ON, the instruction converts the decimal floating-point number in the register in (D1, D0) into the single-precision floating-point number, and stores the conversion result is stored in (D3, D2).

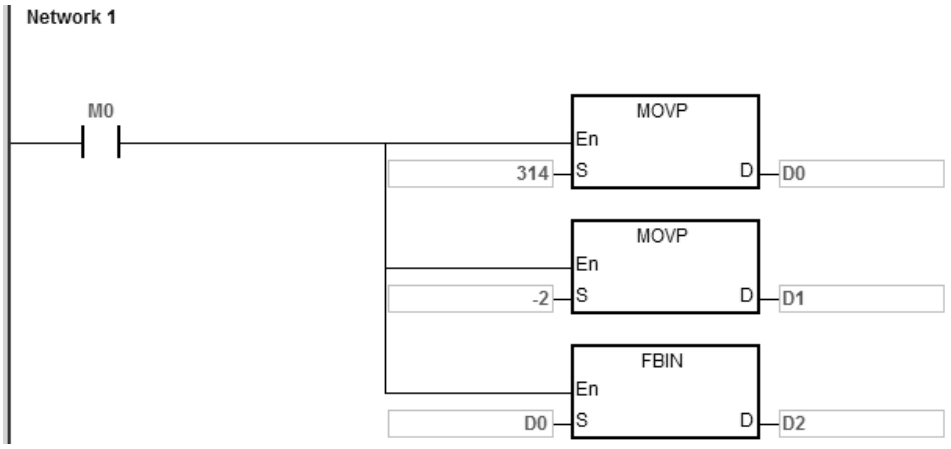




Example 2

1. Before the floating-point operation is performed, use the FLT instruction to convert the binary integer into a single-precision floating-point number. Make sure the value to be converted is a binary integer before conversion. You can use the FBIN instruction to convert the floating-point number into the single-precision floating-point number.

2. When M0 is ON, K314 and K-2 are moved to D0 and D1 respectively, and then FBIN combines them into the decimal floating-point number ($3.14=314\times10^{-2}$).



Additional remarks

If the real number part of the decimal floating-point number in **S** is not between -9,999 to +9,999, or if the exponent of the decimal floating-point number in **S** is not between -41 to +35, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

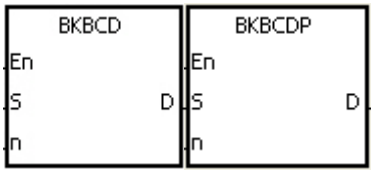
API	Instruction code			Operand	Function
0214		BKBCD	P	**S, n, D**	Converting binary numbers in blocks into binary-coded decimal numbers in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
n					●	●		●	●				○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Data source

n : Data length

D : Conversion result

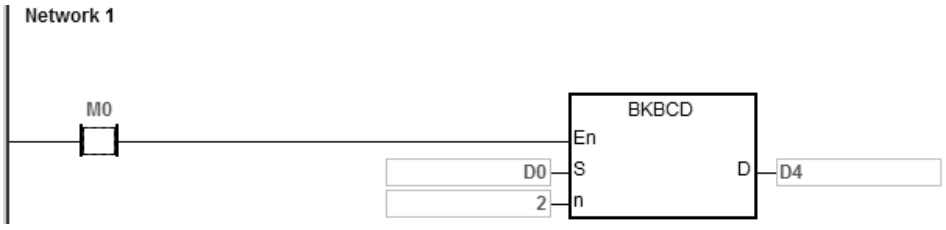
Explanation

1. The instruction converts **n** pieces of data (the binary values) starting from **S** into the binary-coded decimal values, and stores the conversion results in **D**.

2. The operand **n** must be between 1–256.

Example

When M0 is ON, the instruction converts the binary values in D0 and D1 into the binary-coded decimal values, and stores the conversion results in D4 and D5.



Additional remarks

1. If **n** is less than 1, or larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If the devices specified by **S**+**n**-1 and **D**+**n**-1 exceed the range of possible devices, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the conversion result is not between 0–9,999, the instruction is not executed, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not in between 0–9.

4. If **S**–**S**+**n**-1 overlaps **D**–**D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

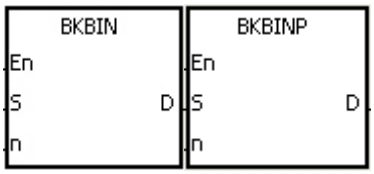
API	Instruction code			Operand	Function
0215		BKBIN	P	**S, n, D**	Converting binary numbers in blocks into binary-coded decimal numbers in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
n					●	●		●	●				○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Data source

n : Data length

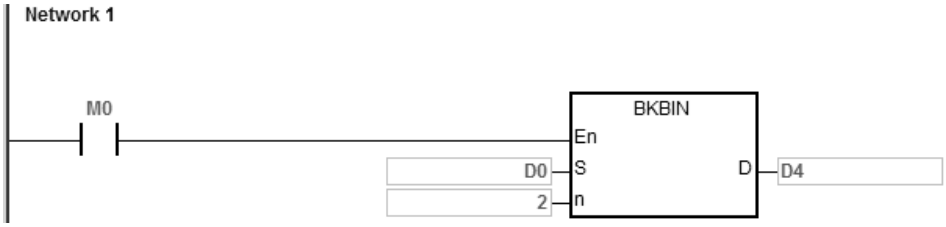
D : Conversion result

Explanation

1. The instruction converts **n** pieces of data (the binary-coded decimal values) starting from **S** into the binary values, and stores the conversion results in **D**.
2. The binary-coded decimal value in **S** must be between 0–9,999.
3. The operand **n** must be between 1 and 256.

Example

When M0 is ON, the instruction converts the binary-code decimal values in D0 and D1 into the binary values, and stores the conversion results in D4 and D5.



Additional remarks

1. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If the devices specified by **S**+**n**-1 and **D**+**n**-1 exceed the range of possible devices, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the data in **S** is not a binary-coded decimal, the instruction is not executed, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not between 0–9.

4. If **S**–**S**+**n**-1 overlap **D**–**D**+**n**-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

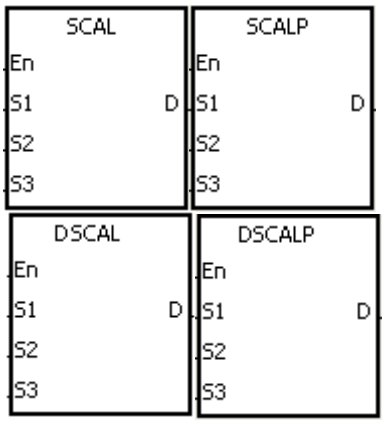
API	Instruction code			Operand	Function
0216		SCAL	P	**S_1, S_2, S_3, D**	Finding a scaled value (point-slope)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○	○	○	○		
S_2					●	●		●	●		○	○	○	○		
S_3					●	●		●	●		○	○	○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source

S_2 : Slope

S_3 : Offset

D : Destination device

Explanation

1. This instruction finds a scaled linear value for the number in **S1** using the slope of the line in **S_2** and the offset in **S_3** to define the scaling factor, and stores the result in **D**.

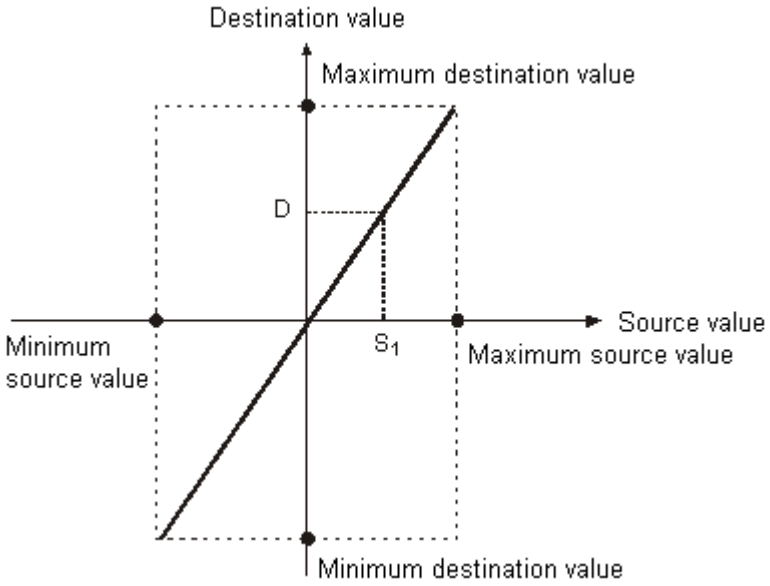
2. The operation equation in the instruction is **D**=(**S_1**×**S_2**)÷1,000+**S_3**

3. To get the values in **S_2** and **S_3**, use the slope equation and the offset equation below first, and then round off the results to the nearest whole digit. Enter the final 16-bit values into **S_2** and **S_3**.

   The slope equation: **S_2**=[(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value)]×1,000

   The offset equation: **S_3**=Minimum destination value–Minimum source value×**S_2**÷1,000

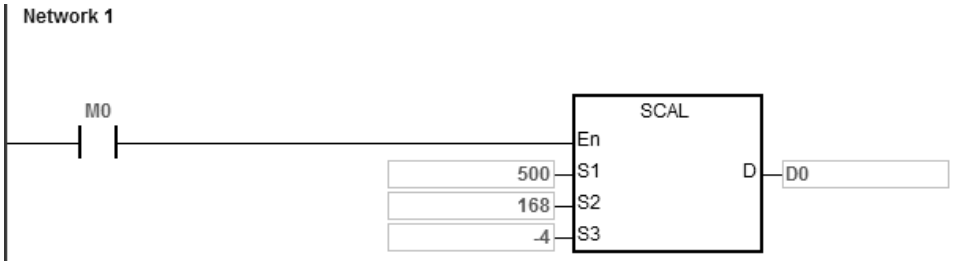
The output curve is shown below:

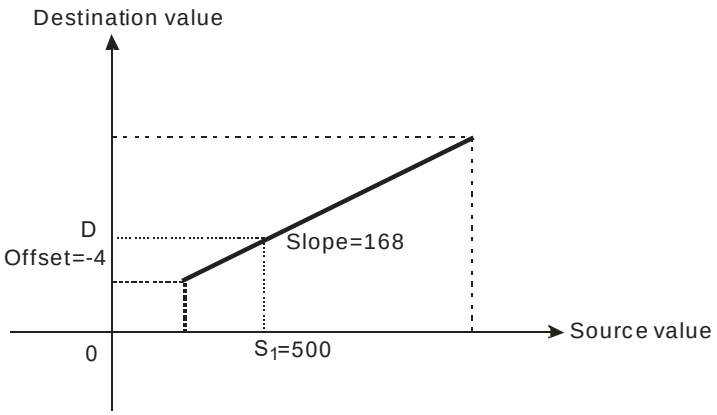


Example 1

1. Suppose the values in **S_1**, **S_2**, and **S_3** are 500, 168, and -4 respectively. When M0 is ON, the SCAL instruction calculates the scaled value, and stores the scaled value in D0.

2. For equation: D0=(500×168)÷1,000+(-4)=80

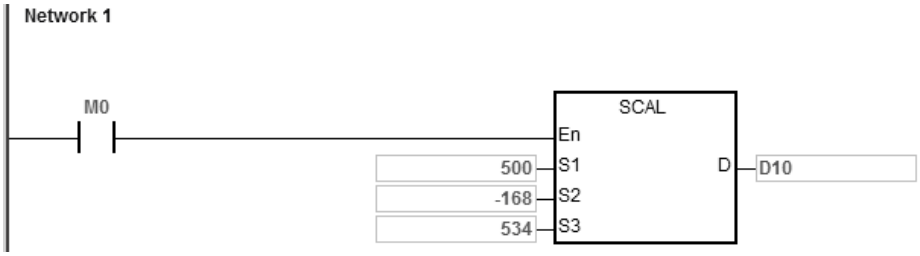


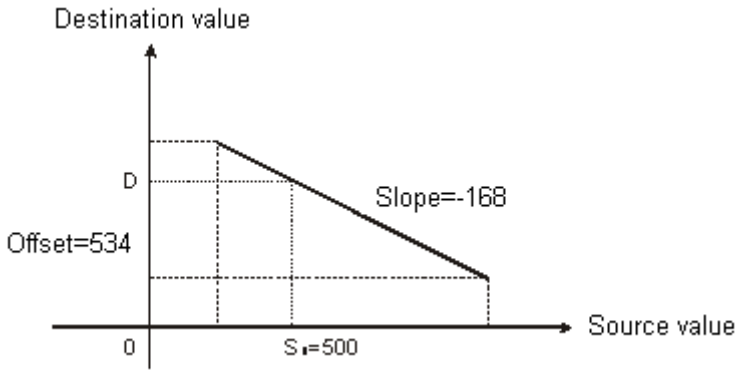


Example 2

1. Suppose the values in **S_1**, **S_2**, and **S_3** are 500, -168, and 534 respectively. When M0 is ON, the SCAL instruction calculates the scaled value, and stores the scaled value in D10.

2. For the equation: D10=(500×-168)÷1,000+534=450





6

Additional remarks

1. You must know the slope and the offset to use SCAL. If the slope and the offset are unknown, you can use the SCLP instruction.

2. When the 16-bit instruction is performed, the value entered into **S_2** must be between –32,768 to 32,767. If the value in **S_2** exceeds the range, use the SCLP instruction.

3. When the 32-bit instruction is performed, the value entered into **S_2** must be between -2,147,483,648 to 2,147,483,647. If the value in **S_2** exceeds the range, use the SCLP instruction.

4. When you use the slope equation, note that the maximum source value should be larger than the minimum source value. However, the maximum destination value is not necessarily larger than the minimum destination value.

5. For the 16-bit instruction, if the value in **D** is larger than 32,767, the value stored in **D** is 32,767. If the value in **D** is less than -32,768, the value stored in **D** is -32,768.

6. When the 32-bit instruction is performed, if the value in **D** is larger than 2,147,483,647, the value stored in **D** will be 2,147,483,6477. If the value in **D** is less than -2,147,483,648, the value stored in **D** will be -2,147,483,648.

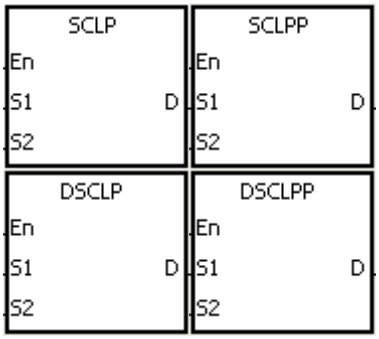
API	Instruction code			Operand	Function
0217	D	SCLP	P	**S_1 · S_2 · S_3 · D**	Finding a scaled value (two points)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		○
S_2					●	●	●	●	●							
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●		●				
S_2		●	●		●	●	●		●				
D		●	●		●	●	●		●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source

S_2 : Parameter

D : Destination device

Explanation

1. This instruction finds a scaled linear value for the value in **S_1** using two points in **S_2** to define the scaling factor, and stores the result in **D**.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

3. The following table list the constant usage for the operand **S_1**

Constant	16-bit instruction	32-bit instruction	
		SM685 ON	SM685 OFF
Constant	○	X	○
Hexadecimal	○	X	○
Floating number	X	○	X

The flag SM685 (whether to use floating point operation or not) can only be used for 32-bit instructions.

4. The operand **S_2** used in the 16-bit instruction is set as shown in the following table.

Device number	Parameter	Setting range
S_2	Maximum source value	-32,768 to 32,767
S_2+1	Minimum source value	-32,768 to 32,767
S_2+2	Maximum destination value	-32,768 to 32,767
S_2+3	Minimum destination value	-32,768 to 32,767

5. The operand **S_2** used in the 16-bit instruction occupies four devices.

6. The operand **S_2** used in the 32-bit instruction is set as shown in the following table.

Device number	Parameter	Setting range	
		Integer	Floating-point number
S_2、**S_2**+1	Maximum source value	-2,147,483,648 to 2,147,483,647	The range of 32-bit floating-point numbers
S_2+2、3	Minimum source value		
S_2+4、5	Maximum destination value		
S_2+6、7	Minimum destination value		

7. The operand **S_2** used in the 32-bit instruction occupies eight devices.

8. If the values in the 32-bit instruction are floating-point numbers, set SM658 to ON. If the values are decimal integers, set SM685 to OFF.

9. The operation equation in the instruction is:
D = [(**S_1**–Minimum source value)×(Maximum destination value–Minimum destination value)]÷(Maximum source value)+Minimum destination value

10. The operational relation between the source value and the destination value is:

    y = kx+b

    y=Destination value (D)

    k=Slope=(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value)

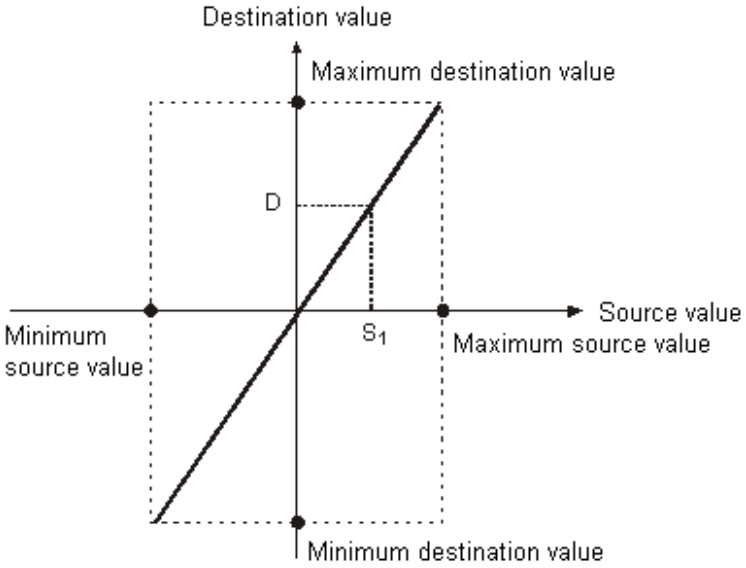
    x=Source value (**S_1**)

    b=Offset =Minimum destination value–Minimum source value×Slope

The parameters above are substituted for y, k, x, and b in the equation y = kx+b to get the operation equation as follows:

y=kx+b=D=k**S**$_1$+b=Slope×**S**$_1$ + Offset = Slope×**S**$_1$ + Minimum destination value–Minimum source value×Slope=Slope×(**S**$_1$–Minimum source value) + Minimum destination value =(**S**$_1$–Minimum source value)×(Maximum destination value–Minimum destination value)÷(Maximum source value–Minimum source value) + Minimum destination value

11. If **S**$_1$ is larger than the maximum source value, the maximum source value is the value in **S**$_1$. If **S**$_1$ is less than the minimum source value, the minimum source value is the value in **S**$_1$. The output curve is shown below.

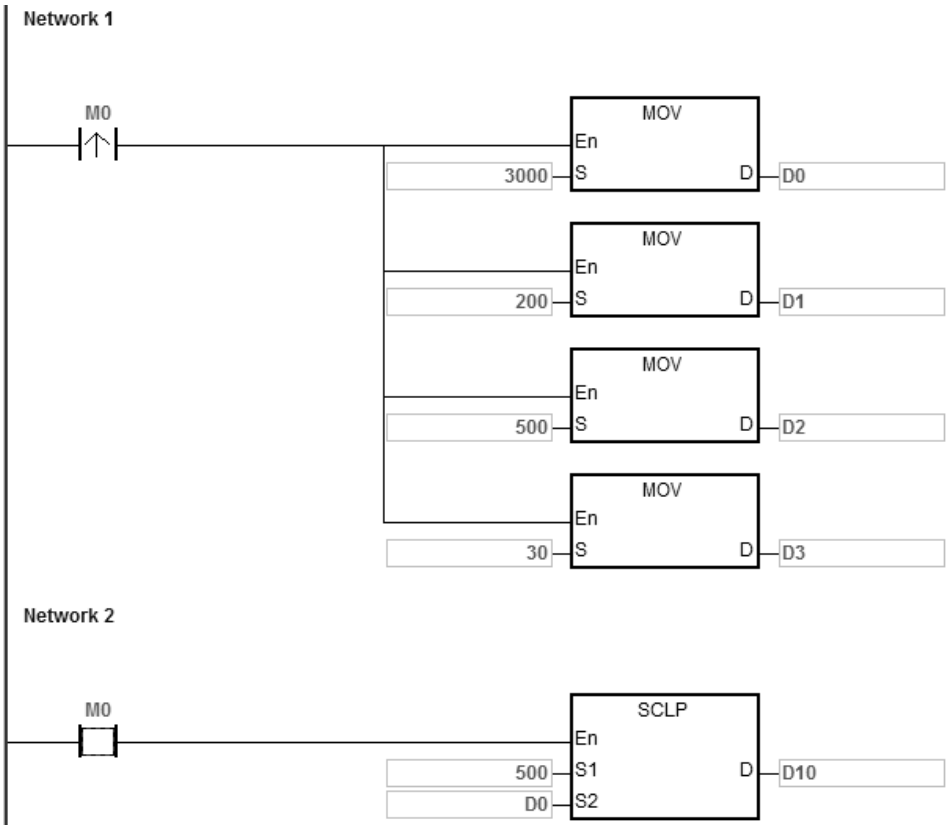


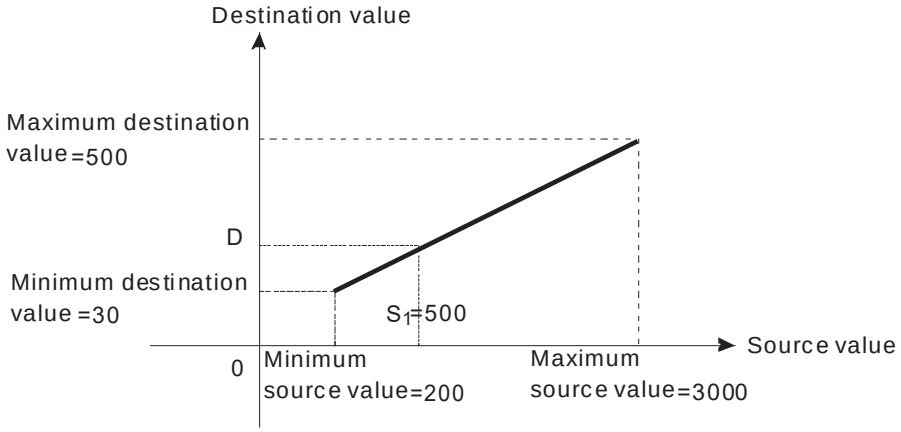
Example 1

1. Suppose the value in **S**$_1$ is 500, the maximum source value in D0 is 3,000, the minimum source value in D1 is 200, the maximum destination value in D2 is 500, and the minimum destination value in D3 is 30. When M0 is ON, the SCLP instruction calculates the scale value and stores it in D10.

2. The operation equation: D10=[(500–200)×(500–30)]÷(3,000–200)+30=80.35

   80.35 is rounded off to the nearest whole digit, and becomes 80. 80 is stored in D10.



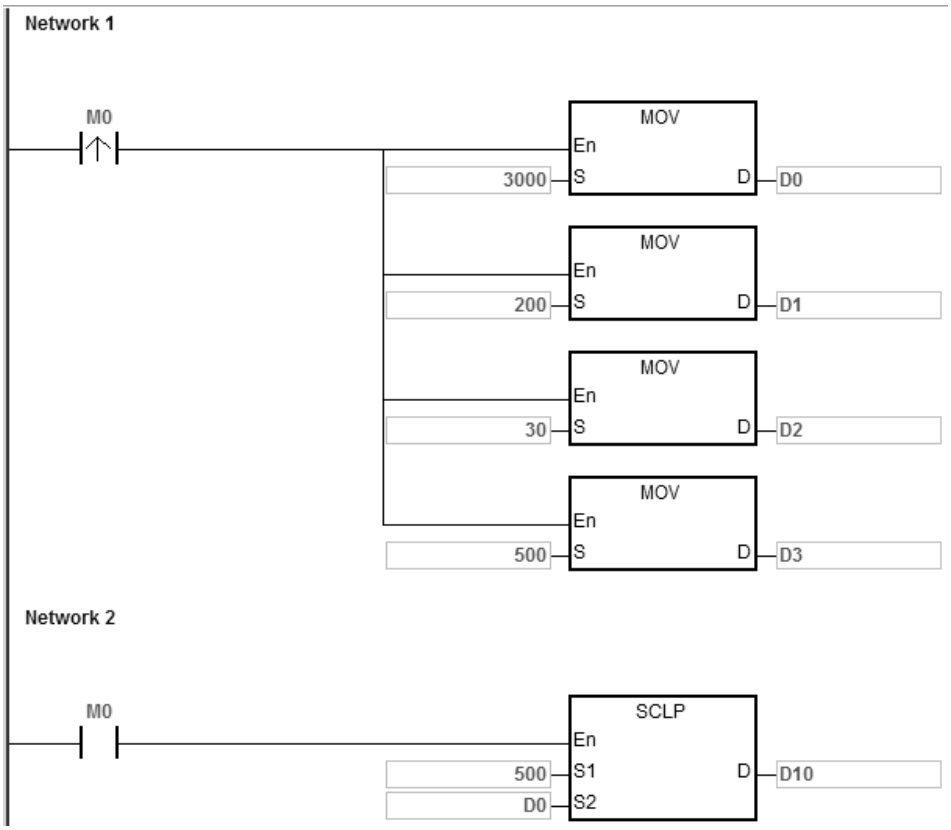


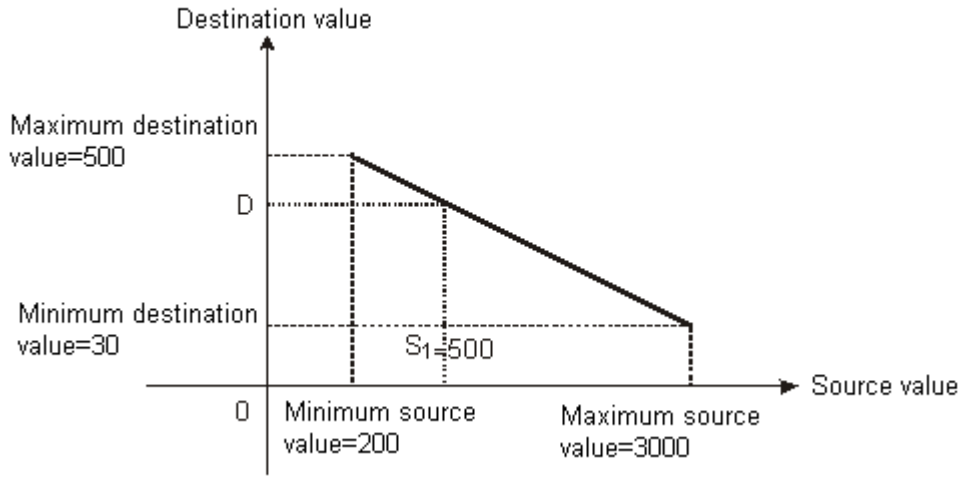
Example 2

1. Suppose the value in **S_1** is 500, the maximum source value in D0 is 3,000, the minimum source value in D1 is 200, the maximum destination value in D2 is 30, and the minimum destination value in D3 is 500. When M0 is ON, the SCLP instruction calculates the scales value, and stores it in D10.

2. The operation equation: D10=[(500–200)×(30–500)]÷(3,000–200)+500=449.64

   449.64 is rounded off to the nearest whole digit, and becomes 450. 450 is stored in D10.



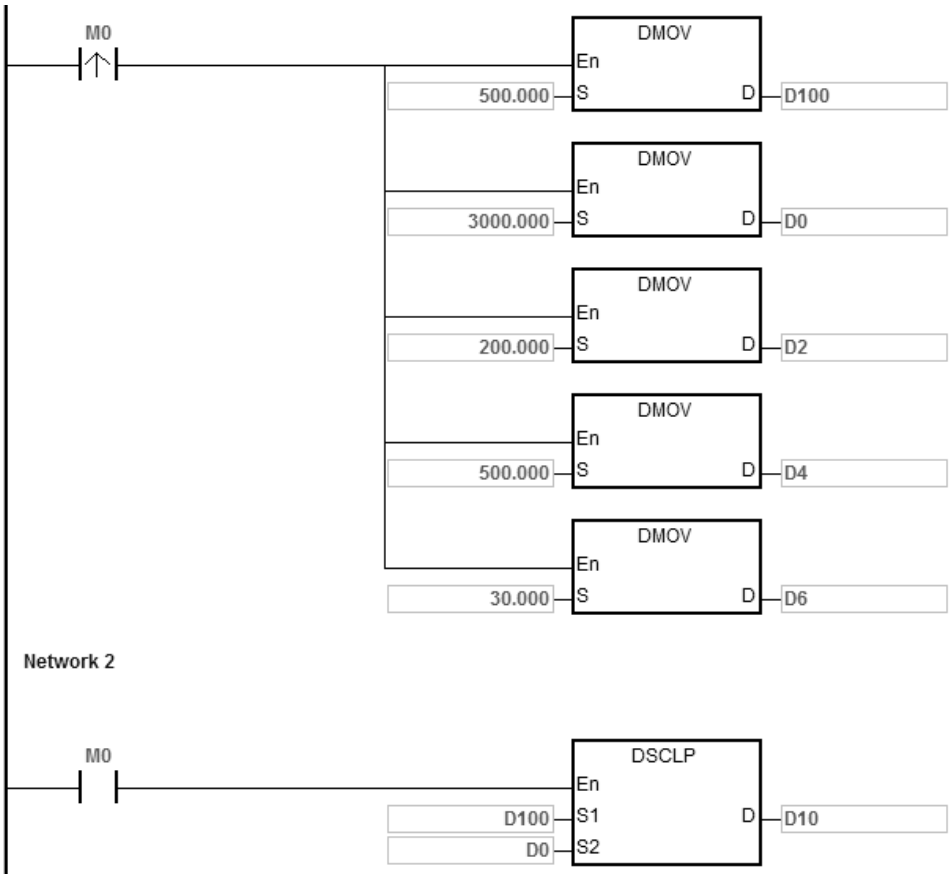


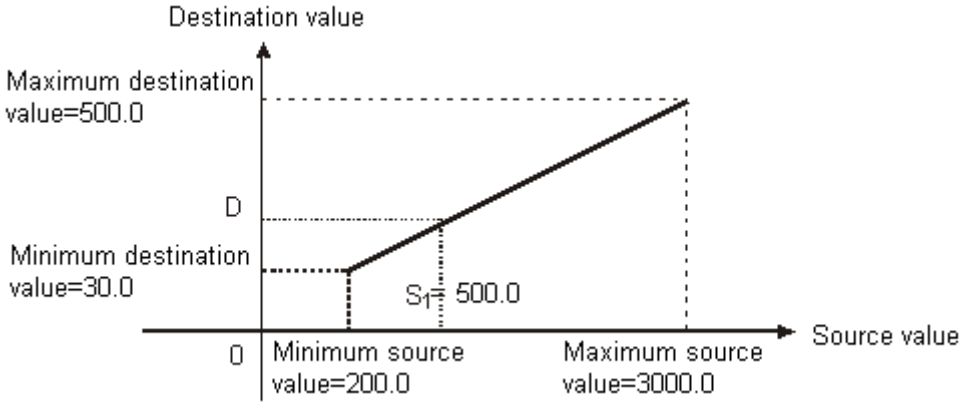
Example 3

1. Suppose the value in **S_1** is 500.0, the maximum source value in D0 is 3000.0, the minimum source value in D2 is 200.0, the maximum destination value in D4 is 500.0, and the minimum destination value in D6 is 30.0. When M0 is ON, SM685 is set to ON, the instruction DSCLP calculates the scale value and stores it in D10.

2. The operation equation: D10=[(500.0–200.0)×(500.0–30.0)]÷(3000.0–200.0)+30.0=80.35

80.35 is rounded off to the nearest whole digit, and becomes 80.0. 80.0 is stored in D10.





Additional remarks

1. The value in **S_1** for 16-bit instructions must be between the minimum source value and the maximum source value; that is, between -32,768 to 32,767. If the value exceeds the boundary value, the calculation uses the boundary value.

2. The integer in **S_1** for 32-bit instructions must be between the minimum source value and the maximum source value; that is, between -2,147,483,648 to 2,147,483,647. If the integer exceeds the boundary value, the calculation uses the boundary value.

3. The floating-point number in **S_1** for 32-bit instructions must be between the minimum source value and the maximum source value; that is, within the range of floating-point numbers. If the floating-point number exceeds the boundary value, the calculation uses the boundary value.

4. Note that the maximum source value must be larger than the minimum source value. However, the maximum destination value is not necessarily larger than the minimum destination value.

5. When the maximum source value is the same as the minimum source value, the instruction is be executed, SM0 is ON and the error code in SR0 is 16#2012.

6. If you declare **S_2** for a 16-bit instruction in ISPSoft, the data type is ARRAY [4] of WORD.

7. If you declare **S_2** for a 32-bit instruction in ISPSoft, the data type is ARRAY [4] of DWORD.

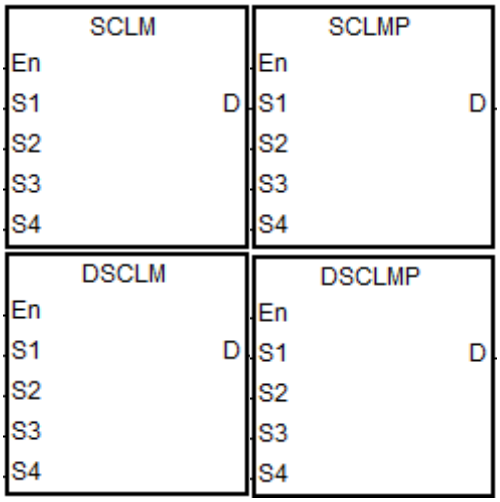
API	Instruction			Operand	Description
0222	D	SCLM	P	**S_1, S_2, S_3, S_4, D**	Multi-point area ratio operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●								
S_2					●	●	●	●					○	○		
S_3					●	●	●	●	●							
S_4					●	●	●	●	●							
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●		●				
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
S_4		●	●		●	●	●						
D		●	●		●	●	●		●				

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source

S_2 : Number of multi-point areas

S_3 : Comparison value in a multi-point area

S_4 : Conversion reference value

D : Operation result

Explanation

1. Only the 32-bit instruction can use HC devices but not E devices.

2. See the following table about data types that the operands **S_1**, **S_3** and **S_4** correspond to. (○ represents 'Usable'. X represents 'Unusable'.)

Constant	16-bit instruction	32-bit instruction	
		SM685 ON	SM685 OFF
K	○	X	○
16#	○	X	○
F	X	○	X

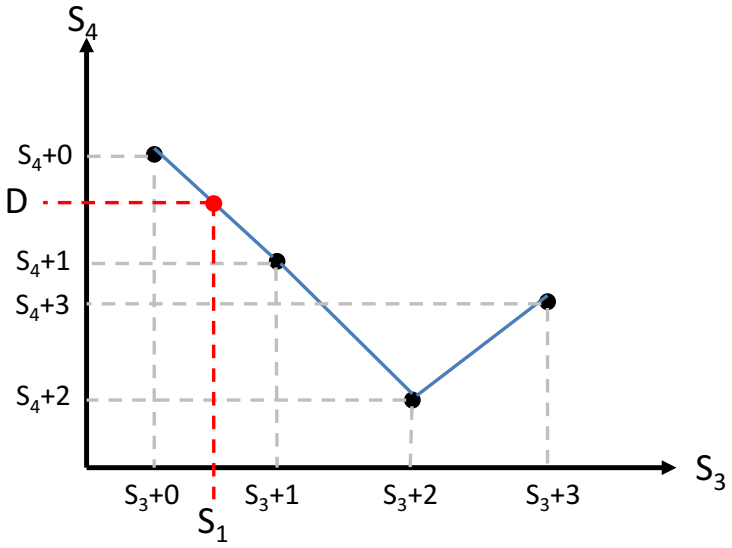
Note: SM685=ON (the floating point number operation) works for the 32-bit instruction only.

3. **S_1** is the data source. **S_2** is the number of multi-point areas and the value should be between 2 and 50. If the value exceeds the range, the instruction will be executed automatically at the minimum value or maximum value. **S_3** is a setting value for comparison in a multi-point area. **S_4** is a conversion reference value that a multi-point area comparison value corresponds to, e.g. the number of areas, **S_2** is 10. Then **S_3** ~ **S_3**+9 are comparison values in 10 areas. **S_4** ~ **S_4**+9 are 10 corresponding conversion reference values.

4. The comparison order for multi-point areas is 0, 1, 2 …**S_2**–1. The comparison rule is **S_1** >= **S_3**+0 and **S_1** < **S_3**+1. If **S_1** value does not belong to an area, the comparison will move on to the next area. For example, **S_1** >= **S_3**+1 and **S_1** < **S_3**+2, the comparison keeps going until the number of comparison times reaches **S_2** – 1.

5. The instruction compares areas in the order from small to large. Please design the value comparison of **S_3** area in the order from small to large as well.

6. See the figure below for the conversion of multi-point area values. (Set the number of areas, **S_2** to 4.)



7. If **S_1** value is between **S_3**+0 and **S_3**+1, the conversion formula: **D** = （ (**S_1** – **S_3**+0) x (**S_4**+1 – **S_4**+0) / (**S_3**+1 – **S_3**+0) ） + **S_4**+0.

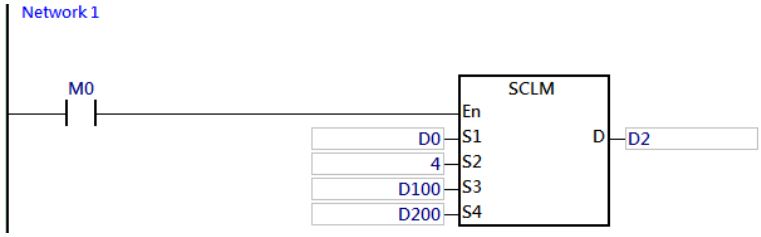
8. If **S_1** value does not belong to any specified area, the execution result in D is explained as below.

   If **S_1** value > the last specified area, D will store the last conversion reference value of S4, e.g. if **S_1** value > **S_3**+3 value in the figure above, **D**=**S_4**+3.

If **S1** value < the first specified area, D will store the first conversion reference value of S4, e.g. if **S1** value < **S3**+0 value in the figure above, **D**=**S4**+0.

9. If **S3** and **S4** of the16-bit instruction are declared on ISPSoft, the data type is ARRAY [**S2**] of WORD.
10. If **S3** and **S4** of the 32-bit instruction are declared on ISPSoft, the data type is ARRAY [**S2**] of DWORD (SM685=OFF) or ARRAY [**S2**] of REAL (SM685=ON).

Example



The comparison values of S3 for multi-point areas are given as follows.

Device	D100	D101	D102	D103
Content	100	200	300	400

The corresponding conversion reference values of **S4** are given as follows.

Device	D200	D201	D202	D203
Content	4000	3000	1500	2000

Here is the explanation about the value in D2 obtained through a conversion based on the data resource D0.

Set D0=10,
Since D0<D100 (in the first area), D2=D200=4000 (the first conversion reference value)

Set D0=K150,
D0 value is in between (D100, D101) = (100, 200) and the corresponding reference value is (D200, D201) = (4000, 3000)
Therefore,
D2= (150-100)* (3000-4000) / (200-100) +4000=3500

Set D0=450
Since D0>D103 (in the last area), D2=D203=2000 (the last conversion reference value)

Set D0=K250
D0 value is between (D101, D102) = (200, 300) and the corresponding reference value is (D201, D202) = (3000, 1500).
Therefore,
D2= (250-200) *(1500-3000) / (300-200) +3000=2250

Set D0=K350
D0 is between (D102, D103) = (300, 400) and the corresponding reference value is (D202, D203) = (1500, 2000)
Therefore,
D2= (350-300) * (2000-1500) / (400-300) +1500=1750

6.4 Data Transfer Instructions

6.4.1 List of Data Transfer Instructions

The following table lists the Data Transfer instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0300	MOV	DMOV	✓	Transferring data
0302	$MOV	–	✓	Transferring a string
0303	CML	DCML	✓	Inverting data
0304	BMOV	DBMOV	✓	Transferring data in blocks
0305	NMOV	DNMOV	✓	Transferring data to multiple devices
0306	XCH	DXCH	✓	Exchanging data
0307	BXCH	–	✓	Exchanging data in blocks
0308	SWAP	DSWAP	✓	Exchanging the high byte with the low byte
0309	SMOV	–	✓	Transferring digits in blocks
0310	MOVB	–	✓	Transferring bits in blocks

6.4.2 Explanation of Data Transfer Instructions

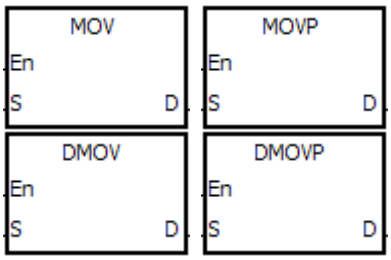
API	Instruction code			Operand	Function
0300	D	MOV	P	**S, D**	Transferring data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		○
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●		●		●	●	
D		●	●		●	●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Data source

D : Data destination

Explanation

1. This instruction transfers the data in **S** to **D**.
2. You must use a 32-bit instruction when the data in **S** is a floating-point number.
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

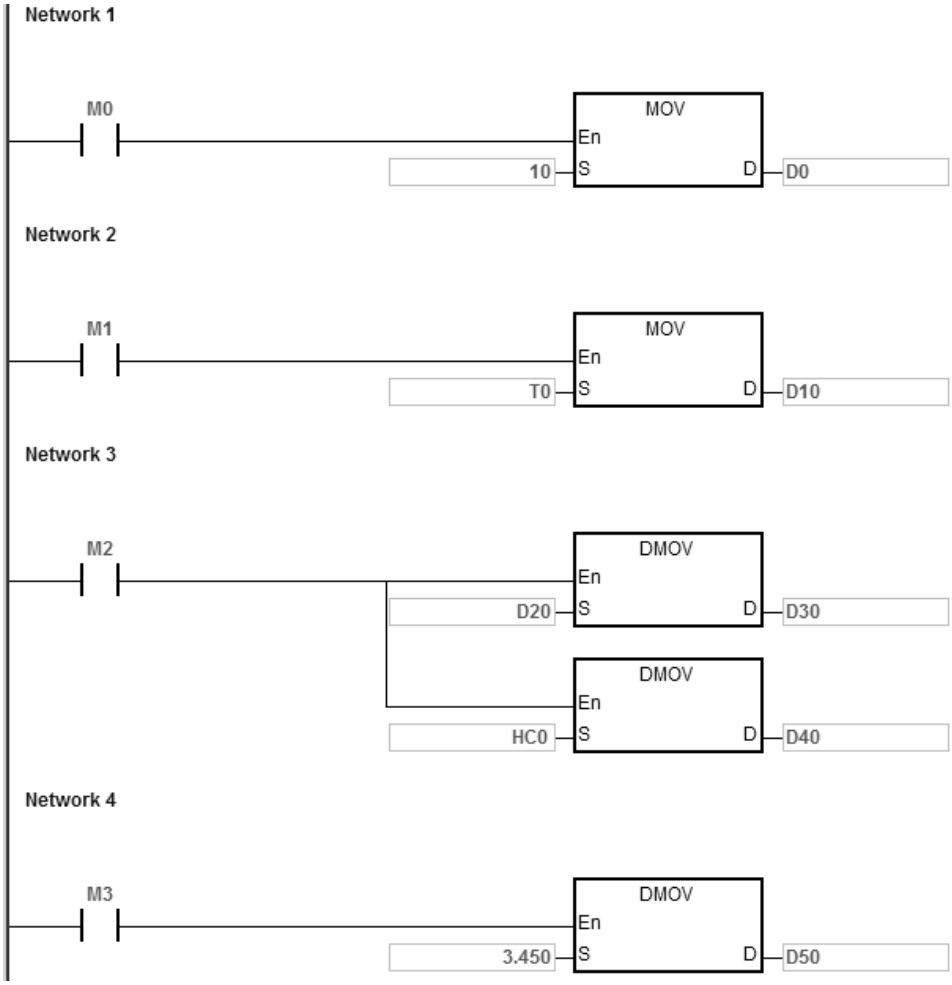
1. To transfer 16-bit data, use MOV.
   - When M0 is OFF, the data in D0 is unchanged. When M0 is ON, the instruction transfers 10 to the data register D0.
   - When M1 is OFF, the data in D10 is unchanged. When M1 is ON, the instruction transfers the current value of T0 to the data register D10.

2. For 32-bit data, use DMOV.

   - When M2 is OFF, the data in (D31, D30) and (D41, D40) is unchanged. When M2 is ON, the instruction transfers the current value in (D21, D20) to (D31, D30), and transfers the current value of HC0 to (D41, D40).

3. For floating-point numbers, use DMOV.

   - When M3 is OFF, the data in (D51, D50) is unchanged. When M3 is ON, the instruction converts the floating-point number 3.450 into a binary floating-point number, and transfers the conversion result is to (D51, D50).



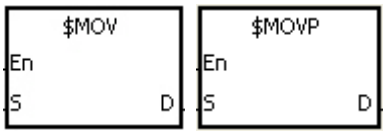
API	Instruction code			Operand	Function
0302		$MOV	P	**S, D**	Transferring a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													●
D													●

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Data source

D : Data destination

Explanation

1. This instruction transfers the string in **S** to **D**, and adds the code 16#00 to the end of the string.
2. When the operand **S** is not a string, the instruction adds the code 16#00 to the end of the data transferred.
3. When the ending code16#00 cannot be found in **S** for 256 characters in a row or even beyond the device range, the instruction is not executed; SM0 is ON and the error code in SR0 is 16#200E.
4. When the operand **S** is not a string and the instruction is executed, the string starting with the data in the device specified by **S** (including 16#00) is transferred to **D**. When the instruction is not executed, the data in **D** is unchanged.
5. If **D** is not sufficient to contain the string composed of the values in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. Suppose the operand **S** is not a string. When the instruction is executed and the first character in **S** is the code 16#00, 16#00 is still transferred to **D**.

7. When 16#00 appears in the low byte, the execution of the instruction is as follows.

Before the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#35	16#34
S+3	16#30	16#00

⇒

	B15~b8	b7~b0
D	16#38	16#39
D+1	16#36	16#37
D+2	16#34	16#35
D+3	16#32	16#33

After the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#35	16#34
S+3	16#30	16#00

16#30 in the high byte is not transferred.

⇒

	b15~b8	b7~b0
D	16#31	16#30
D+1	16#33	16#32
D+2	16#35	16#34
D+3	16#00	16#00

16#32 in the high byte turns into 16#00.

8. When 16#00 appears in the high byte, the execution of the instruction is as follows. The transfer stops when the code 16#00, leaving the remainder of **D** unchanged.

Before the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#00	16#34
S+3	16#37	16#36

⇒

	b15~b8	b7~b0
D	16#38	16#39
D+1	16#36	16#37
D+2	16#34	16#35
D+3	16#32	16#33

After the instruction is executed:

	b15~b8	b7~b0
S	16#31	16#30
S+1	16#33	16#32
S+2	16#00	16#34
S+3	16#37	16#36

⇒

	b15~b8	b7~b0
D	16#31	16#30
D+1	16#33	16#32
D+2	16#00	16#34
D+3	16#32	16#33

9. When **S** overlaps **D** and the device number of **S** is less than the device number of **D**, the transfer of the data to **D** starts form the ending code 16#00.

Before the instruction is executed:

	b15~b8	b7~b0
D0	16#31	16#30
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00

⇒

	b15~b8	b7~b0
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00
D4	16#38	16#37

After the instruction is executed:

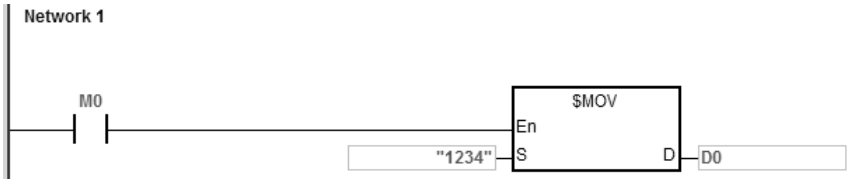
	b15~b8	b7~b0
D0	16#31	16#30
D1	16#33	16#32
D2	16#35	16#34
D3	16#30	16#00

⇒

	b15~b8	b7~b0
D1	16#31	16#30
D2	16#33	16#32
D3	16#35	16#34
D4	16#00	16#00

Example 1

Suppose the data in **S** is the string “1234” (even number of bytes). When M0 is enabled, the data 1234 and the ending code 16#00 is transferred to D0–D3 and 16#00 is added to the high byte in **D**, as follows.



The operand **S**:

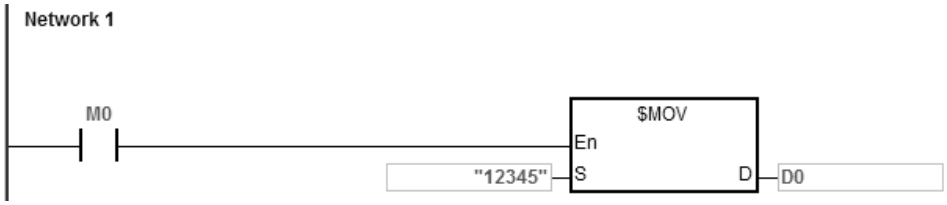
String	‘1’	‘2’	‘3’	‘4’
Hexadecimal value	16#31	16#32	16#33	16#34

After the instruction is executed, the data in **D** is as follows.

Device	High byte	Low byte	Note
D0	16#32	16#31	‘1’=16#31; ‘2’=16#32
D1	16#34	16#33	‘3’=16#33; ‘4’=16#34
D2	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D3	Unchanged	Unchanged	

Example 2

Suppose the data in **S** is the string “12345” (odd number of bytes). When M0 is enabled, the data 12345 is transferred to D0–D3 as follows.



The operand **S**:

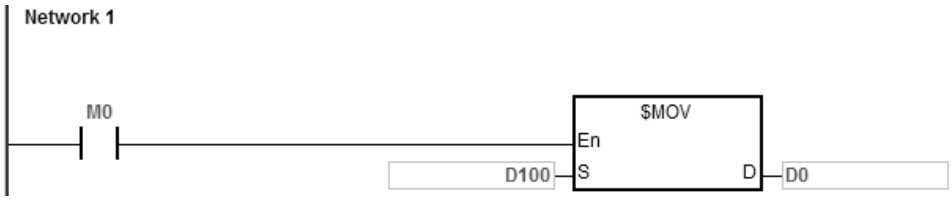
String	‘1’	‘2’	‘3’	‘4’	‘5’
Hexadecimal value	16#31	16#32	16#33	16#34	16#35

After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D0	16#32	16#31	‘1’=16#31; ‘2’=16#32
D1	16#34	16#33	‘3’=16#33; ‘4’=16#34
D2	16#00	16#35	The ending code 16#00 is in the high byte.
D3	Unchanged	Unchanged	

Example 3

When the data in **S** is not a string and the ending code 16#00 appears in the low byte, the execution of the instruction is as follows.



The operand **S**:

Device	High byte	Low byte	Note
D100	16#31	16#30	‘1’=16#31; ‘0’=16#30
D101	16#33	16#32	‘3’=16#33; ‘2’=16#32

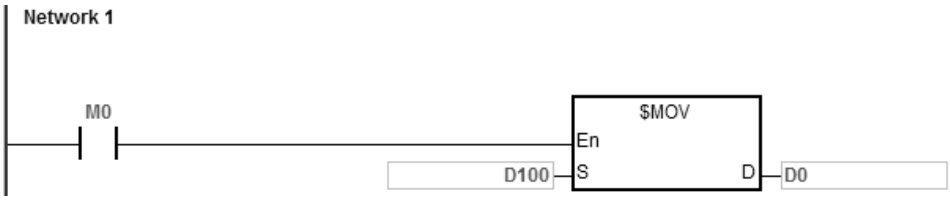
Device	High byte	Low byte	Note
D102	16#35	16#34	'5'=16#35; '4'=16#34
D103	16#30	16#00	'0'=16#30; 16#00 is the ending code.

After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D0	16#31	16#30	'1'=16#31; '0'=16#30
D1	16#33	16#32	'3'=16#33; '2'=16#32
D2	16#35	16#34	'5'=16#35; '4'=16#34
D3	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D4	Unchanged	Unchanged	

Example 4

When the data in **S** is not a string and the ending code 16#00 appears in the high byte, the execution of the instruction is as follows.



The operand **S**:

Device	High byte	Low byte	Note
D100	16#31	16#30	'1'=16#31; '0'=16#30
D101	16#33	16#32	'3'=16#33; '2'=16#32
D102	16#00	16#34	16#00 is the ending code. '4'=16#34
D103	16#37	16#36	'7'=16#37; '6'=16#36

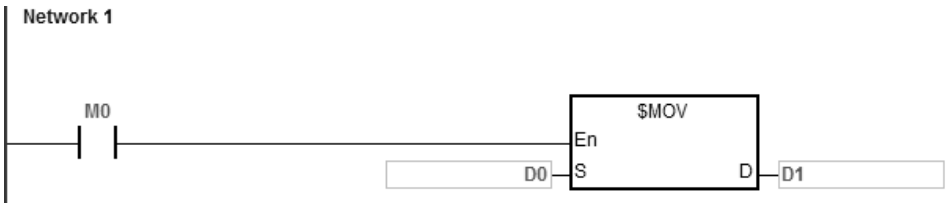
After the instruction is executed, the data in the operand **D** is as follows.

Device	High byte	Low byte	Note
D0	16#31	16#30	'1'=16#31; '0'=16#30
D1	16#33	16#32	'3'=16#33; '2'=16#32

D2	16#00	16#34	16#00 is the ending code. ‘4’=16#34
D3	Unchanged	Unchanged	

Example 5

When **S** overlaps **D**, and the device number of **S** is less than the device number of **D**, the transfer of the data to **D** starts from the ending code 16#00.



The operand **S**:

Device	High byte	Low byte	Note
D0	16#31	16#30	‘1’=16#31; ‘0’=16#30
D1	16#33	16#32	‘3’=16#33; ‘2’=16#32
D2	16#35	16#34	‘5’=16#35; ‘4’=16#34
D3	16#30	16#00	‘0’=16#30; 16#00 is the ending code.
D4	16#38	16#37	‘8’=16#38; ‘7’=16#37

After the instruction is executed, the data in **D** is as follows.

Device	High byte	Low byte	Note
D1	16#31	16#30	‘1’=16#31; ‘0’=16#30
D2	16#33	16#32	‘3’=16#33; ‘2’=16#32
D3	16#35	16#34	‘5’=16#35; ‘4’=16#34
D4	16#00	16#00	The ending code 16#00 is in the low byte. 16#00 is automatically added in the high byte.
D5	Unchanged	Unchanged	

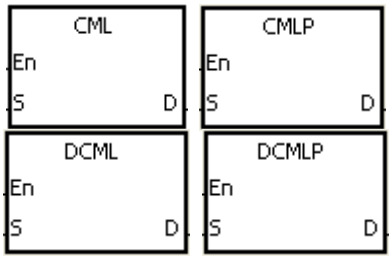
API	Instruction code			Operand	Function
0303	D	CML	P	**S, D**	Inverting data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Data source

D : Data destination

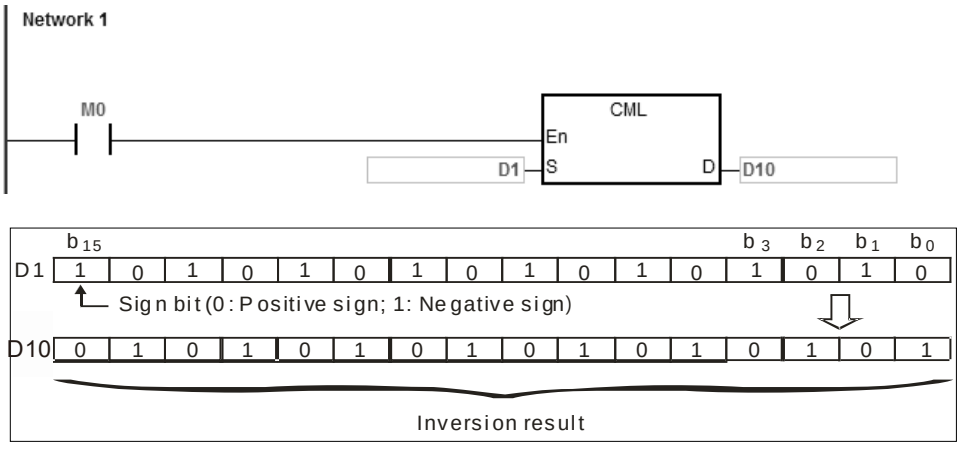
Explanation

1. This instruction inverts all bits in **S**; that is, 0 becomes 1, and 1 becomes 0, and stores the inversion result in **D**. If the data in **S** is a constant, the instruction converts it into a binary value.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1

When M0 is ON, the instruction inverts b0~b15 in D1, and stores the conversion result in b0~b15 in D10.



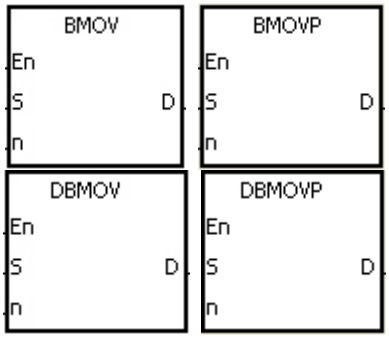
API	Instruction code			Operand	Function
0304	**D**	BMOV	P	**S, D, n**	Transferring data in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●							
D					●	●	●	●								
n					●	●	●	●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Data source

D : Data destination

n : Data length

Explanation

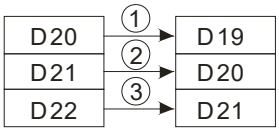
1. This instruction transfers **n** pieces of data in a block starting from the device specified by **S** to the devices starting from the device specified by **D**.

2. The value in **n** must be between 1–256.

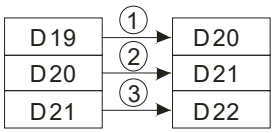
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

4. To prevent overlapping the source and the destination, the instruction transfers the data in the following way (using the 16-bit instruction as an example).

   When the device number of **S** is larger than the device number of **D**, the data is transferred in the order from ① to ③.

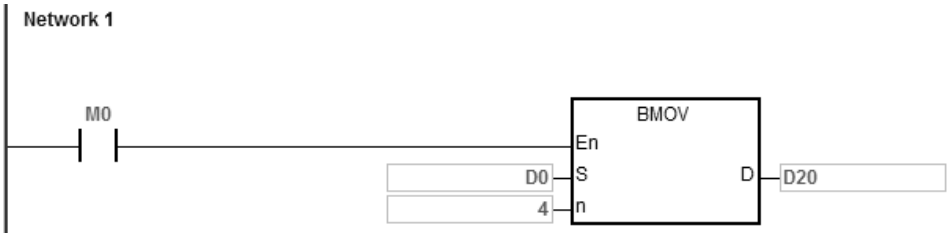


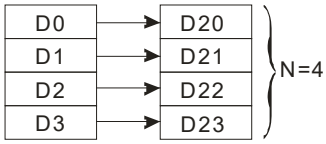
When the device number of **S** is less than the device number of **D**, the data is transferred in the order from ③ to ①.



Example 1

When M0 is ON, the instruction transfers the data in D0–D3 to D20–D23.

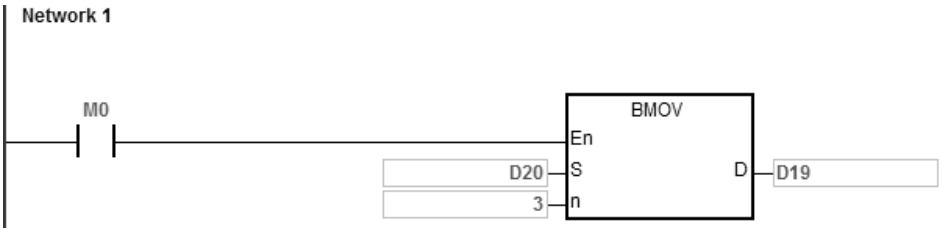


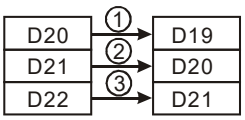


Example 2

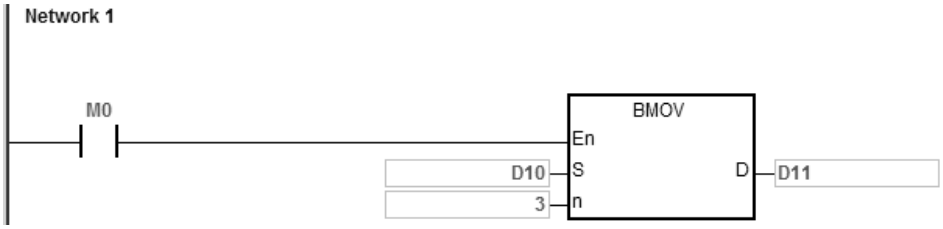
To prevent overlapping the source and the destination, the data is transferred in the following way.

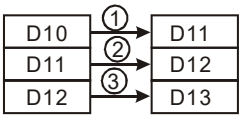
1. When the device number of **S** is larger than the device number of **D**, the data is transferred in the order from ① to ③.





2. When the device number of **S** is less than the device number of **D**, the data is transferred in the order from ③ to ①.





Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

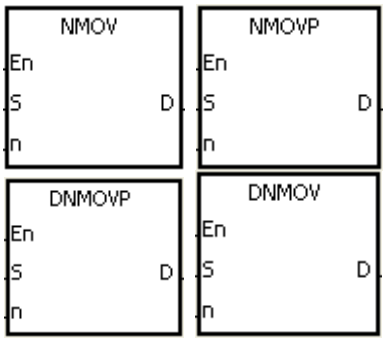
API	Instruction code			Operand	Function
0305	D	NMOV	P	**S, D, n**	Transferring data to multiple devices

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●				○	○		
D					●	●	●	●								
n					●	●	●	●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Data source

D : Data destination

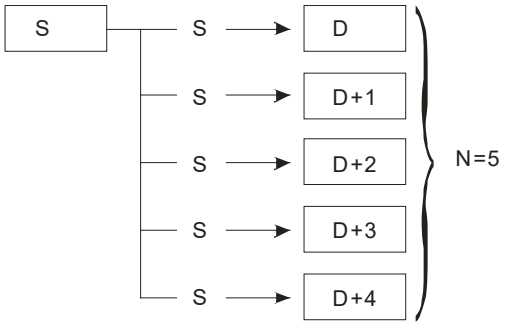
n : Data length

Explanation

1. This instruction transfers the data in **S** to the **n** devices starting from the device specified by **D**. When the instruction is not executed, the data in **D** is unchanged.

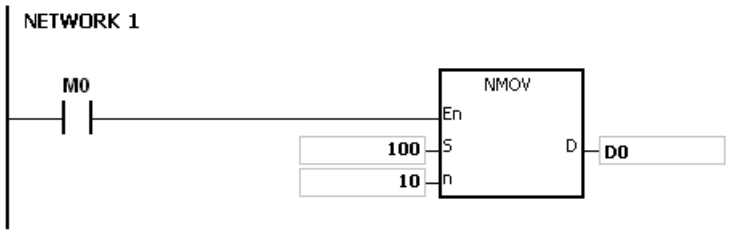
2. Only the 32-bit instructions can use the 32-bit counter.

3. The value in **n** in the NMOV instruction must be between 1–256.



Example

When M0 is ON, 100 is transferred to D0-D9.



Additional remarks

1. If **D**-**D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **n** in the 16-bit instruction is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

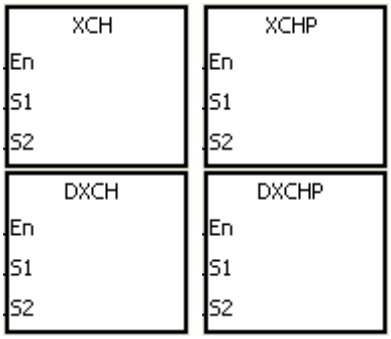
API	Instruction code			Operand	Function
0306	D	XCH	P	S_1, S_2	Exchanging data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●				○				
S_2					●	●	●	●				○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data to exchange

S_2 : Data to exchange

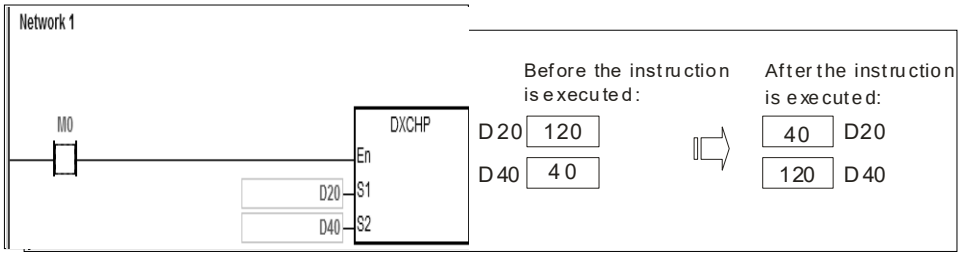
6

Explanation

1. This instruction exchanges the data in the device specified by S_1 with the data in the device specified by S_2.
2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

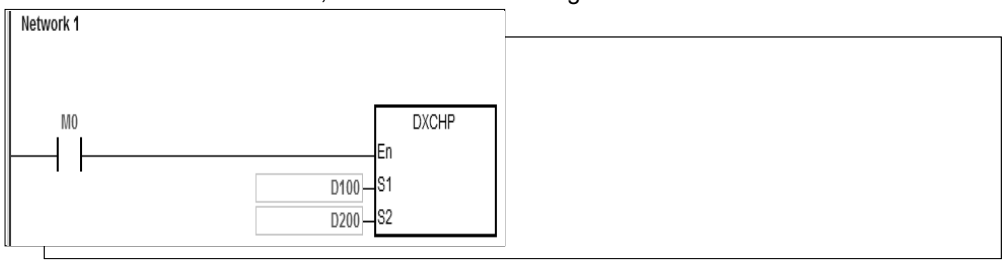
Example 1

When M0 is switched from OFF to ON, the instruction exchanges the data in D20 with the data in D40.



Example 2

When M0 switches from OFF to ON, the instruction exchanges the data in D100 with the data in D200.



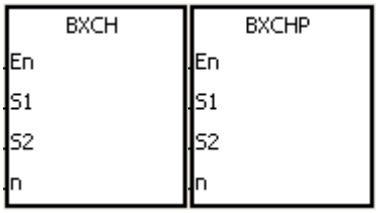
API	Instruction code			Operand	Function
0307		BXCH	P	**S_1 · S_2 · n**	Exchanging data in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●								
S_2					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Data to exchange

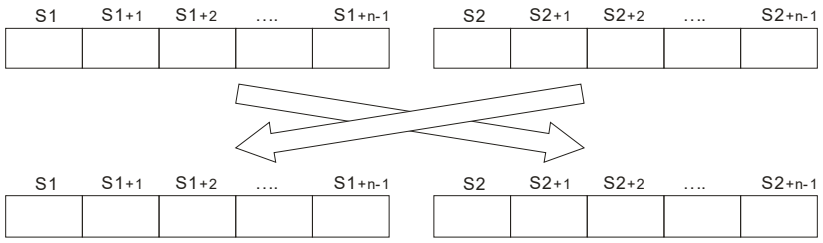
S_2 : Data to exchange

n : Data length

Explanation

1. This instruction exchanges the data in **S_1**–**S_1**+**n**-1 with the data in **S_2**–**S_2**+**n**-1.

2. The value in **n** must be between 1–256.



Example

When M0 is ON, the instruction exchanges the data in D10–D14 with the data in D100–D104.

D10	D11	D12	D13	D14
1	2	3	4	5

D100	D101	D102	D103	D104
16	17	18	19	20

After the instruction is executed

D10	D11	D12	D13	D14
16	17	18	19	20

D100	D101	D102	D103	D104
1	2	3	4	5

Additional remarks

1. If **S_1+n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S_2+n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

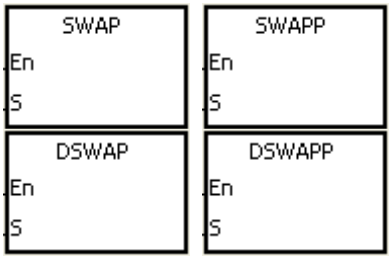
API	Instruction code			Operand	Function
0308	D	SWAP	P	**S**	Exchanging the high byte with the low byte

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



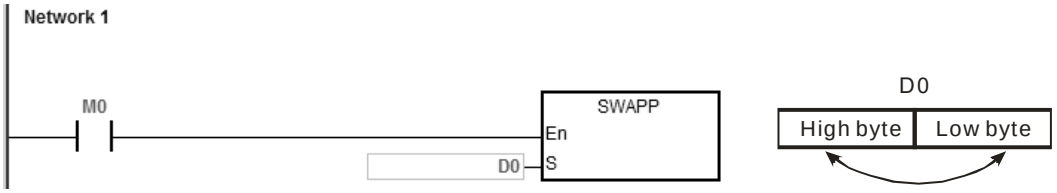
S : Data source

Explanation

1. The 16-bit instruction exchanges the data in the low byte in **S** with the data in the high byte in **S**.
2. The 32-bit instruction exchanges the data in the low byte of the high word in **S** with the data in the high byte of the high word in **S**, and exchanges the data in the low byte of the low word in **S** with the data in the high byte of the low word in **S**.
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

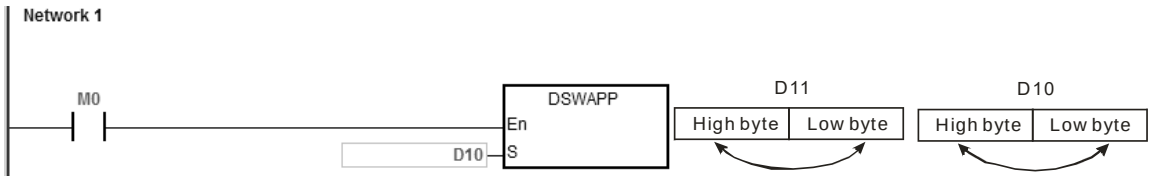
Example 1

When M0 is ON, the instruction exchanges the data in the low byte in D0 with the data in the high byte in D0.



Example 2

When M0 is ON, the instruction exchanges the data in the low byte in D11 with the data in the high byte in D11, and exchanges the data in the low byte in D10 with the data in the high byte in D10.



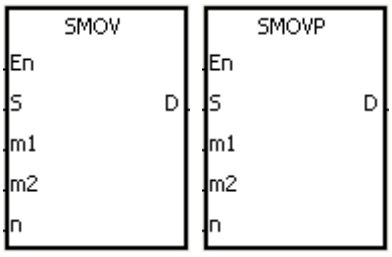
API	Instruction code			Operand	Function
0309		SMOV	P	**S, m_1, m_2, D, n**	Transferring digits in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●				○	○		
m_1					●	●		●	●				○	○		
m_2					●	●		●	●				○	○		
D					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
m_1		●			●	●							
m_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Data source

m_1 : Start digit to transfer from the source device

m_2 : Number of digits to transfer

D : Data destination

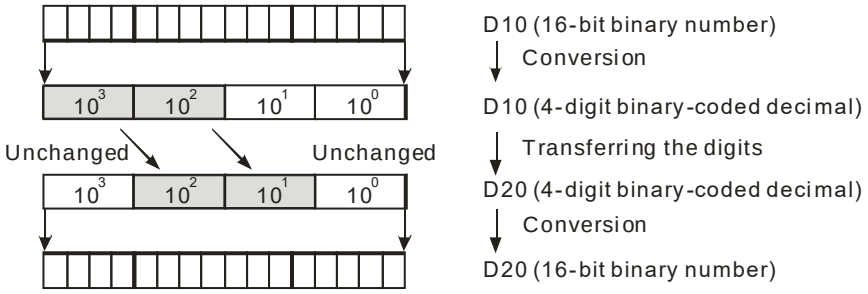
n : Start digit where the source data is stored in the destination device

Explanation

1. This instruction allocates and combines data. The instruction transfers **m_2** digits of the number starting from the **m_1^{th}** digit of the number in **S** to the **m_2** digits of the number starting from the **n^{th}** digit of the number in **D**.

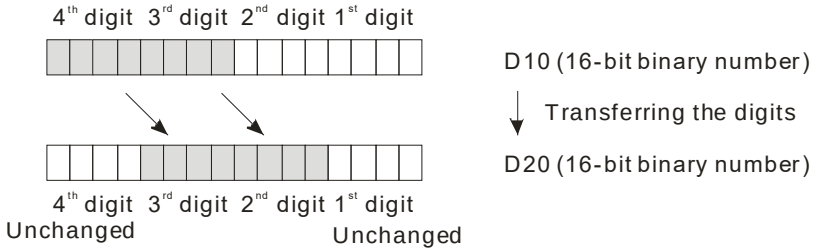
2. The value in **m_1** must be between 1–4. The value in **m_2** must be between 1–**m_1**. The value in **n** must be between **m_2**–4 (the instruction treats four bits as a unit.)

3. When SM605 is OFF, the data in **S** are binary-coded decimal numbers.



Suppose the number in **S** is K1234, and the number in **D** is K5678. After the instruction is executed, the number in **S** is 1234, and the number in **D** is 5128.

4. When SM605 is ON, the data involved in the instruction is binary numbers.

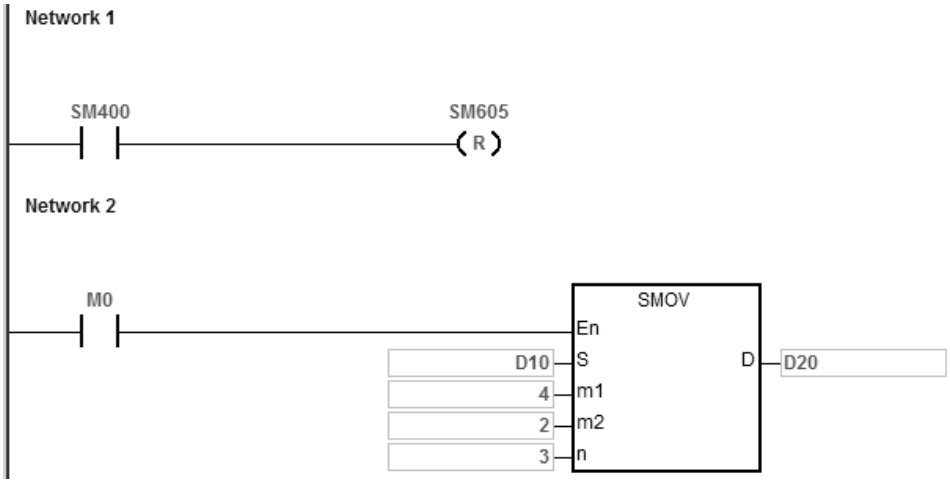


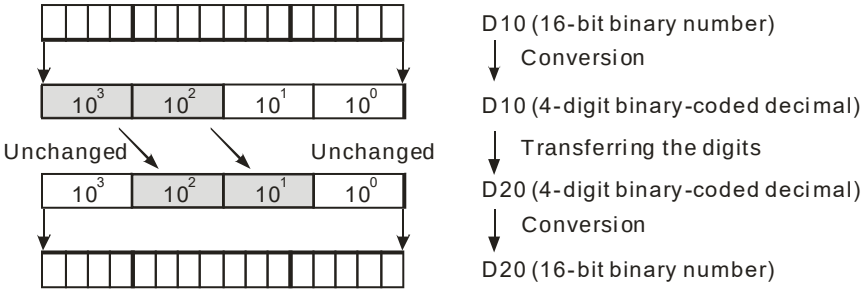
Suppose the number in **S** is 16#1234, and the number in **D** is 16#5678. After the instruction is executed, the number in **S** is 16#1234, and the number in **D** is 16#5128.

Example 1

1. When SM605 is OFF, the data in **S** are binary-coded decimal numbers. When M0 is ON, the instruction transfers two digits of the decimal number starting from the fourth digit of the decimal number (the digit in the thousands place of the decimal number) in D10 to the two digits of the decimal number starting from the third digit of the decimal number (the digit in the hundreds place of the decimal number) in D20. After the instruction is executed, the digits in the thousands place of the decimal number (10^3) and the ones place of the decimal number (10^0) in D20 are unchanged.

2. When the binary-code decimal number is not between 0–9,999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.



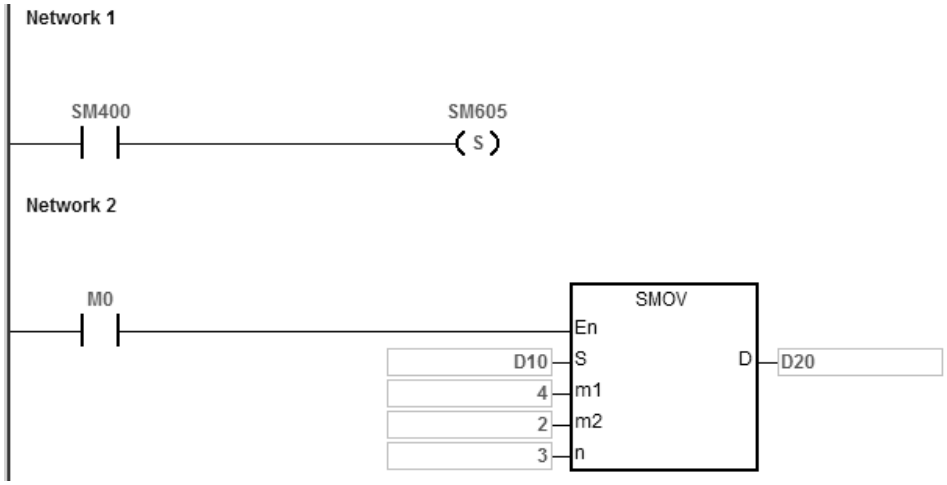


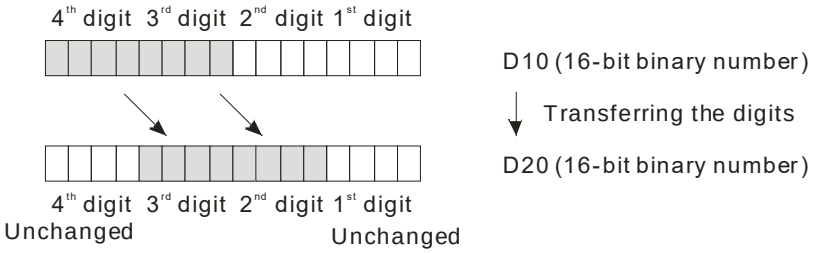
Suppose the number in D10 is 1234, and the number in D20 is 5678. After the instruction is executed, the number in D10 is unchanged, and the number in D20 is 5128.

6

Example 2

When SM605 is ON, the data are binary numbers. The SMOV instruction transfers the digit composed of four bits. The instruction does not transform the data into binary-coded decimal numbers.





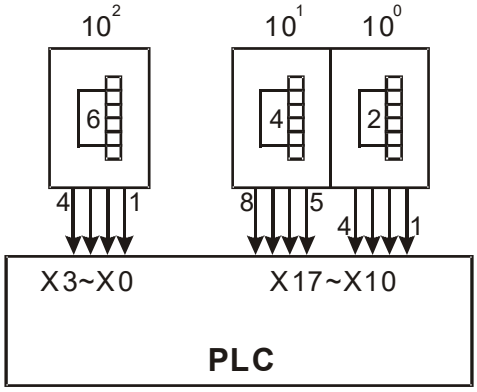
Suppose the number in D10 is 16#1234, and the number in D20 is 16#5678. After the instruction is executed, the number in D10 is unchanged, and the number in D20 is 16#5128.

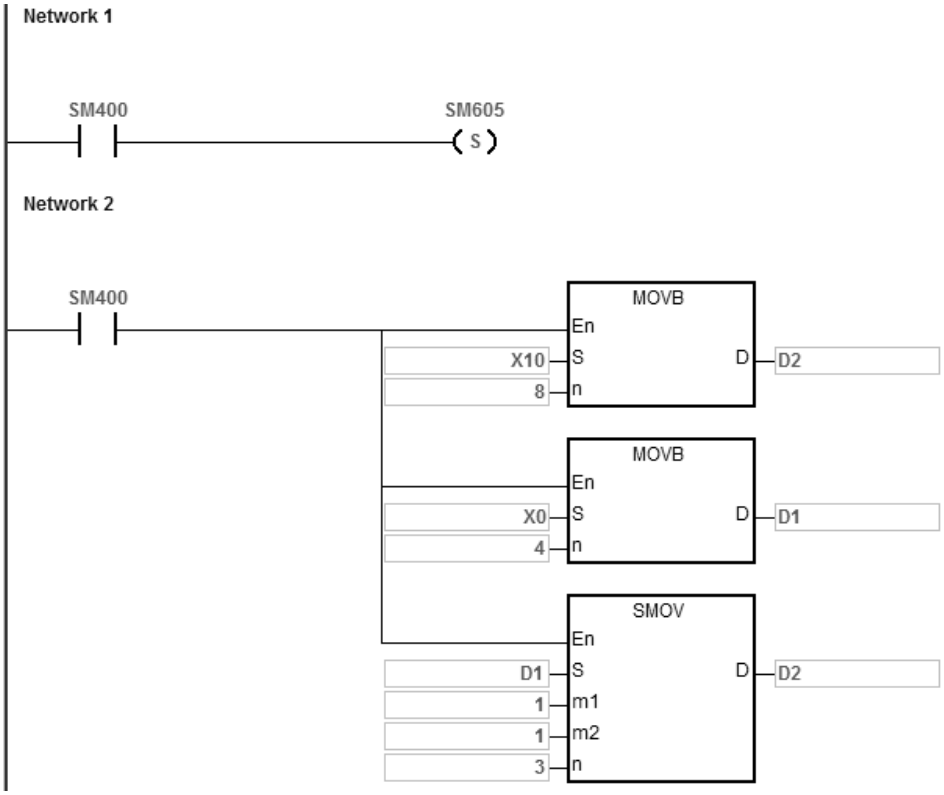
Example 3

1. You can use the instruction to combine the values of the DIP switches that are connected to the input terminals whose numbers are not consecutive.

2. The two digits of the value of the DIP switch at the right are transferred to the two digits of the number which start from the second digit of the number in D2, and the one digit of the value of the DIP switch at the left is transferred to the first digit of the number in D1.

3. You can use the SMOV instruction to transfer the first digit of the number in D1 to the third digit of the number in D2. In other words, the two DIP switches can be combined into one DIP switch by means of the SMOV instruction.





Additional remarks

1. Suppose the data are binary-coded decimal numbers. If the number in **S** is not between 0–9999, or if the number in **D** is not between 0–9999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.

2. If $\mathbf{m_1}$ is less than 1, or if $\mathbf{m_1}$ is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If $\mathbf{m_2}$ is less than 1, or if $\mathbf{m_2}$ is larger than $\mathbf{m_1}$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If **n** is less than $\mathbf{m_2}$, or if **n** is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

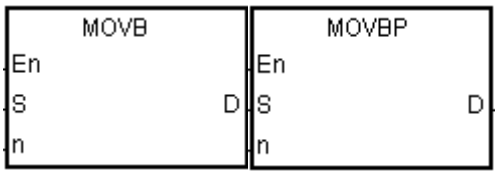
API	Instruction code			Operand	Function
0310		MOVB	P	**S, n, D**	Transferring bits in blocks

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●	●	●		●						
n					●	●		●	●		○	○	○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
n		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Data source

n : Data length

D : Data destination

Explanation

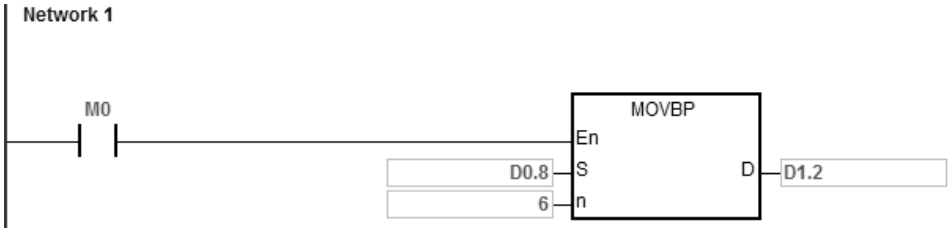
1. This instruction transfers **n** pieces of data in devices starting from the device specified by **S** to the devices starting from the device specified by **D**.

2. When **S** is T, C or HC, the instruction transfers only the state of the device, but does not transfer the current value of the device.

3. The value in **n** must be between 1–256. When **n** not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

Example

When M0 is ON, the instruction transfers the data in D0.8–D0.13 to D1.2–D1.7.



Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.5 Jump Instructions

6.5.1 List of Jump Instructions

The following table lists the Jump instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0400	CJ	–	✓	Conditional jump
0401	JMP	–	–	Unconditional jump
0402	GOEND	–	–	Jumping to the end of the program

6.5.2 Explanation of Jump Instructions

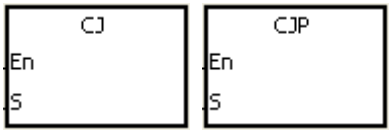
API	Instruction code			Operand	Function
0400		CJ	P	**S**	Conditional jump

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Jump destination

Explanation

1. This instruction jumps from the current program execution to a label (destination) in a different part of the program in the PLC. You specify the label (pointer) in **S**. You can use the CJ or CJP instruction to shorten the scan time. You can also use the CJ or CJP instruction when using a dual output.

2. If the program specified by the jump destination (label) is prior to the CJ instruction, the watchdog timer error occurs, and the PLC stops running the program. Use this instruction carefully.

3. You can specify the same label repeatedly with the multiple different CJ instructions.

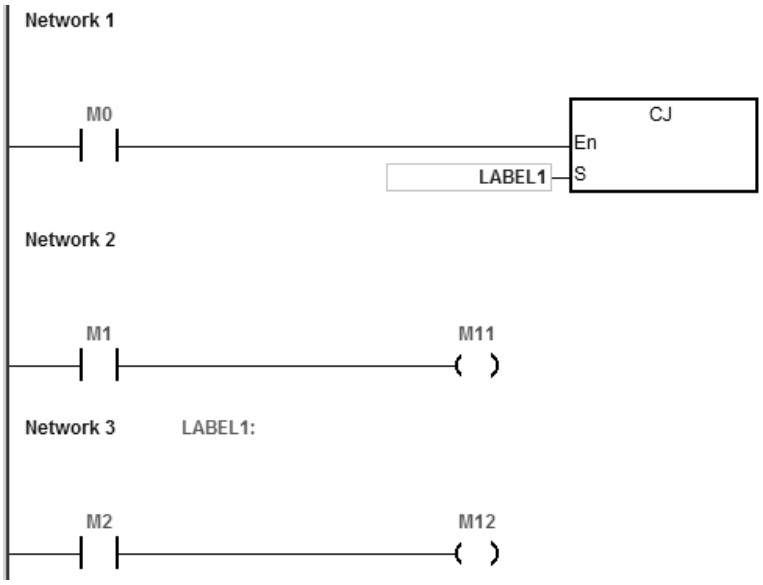
4. When the instruction is executed, the actions of the devices are as described below.

   - The state of Y, the state of M, and the state of **S** remain the same as before the execution of the jump.
   - The timer keeps counting and when it reaches the time setting value, the program drives the output T-coil.
   - For more information on the MC and MCR instructions, refer to Example 2 below.
   - The general applied instructions are not executed.

Example 1

1. When M0 is ON, the program execution jumps from NETWORK 1 to LABEL1 (NETWORK 3) and skips NETWORK 2.

2. When M0 is OFF, the execution of the program goes from NETWORK 1 to NETWORK 2 to NETWORK 3 in sequence, and the CJ instruction is not executed.



Example 2

1. You can use the CJ instruction between the MC and the MCR instructions in the five conditions below.

   (a) The execution of the program jumps from the part of the program outside one MC/MCR loop to the part of the program outside another MC/MCR loop.

   (b) The execution of the program jumps from the part of the program outside the MC/MCR loop to the part of the program inside the MC/MCR loop.

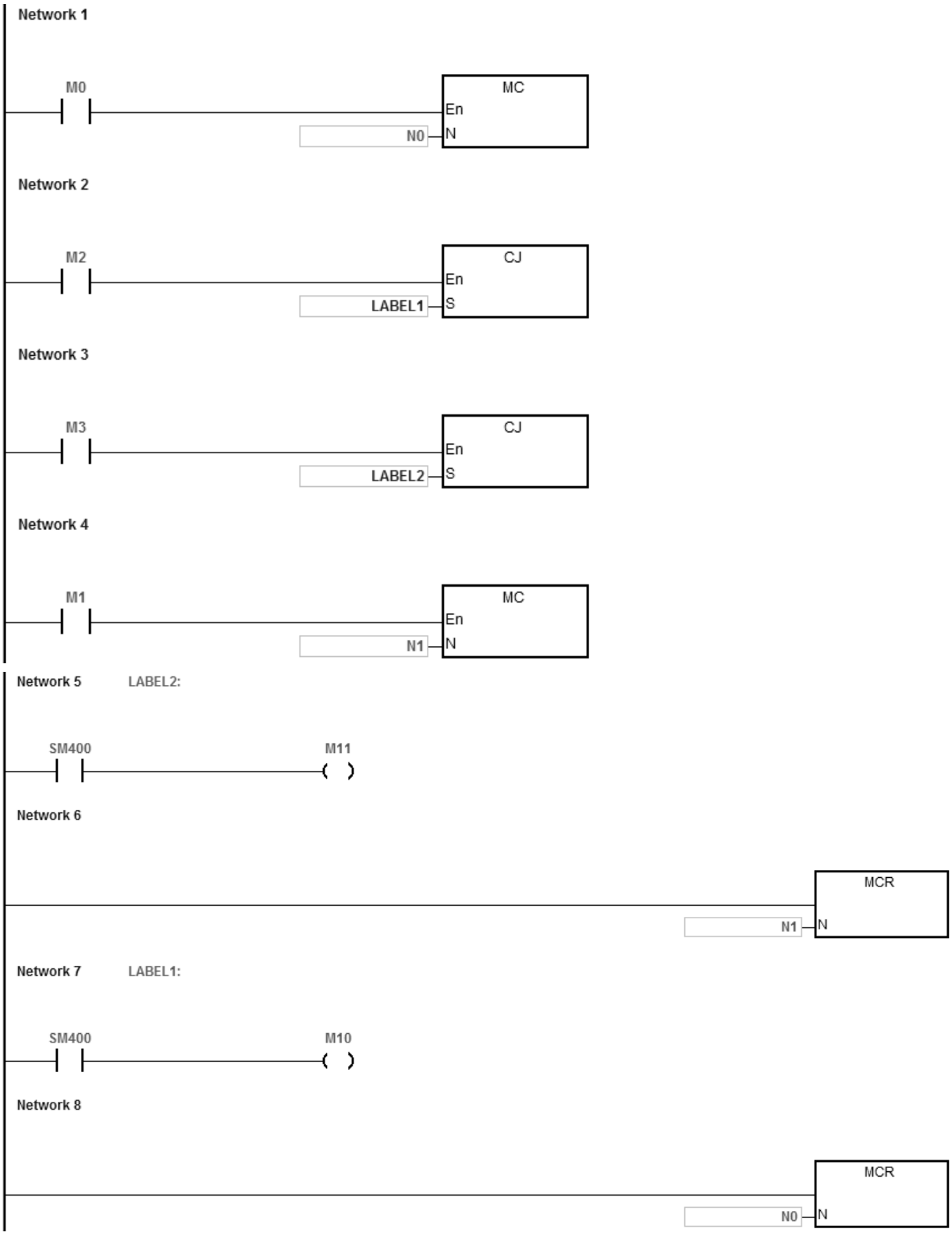
   (c) The execution of the program jumps from the part of the program inside the MC/MCR loop to the part of the program inside the MC/MCR loop.

   (d) The execution of the program jumps from the part of the program inside the MC/MCR loop to the part of the program outside the MC/MCR loop.

   (e) The execution of the program jumps from the part of the program inside one the MC/MCR loop to the part of the program inside another the MC/MCR loop.

2. When the PLC executes an MC instruction, it puts the previous state of the switch contact onto the top of the stack inside the PLC. The stack is controlled by the PLC, and cannot be changed. When the PLC executes the MCR instruction, the PLC pops the previous state of the switch contact from the top of the stack. Under the conditions listed in (b), (d), and (e) above, the number of times the items are pushed onto the stack may be different from the number of times the items are popped from the stack. When this situation occurs, at most 32 items can be pushed

onto the stack; items can be popped from the stack until the stack is empty. Therefore, when you use CJ or CJP with MC and MCR, be careful of how the program pushes items onto the stack and pops items from the stack.

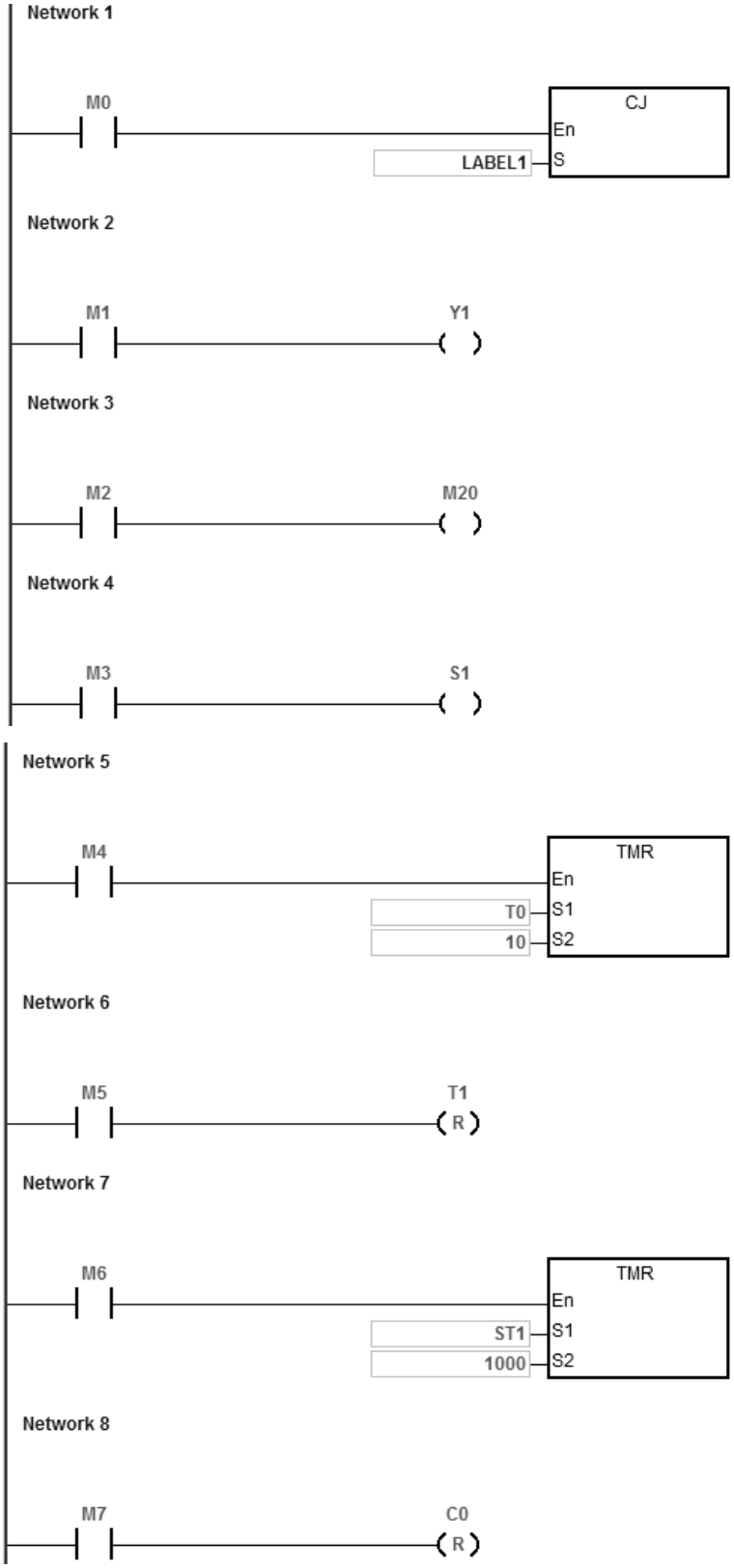


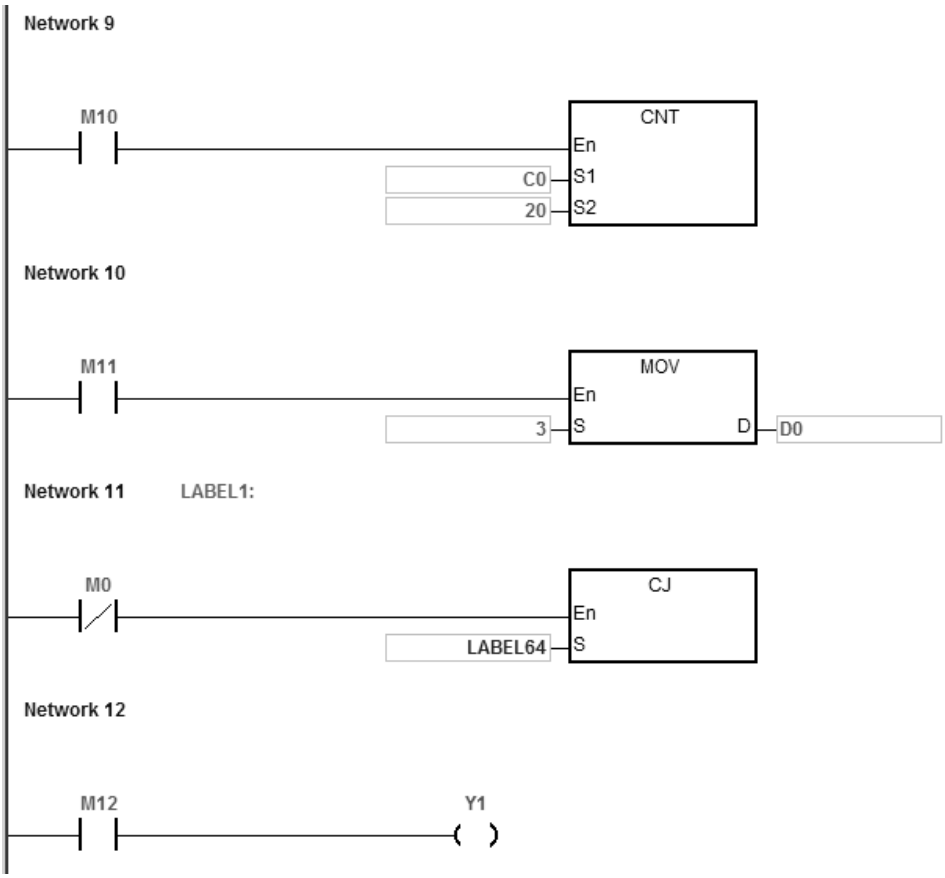
Example 3

The states of the devices are listed below.

Device	State of the contact before the execution of CJ M0=OFF	State of the contact during the execution of CJ M0=ON	State of the output coil during the execution of CJ M0=ON
Y, M, and S	M1, M2, and M3 are OFF.	M1, M2, and M3 switch from OFF to ON.	Y1[*1], M20, and S1 are OFF.
	M1, M2, and M3 are ON.	M1, M2, and M3 switch from ON to OFF.	Y1[*1], M20, and S1 are ON.
Timer	M4 is OFF.	M4 switches from OFF to ON.	The timer is not enabled.
	M4 is ON.	M4 switches from ON to OFF	The timer keeps counting and when the timer setting value is reached, it drives the output T-coil.
Accumulative timer	M6 is OFF.	M6 switches from OFF to ON.	ST1 is not enabled.
	M6 is ON.	M6 switches from ON to OFF.	The accumulative timer keeps counting and when the timer setting value is reached, it drives the output T-coil.
Counter	M7 and M10 are OFF.	M10 is ON/OFF.	The counter is not enabled.
	M7 is OFF. M10 is ON/OFF.	M10 is ON/OFF.	C0 stops counting. When M0 switches OFF, C0 keeps counting.
Applied instruction	M11 is OFF.	M11 switches from OFF to ON	The applied instruction is not executed.
	M11 is ON.	M11 switches from ON to OFF	The applied instruction is skipped (not executed).

*1:Y1 is a dual output. When M0 is OFF, Y1 is controlled by M1. When M0 is ON, Y1 is controlled by M12.


Network 1
M0
CJ
En
LABEL1
S
Network 2
M1
Y1
Network 3
M2
M20
Network 4
M3
S1
Network 5
M4
TMR
En
T0
S1
10
S2
Network 6
M5
T1
R
Network 7
M6
TMR
En
ST1
S1
1000
S2
Network 8
M7
C0
R



Additional remarks

Refer to the ISPSoft User Manual for more information on the use of labels (pointers) with Jump instructions.

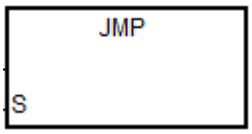
API	Instruction code	Operand	Function
0401	JMP	**S**	Unconditional jump

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : Jump destination

Explanation

1. This instruction causes the execution of the program to jump to the part of the program specified by the label in **S** (pointer) without any condition.
2. If the program specified by the label is prior to the instruction JMP, the watchdog timer error occurs, and the PLC stops running the program. Use this instruction carefully.
3. Refer to the CJ instruction (API 400) for more information on the states of devices while executing this instruction.
4. Refer to the ISPSoft User Manual for more information on the use of labels (pointers) with Jump instructions.

API	Instruction code			Operand	Function
0402		GOEND		—	Jumping to END

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

Explanation

1. This instruction causes program execution to jump to END in the program.

2. Function blocks and interrupt tasks do not support the GOEND instruction. You cannot use the instruction between the FOR instruction and the NEXT instruction.

3. When the PLC executes the GOEND instruction, the instructions skipped are not executed, the data in all devices is unchanged, and the states of all devices are also unchanged.

6.6 Program Execution Instructions

6.6.1 List of Program Execution Instructions

The following table lists the Program Execution instructions covered in this section.

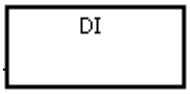
API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0500	DI	–	–	Disabling the interrupt function
0501	EI	–	–	Enabling the interrupt function
0503	EIX	–	–	Disabling a specific interrupt
0504	DIX	–	–	Enabling a specific interrupt

6.6.2 Explanation of Program Execution Instructions

API	Instruction code	Operand	Function
0500	DI	-	Disabling the interrupt function

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



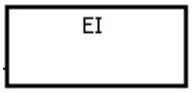
Explanation

Refer to the EI instruction (API 0501) for more information.

API	Instruction code			Operand	Function
0501		EI		-	Enabling the interrupt function

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Explanation

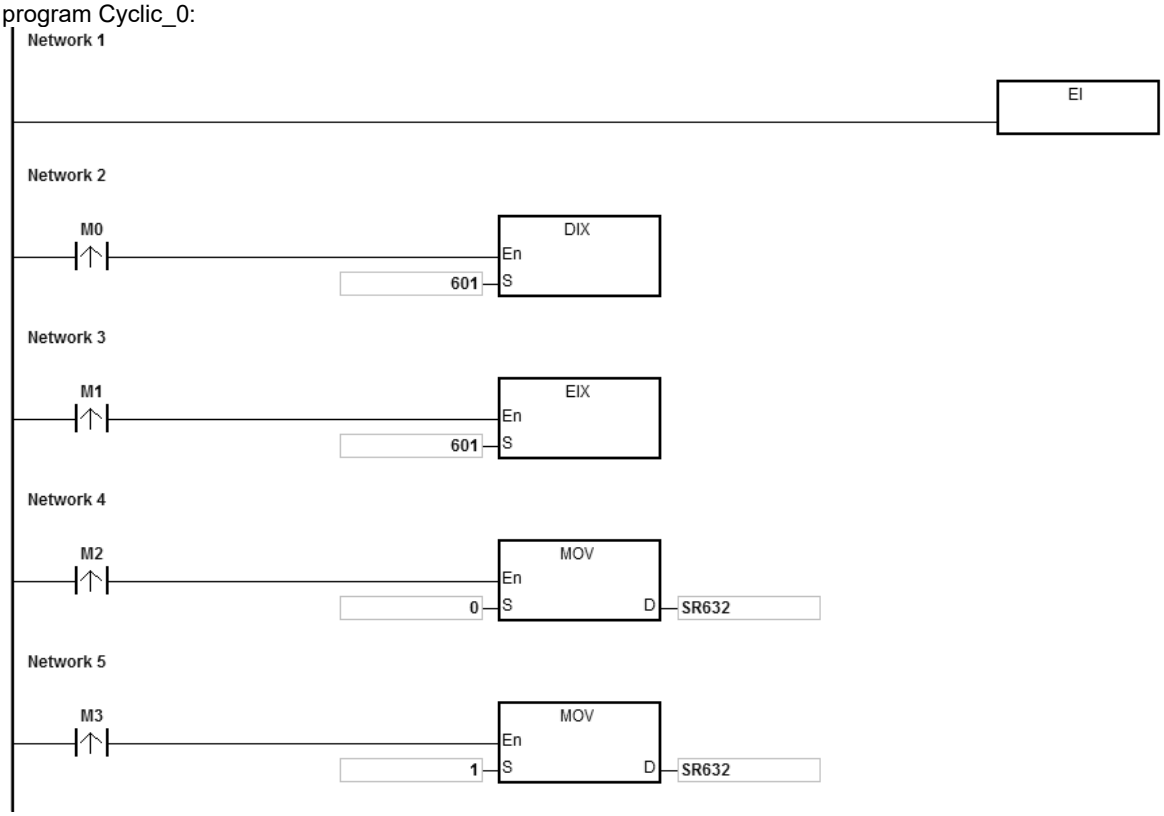
1. Use the EI instruction to enable interrupt tasks in a program (refer to next page for more information on tasks).
2. You can use the interrupt task between the EI instruction and the DI instruction in a program. You can choose not to use the DI instruction when there is no part of the program in which the interrupt is disabled.
3. During the execution of one interrupt task, a new interrupt generated is not executed, but is stored. After the execution of the present interrupt task is complete, the next interrupt task is executed. For example, during the execution of I0 (by trigged order #1), 2 new I0s (by trigged order #2, by trigged order #3) are generated, only by trigged order #2 I0 will be stored for later execution. by trigged order #3 IO is not stored for execution.
4. When several interrupts occur, the interrupt task with the highest priority is executed first. When several interrupts occur simultaneously, the interrupt task with the smallest pointer number is executed first.
5. When the interrupt task occurs between DI and EI, it cannot be executed, and the interrupt request is ignored. It is suggested that you not use the instruction DI to disable interrupts while PLC is running.
6. When the immediate I/O signal is required in the execution of the interrupt task, you can use the REF instruction or the device DX/DY in the program to refresh the state of the I/O.
7. Every interrupt number has a temporary function that can be masked. See below for the list of interrupt numbers.

Example:

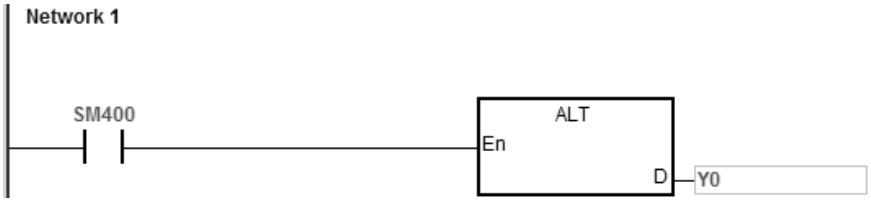
- Set up the timed I601 interrupt task to 500ms in HWCONFIG in ISPSoft.
- When the PLC runs the program Cyclic_0, it scans the EI instruction, enables interrupt tasks, and then executes the I601 interrupt task. When the interrupt task execution is complete, the main program is executed.
- When M0 is ON, the I601 timer interrupt task is disabled.
- When M1 is ON, the I601 timer interrupt task is enabled.

- When M2 is ON, the SR632 is 0 and the I601 timer interrupt task is disabled.
- When M3 is ON, the SR632 is 1 and the I601 timer interrupt task is enabled.

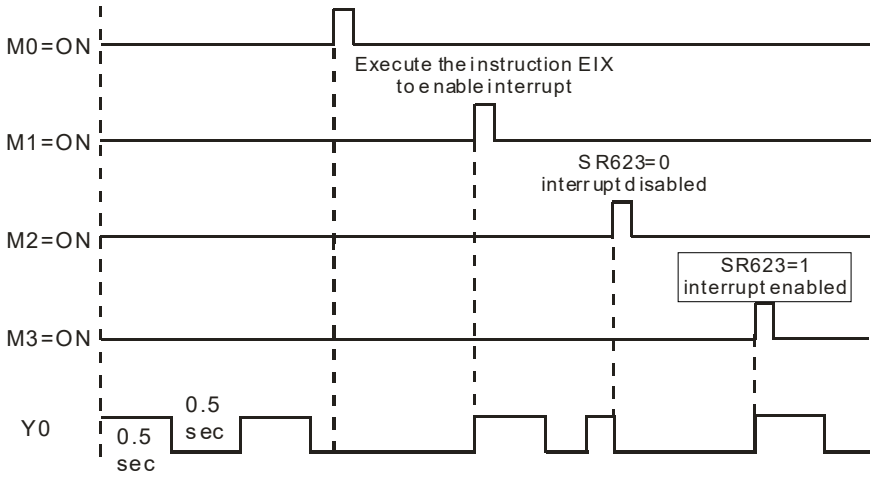
The program Cyclic_0:



The interrupt task:



Timer interrupts in a diagram:



Additional remarks

There are 7 types of interrupt tasks:

1. External interrupts (I000–I115)

   I000 specifies that the input X0 is falling edge triggered.

   I100 specifies that the input X1 is rising edge triggered.

   I101 specifies that the input X1 is falling edge triggered. The rest can be done in the same manner.

2. Hardware high-speed comparison interrupts (I200–I253)

   This type of interrupts can be further divided into 6 groups. Each group corresponds to a hardware high-speed counter (refer to the DCNT instruction API 1003 for more information). Each group has with 4 interrupt numbers (refer to the DHSCS instruction API 1005 for more information). For example, the interrupt numbers for the first group are I200–I203, and for the second group are I210–I213.

3. Software high-speed comparison interrupts (I260–I267)

   There are 8 interrupts for software high-speed comparisons and these 8 interrupts are shared with 8 high-speed counters.

4. Communication interrupts

   You can use the communication interrupt as the RS instruction; that is, receiving a specific character triggers the interrupt, or you can use it as a general interrupt. Refer to the COMRS instruction (API1812) for more information.

   COM1: I300

COM2: I302

5. High-speed output interrupts (I500–I519)

When the pulse output is complete, the interrupt request is sent. The interrupts (I500–I505) for the completing the execution of the positioning instruction work with special devices (SM) to activate the interrupt service. For example, when the DDRVI instruction completes the execution of the first axis, the interrupt request I500 is sent; you can set SM471 to ON to activate the interrupt service. The interrupts (I510–I519) for the completing the execution of the position planning table instruction work with the TPO instruction. When the pulse output is complete, the interrupt request is sent.

6. Timer interrupts (I601–I604)

Set the timer interrupts set in HWCONFIG.

For the timer interrupts I601–I603: The default value is 10 milliseconds (unit: 1ms) (1–2000 milliseconds).

For the timer interrupts I604: The default value is 1 milliseconds (unit: 0.1ms) (0.1–200 milliseconds).

The complete list of interrupt numbers, descriptions and the maskable interrupts (SR) are listed in the following table.

Interrupt number	Description	Maskable interrupts SR	Bit No.
I000	External interrupt: input X0 is falling edge triggered.	SR623	0
I001	External interrupt: input X1 is falling edge triggered.		1
I002	External interrupt: input X2 is falling edge triggered.		2
I003	External interrupt: input X3 is falling edge triggered.		3
I004	External interrupt: input X4 is falling edge triggered.		4
I005	External interrupt: input X5 is falling edge triggered.		5
I006	External interrupt: input X6 is falling edge triggered.		6
I007	External interrupt: input X7 is falling edge triggered.		7
I008	External interrupt: input X10 is falling edge triggered.		8
I009	External interrupt: input X11 is falling edge triggered.		9
I010	External interrupt: input X12 is falling edge triggered.		10
I011	External interrupt: input X13 is falling edge triggered.		11
I012	External interrupt: input X14 is falling edge triggered.		12
I013	External interrupt: input X15 is falling edge triggered.		13
I014	External interrupt: input X16 is falling edge triggered.		14
I015	External interrupt: input X17 is falling edge triggered.		15

Interrupt number	Description	Maskable interrupts SR	Bit No.
I100	External interrupt: input X0 is rising-edge triggered.	SR624	0
I101	External interrupt: input X1 is rising-edge triggered.		1
I102	External interrupt: input X2 is rising-edge triggered.		2
I103	External interrupt: input X3 is rising-edge triggered.		3
I104	External interrupt: input X4 is rising-edge triggered.		4
I105	External interrupt: input X5 is rising-edge triggered.		5
I106	External interrupt: input X6 is rising-edge triggered.		6
I107	External interrupt: input X7 is rising-edge triggered.		7
I108	External interrupt: input X10 is rising-edge triggered.		8
I109	External interrupt: input X11 is rising-edge triggered.		9
I110	External interrupt: input X12 is rising-edge triggered.		10
I111	External interrupt: input X13 is rising-edge triggered.		11
I112	External interrupt: input X14 is rising-edge triggered.		12
I113	External interrupt: input X15 is rising-edge triggered.		13
I114	External interrupt: input X16 is rising-edge triggered.		14
I115	External interrupt: input X17 is rising-edge triggered.		15
I200	High-speed comparison interrupt 1 for the hardware high-speed counter 1	SR625	0
I201	High-speed comparison interrupt 2 for the hardware high-speed counter 1		1
I202	High-speed comparison interrupt 3 for the hardware high-speed counter 1		2
I203	High-speed comparison interrupt 4 for the hardware high-speed counter 1		3
I210	High-speed comparison interrupt 1 for the hardware high-speed counter 2		4
I211	High-speed comparison interrupt 2 for the hardware high-speed counter 2		5
I212	High-speed comparison interrupt 3 for the hardware high-speed counter 2		6
I213	High-speed comparison interrupt 4 for the hardware high-speed counter 2		7

Interrupt number	Description	Maskable interrupts SR	Bit No.
I220	High-speed comparison interrupt 1 for the hardware high-speed counter 3		8
I221	High-speed comparison interrupt 2 for the hardware high-speed counter 3		9
I222	High-speed comparison interrupt 3 for the hardware high-speed counter 3		10
I223	High-speed comparison interrupt 4 for the hardware high-speed counter 3		11
I230	High-speed comparison interrupt 1 for the hardware high-speed counter 4		12
I231	High-speed comparison interrupt 2 for the hardware high-speed counter 4		13
I232	High-speed comparison interrupt 3 for the hardware high-speed counter 4		14
I233	High-speed comparison interrupt 4 for the hardware high-speed counter 4		15
I240	High-speed comparison interrupt 1 for the hardware high-speed counter 5	SR626	0
I241	High-speed comparison interrupt 2 for the hardware high-speed counter 5		1
I242	High-speed comparison interrupt 3 for the hardware high-speed counter 5		2
I243	High-speed comparison interrupt 4 for the hardware high-speed counter 5		3
I250	High-speed comparison interrupt 1 for the hardware high-speed counter 6		4
I251	High-speed comparison interrupt 2 for the hardware high-speed counter 6		5
I252	High-speed comparison interrupt 3 for the hardware high-speed counter 6		6
I253	High-speed comparison interrupt 4 for the hardware high-speed counter 6		7
I260	High-speed comparison interrupt 1 for the software	SR627	0

Interrupt number	Description	Maskable interrupts SR	Bit No.
	high-speed counter		
I261	High-speed comparison interrupt 2 for the software high-speed counter		1
I262	High-speed comparison interrupt 3 for the software high-speed counter		2
I263	High-speed comparison interrupt 4 for the software high-speed counter		3
I264	High-speed comparison interrupt 5 for the software high-speed counter		4
I265	High-speed comparison interrupt 6 for the software high-speed counter		5
I266	High-speed comparison interrupt 7 for the software high-speed counter		6
I267	High-speed comparison interrupt 8 for the software high-speed counter		7
I300	Receiving a specific word triggers communication interruption in COM1	SR628	0
I301	Reserved		1
I302	Receiving a specific word triggers communication interruption in COM1		2
I303	Reserved		3
I304	Reserved		4
I305	Reserved		5
I306	Reserved		6
I307	Reserved		7
I500	High-speed output interrupt: the 1st axis positioning instruction completes	SR629	0
I501	High-speed output interrupt: the 2nd axis positioning instruction completes		1
I502	High-speed output interrupt: the 3rd axis positioning instruction completes		2
I503	High-speed output interrupt: the 4th axis positioning		3

Interrupt number	Description	Maskable interrupts SR	Bit No.
	instruction completes		
I504	High-speed output interrupt: the 5th axis positioning instruction completes		4
I505	High-speed output interrupt: the 6th axis positioning instruction completes		5
I510	High-speed output interrupt 1: the position planning table instruction completes	SR630	0
I511	High-speed output interrupt 2: the position planning table instruction completes		1
I512	High-speed output interrupt 3: the position planning table instruction completes		2
I513	High-speed output interrupt 4: the position planning table instruction completes		3
I514	High-speed output interrupt 5: the position planning table instruction completes		4
I515	High-speed output interrupt 6: the position planning table instruction completes		5
I516	High-speed output interrupt 7: the position planning table instruction completes		6
I517	High-speed output interrupt 8: the position planning table instruction completes		7
I518	High-speed output interrupt 9: the position planning table instruction completes		8
I519	High-speed output interrupt 10: the position planning table instruction completes		9
I601	Timer interrupts 1 (unit 1ms)	SR632	0
I602	Timer interrupts 1 (unit 1ms)		1
I603	Timer interrupts 1 (unit 1ms)		2
I604	Timer interrupts 1 (unit 0.1ms)		3

Note: When several interrupts occur simultaneously, the interrupt task whose pointer number is smallest is executed first. The PLC completes the on-going interrupt, and then execute other interrupts according to their pointer numbers. For example, during the execution of I400 interrupt, if I500 and I300 occur simultaneously, the PLC executes the I300 interrupt (smaller pointer number) after executing the I400 interrupt.

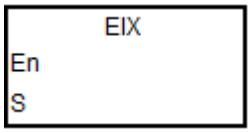
API	Instruction code			Operand	Function
0503		EIX		**S**	Enabling a specific interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S													○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : Data source

Explanation

1. The data source **S** can only contain a decimal number, and the number must be an interrupt number. If the number is not an interrupt number, the instruction is not executed and no warning is shown. For example, use EIX500 in **S** when you want to enable the I500 interrupt. Refer to the interrupt number list in the explanation of the EI instruction.
2. The default for interrupt tasks in the ES3 Series is enabled. If you use the DIX instruction to disable the interrupts, you must use the EIX instruction to enable the interrupts.
3. You can use this instruction to enable the interrupt tasks in SR623–SR634.
4. If this instruction is not executed, then the contents of SR623–SR634 determine whether an interrupt task is performed or not.
5. Refer to the examples for the EI instruction (API 0501) for more information.

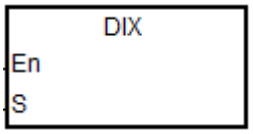
API	Instruction code	Operand	Function
0504	DIX	**S**	Disabling a specific interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S													○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S													

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : Data source

Explanation

1. The data source **S** can contain only decimal numbers, and the number must be an interrupt number. If the number in **S** is not an interrupt number, the instruction is not executed and no warning is shown. For example, use DIX500 when you want to disable the I500 interrupt. Refer to the interrupt number list in the explanation of the EI instruction.
2. The default for interrupt tasks in the ES3 Series is enabled. Use the DIX instruction to disable the interrupts.
3. You can use this instruction to disable the interrupt tasks in SR623–SR634.
4. If this instruction is not executed, then the contents of SR623–SR634 determine whether an interrupt task is performed or not.
5. Refer to the examples for the EI instruction (API 0501) for more information.

6.7 I/O Refreshing Instructions

6.7.1 I/O List of I/O Refreshing Instructions

The following table lists the I/O Refreshing instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0600	REF	–	✓	Refreshing the I/O
0601	–	DHSRF	✓	Immediate refresh of a high-speed comparative value
0602	REFF	–	✓	Refreshing the I/O filtering time

6.7.2 Explanation of I/O Refreshing Instructions

API	Instruction code			Operand	Function
0600		REF	P	**D, n**	Refreshing the I/O

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D	○	○														
n								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

D : I/O point state to refresh

n : Number of I/O points states to refresh

Explanation

1. The I/O states are normally not refreshed until the PLC executes the END instruction. When the PLC starts scanning the program, it reads and stores the states of the external inputs in memory. After executing the END instruction, the PLC sends the states of the outputs in the memory to the output terminals. Therefore, when you need the latest I/O data during the operation process, you can use this instruction, or use the device DX/DY to refresh the input/output.
   Note: the immediate I/O updates only apply for I/Os of the CPU.

2. The operand **n** must be multiples of eight, such as 8, 16, 24 and up to 256. If the value here is less than a multiple of eight, it will be seen as the next multiple of eight. For example, the value 20 will be seen as its next multiple of eight, 24.

3. The number of the high-speed output point is stored in D device. If **n** is 1, it indicates to refresh the high-speed output value of the corresponding SR immediately. If **n** is 0, it indicates to stop high-speed output and refresh the SR current value. For example, during the execution of this instruction, if **n** is 0 and the external interrupt input is received through X0, it indicates an external interrupt occurs in X0 and high-speed outputting through Y0 should be stopped immediately. The PLC sets the stop flag SM463 to ON and refresh the current corresponding output position in SR. Note: if the output completion auto-reset flag is set to ON, the PLC sets the output completion

auto-rest flag to OFF and refresh the current corresponding output position in SR. But the PLC does not set the stop flag SM463 to ON.

Value in **n**	D device	Action Descriptions
n = multiples of eight	Y0 or X0	Refresh I/O immediately See Example 1 and 2
n = 1	High-speed output point	Refresh new pulse positon See Example 3
n = 0	High-speed output point Without output completion auto-reset flag	Stop high-speed outputting, set the stop flag SM463 to ON and refresh the current corresponding output position. See Example 3
n = 0	High-speed output point With output completion auto-reset flag	Set the output completion auto-reset flag to OFF and refresh the current corresponding output position. See Example 3.
n = 1	Any input pint	Refresh data mapping area in DS301 TxPDO (Slave to Master) immediately
n = 1	Any output point	Refresh data mapping area in DS301 RxPDO (Master to Slave) immediately

Example 1

1. When M0 is ON, the PLC reads 16 states of the inputs X0–X7 and X10-17 immediately, and refreshes the input signals without any delay.

Example 2

When M0 is ON, the 8 output signals from Y0–Y7 are sent to the output terminals. The output signals are refreshed immediately without waiting for the END instruction to be executed.

Additional remarks

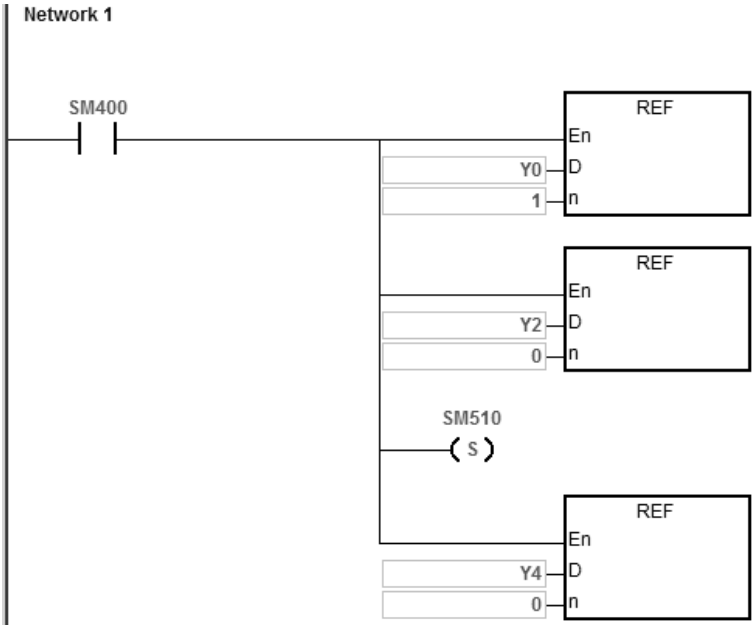
1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

Example 3

1. During the execution of this instruction, if the external interrupt input is received through X0

Value in **n**	D device	Action Descriptions
n = 1	High-speed output point Y0	Refresh new pulse positon Y0 immediately (SR460)
n = 0	High-speed output point Y2 Without output completion auto-reset flag	1. Stop high-speed Y2 outputting 2. Set the stop flag SM483 to ON 3. Refresh the current corresponding output position SR480
n = 0	High-speed output point Y4 With output completion auto-reset flag SM510	1. Set the output completion auto-reset flag SM510 to OFF 2. Refresh the current corresponding output position. SR500

X0 external interrupt program:



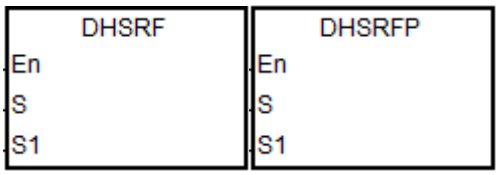
API	Instruction			Operand	Description
0601	D	HSRF	P	**S**，**S_1**	Immediate refresh of a high-speed comparative value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							○									
S_1								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S												●	
S_1			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	—	ES3

Symbol



S : High-speed counter to be refreshed

S_1 : Quantity of I/O to be updated

Explanation

1. The timing for the PLC to update the comparative value in its comparator is when the DHSCS or DHSCR instruction is scanned by a program successfully. However, the refresh may fail if the scan time is too long or the input signal comes too fast. In this event, users can use the instruction to assign the new comparative value to the hardware comparator in the PLC and achieve the real-time comparison.

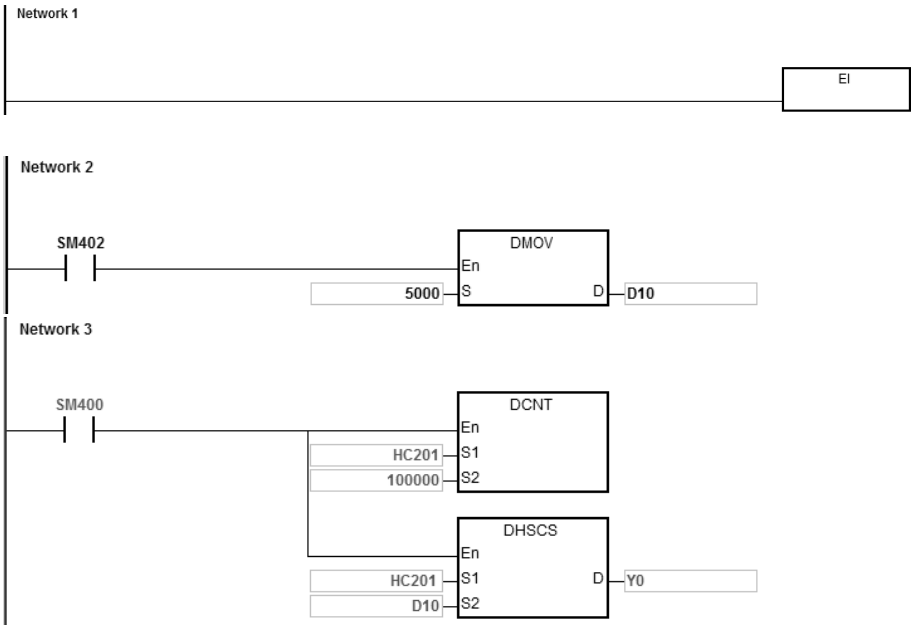
2. **S** is the No of the specified high-speed counter to be refreshed. **S_1** has the same component or variable name as that in the DHSCS or DHSCR instruction and the immediate value can not be set for **S_1**. If **S_1** does not have the same operand as that in the high-speed comparison instruction which has been enabled, the instruction execution will not take effect.

Example

1. As PLC runs, the comparative value in DHSCS instruction is 5000.

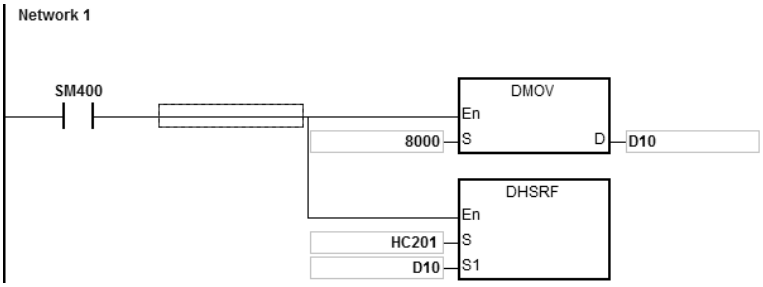
2. When the X0 external interrupt occurs, the comparative value in DHSCS is set to 8000 immediately.

   Main program:



   X0 external interrupt program:

   Assign the new comparative value to the same variable (E.g. D10 in the example) first and then execute DHSRF instruction for the update.



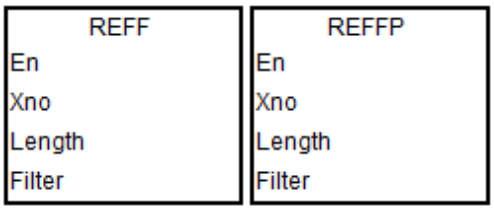
API	Instruction code			Operand	Function
0602		REFF	P	**Xno, Length, Filter**	Refreshing the I/O filtering time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Xno	○															
Length					●	●		●	●				○	○		
Filter					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Xno	●												
Length		●			●	●							
Filter		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



Xno : The starting device number of X to refresh

n : Number of I/O points states to refresh

Filter : Filtering time to refresh (unit: μs)

Explanation

1. This instruction works only with ISPSoft V3.06 or later versions.
2. **Xno** is the starting input number to refresh (X0~X7 and X10~17). **Length** is the length to refresh. When **Xno** is X3, the value of **Length** is 3, after executing the REFF instruction, the input filtering time of X3 ~ X5 are refreshed.
3. **Xno** and **Length** should be set in the input of PLC input system. If the setting exceeds the range, even if you have executed REFF instruction, the I/O will NOT be refreshed and error message will NOT be shown.
4. The filtering time unit is μs. This instruction is for refreshing the current filtering time; filtering time set in HWCONFIG will NOT be affected. But when the value of **Filter** in the instruction exceeds the setting range in HWCONFIG, PLC treats the value of **Filter** as the maximum or the minimum value set in HWCONFIG.
5. When power-on, the filtering time of the CPU is refreshed according to the filtering time set in HWCONFIG.
6. The REFF instruction refreshes the filtering time only at its first execution.

6.8 Miscellaneous Instructions

6.8.1 List of Miscellaneous Instructions

The following table lists the miscellaneous instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-Bit	32-Bit		
0700	ALT	–	✓	Alternating between ON and OFF
0701	TTMR	–	–	Teach mode timer
0702	STMR	–	–	Special timer
0703	RAMP	DRAMP	–	Cyclic ramp signal
0704	MTR	–	–	Matrix input
0705	ABSD	DABSD	–	Absolute drum sequencer
0706	INCD	–	–	Incremental drum sequencer
0708	–	DPIDE	–	PID algorithm
0709	XCMP	–	–	Setting up to compare the inputs of multiple work stations
0710	YOUT	–	–	Comparing the outputs of multiple work stations
0711	–	DSUNRS	✓	Sunrise and sunset times

6.8.2 Explanation of Miscellaneous Instructions

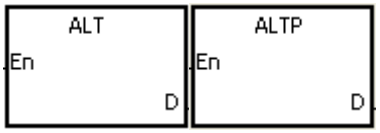
API	Instruction code			Operand	Function
0700		ALT	P	**D**	Alternating between ON and OFF

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D	●	●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
ES3	ES3	-

Symbol



D : Destination device

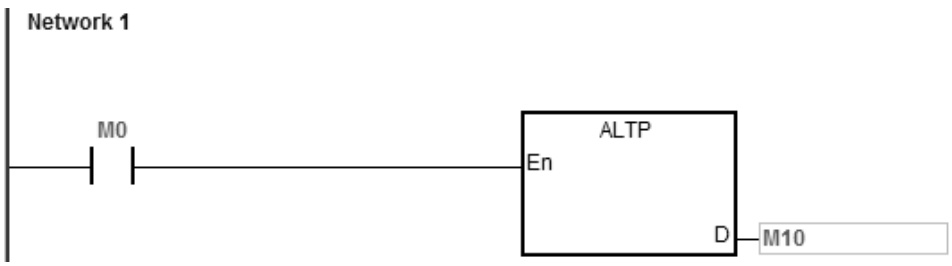
Explanation

1. This instruction alternates the state of the device specified by **D** between ON and OFF.

2. In general, use the ALTP pulse instruction.

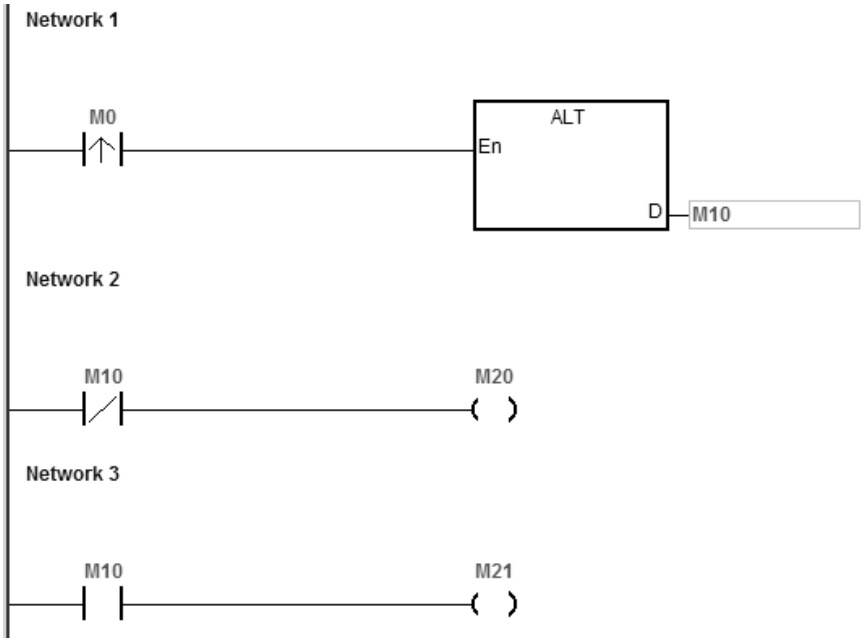
Example 1

When M0 switches from OFF to ON for the first time, M1 is ON. When M0 switches from OFF to ON for the second time, M1 is OFF.



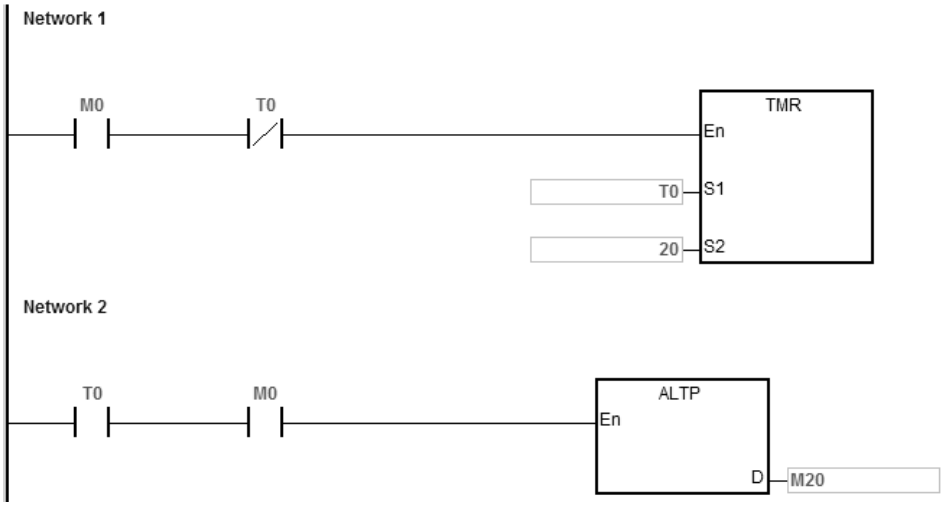
Example 2

In the beginning, M0 is OFF; therefore, M20 is ON, and M21 is OFF. When X0 switches from OFF to ON for the first time, M10 is ON; therefore, M20 is OFF, and M21 is ON. When X0 switches from OFF to ON for the second time, M10 is OFF; therefore, M20 is ON, and M21 is OFF.



Example 3

When M0 is ON, T0 generates a pulse every two seconds. The output M20 alternates between ON and OFF according to the pulses generated by T0.



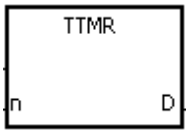
API	Instruction code	Operand	Function
0701	TTMR	**D, n**	Teach mode timer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D								●			○					
n								●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



D : Recorded time

n : Multiplier

Explanation

1. This instruction uses seconds as the unit of time. The time for which a button switch has been turned ON is multiplied by **n**, and the product is stored in **D**. **D**+1 is for system use only. When the instruction is executed, the value in **D**+1 cannot be altered. Otherwise, the time is counted incorrectly.

2. When the conditional contact is ON, **D** is reset to 0.

3. Setting the multiplier: when **n** is 0, **D** uses a second as the timing unit. When **n** is 1, the time for which the button switch has been turned ON is multiplied by 10, and **D** uses 100 milliseconds as the timing unit. When **n** is 2, the time for which the button switch has been turned ON is multiplied by 100, and **D** uses 10 milliseconds as the timing unit. The greater the value in **n**, the higher the timing resolution.

n	D
K0 (unit: 1 second)	1×T
K1 (unit: 100 milliseconds)	10×T
K2 (unit: 10 milliseconds)	100×T

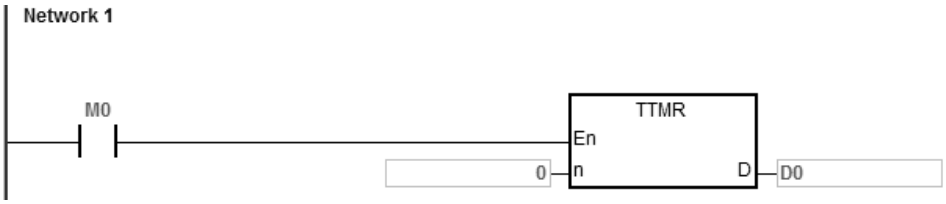
4. When you use on-line editing, reset the conditional contact to initialize the instruction.

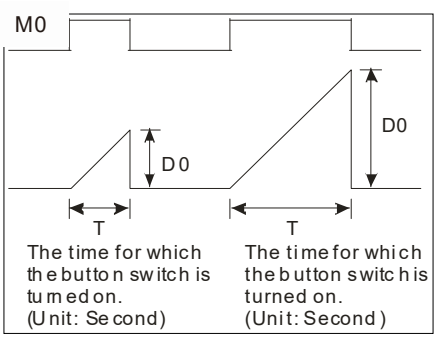
5. The value in **n** must be between 0–2.

Example 1

1. The instruction multiplies the time for which the button switch M0 has been turned ON by n, and stores the product in D0. You can use the button switch (ON) to record the time.

2. When M0 is switched OFF, the value in D0 is unchanged.





Additional remarks

1. If **D**+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **n** is not between 0–2, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

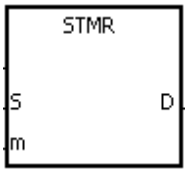
API	Instruction code	Operand	Function
0702	STMR	**S, m, D**	Special timer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					○											
m								●	●		○		○	○		
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S											●		
m		●			●	●							
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



S : Timer number (T0-T511)

m : Setting value of the timer

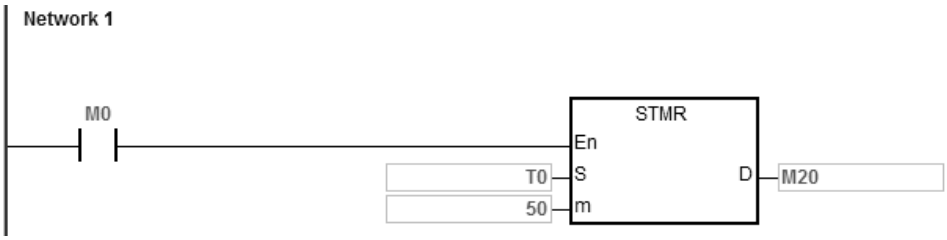
D : Output device

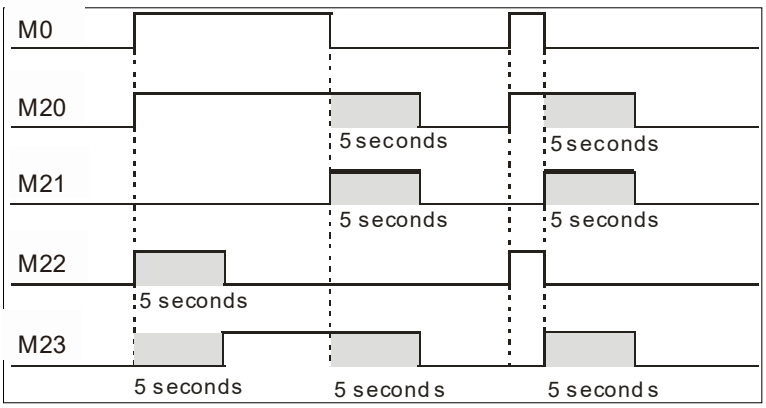
Explanation

1. This instruction generates timing for the off-delay relay, the one-shot circuit, and the flashing circuit.
2. This instruction uses 100 milliseconds as the timing unit. If the setting value for **m** is 50, the time value is 5 seconds.
3. You cannot use this timer repeatedly.
4. **D** occupies four consecutive devices.
5. Before the instruction is executed, reset **D**-**D**+3.
6. When the conditional contact is not enabled and the value of the device meets one of the two conditions mentioned below, **D**, **D**+1, and **D**+3 are ON for **m** seconds before they are switched OFF. When the conditional contact is not enabled and the value of the device does not meet either of the two conditions mentioned below, **D**-**D**+3 keep OFF.
   - The value of the timer is less than or equal to **m**, **D** is ON, and **D**+1 is OFF.
   - The value of the timer is less than **m**, **D** +2 is OFF, and **D**, **D**+1, and **D**+3 are ON.
7. When the on-line editing is used, Reset the conditional contact to initialize the instruction.
8. The value in **m** must be between 1–32767.

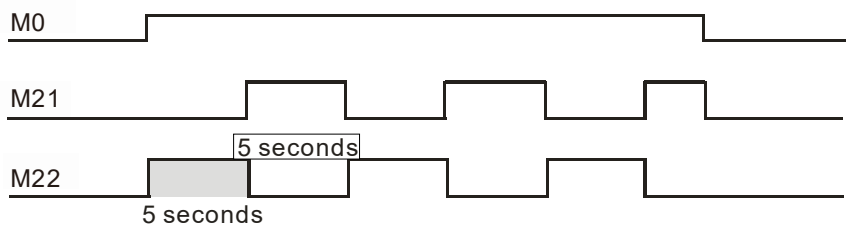
Example

1. When M0 is ON, the instruction specifies the timer T0, and the setting value of T0 is five seconds.
2. M20 is the off-delay contact. When M0 switches to ON, M20 is ON. Five seconds after M0 switches to OFF, M20 is OFF.
3. When M0 switches to OFF, M21 is ON for five seconds.
4. When M0 switches ON, M22 is ON for five seconds.
5. Five seconds after M0 switches to ON, M23 is ON. Five seconds after M0 switches to OFF, M23 is OFF.





6. When the conditional contact M0 is followed by the b contact M23, the flasher circuit passes through M21 and M22. When M0 is switched OFF, M20, M21, and M23 are switched OFF, and T0 is reset to 0.



Additional remarks

1. If **D**+3 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **m** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If you declare the operand **D** in ISPSoft, the data type is ARRAY [4] of BOOL.

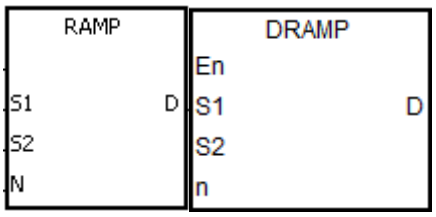
API	Instruction code			Operand	Function
0703	D	RAMP		**S_1, S_2, D, n**	Cyclic ramp signal

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○	○	○	○		
S_2							●	●	●		○	○	○	○		
D							●	●								
n							●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	ES3

Symbol



S_1 : Initial value of the ramp signal

S_2 : Final value of the ramp signal

D : Duration of the ramp signal

n : Number of scan cycles

Explanation

1. This instruction gets the linear slope, which has an absolute relationship with the scan time. Therefore it is suggested that you set a fixed scan time or write this instruction in a timer interrupt task.

2. You write the initial value and final value of the ramp signal into **S_1** and **S_2** respectively in advance. When M0 is ON, **D** increases from the setting value in **S_1** to the setting value in **S_2**. The number of scan cycles is stored in **D**+1. When the value in **D** is equal to that in **S_2**, or when the value in **D**+1 is equal to **n** (to the number of scan cycles), then SM687 is ON.

3. When the conditional contact is not enabled, the value in **D**, and **D**+1 are both 0, and SM687 is OFF.

4. When using on-line editing, Reset the conditional contact to initialize the instruction.

5. Refer to the ISPSoft User Manual for more information on setting a fixed scan time.

6. The value of **n** must be between 1–32767. When **n** is out of range, this instruction is not executed.

7. Only the 32-bit instructions can use the 32-bit counter, but not the device **E**.

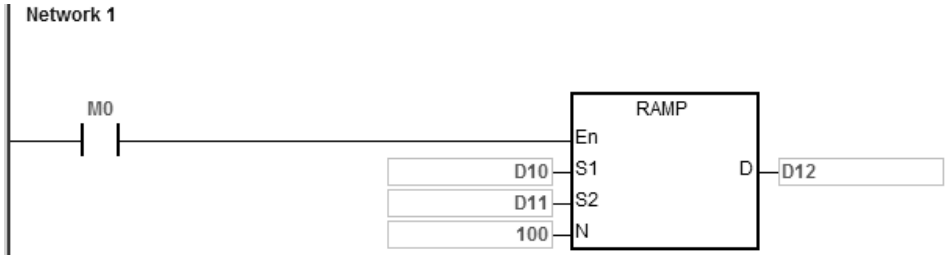
8. Use the SM686 flag to reset the value in **D** to 0. Refer to the examples below for details.

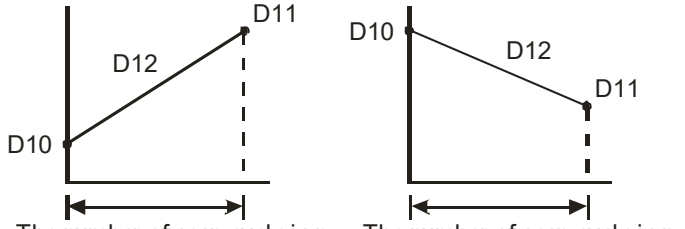
Example

When you use the instruction with an analog signal output, it acts to cushion the starting and stopping of the machinery.

1. During execution, when M0 switches to OFF, the execution of the instruction stops. When M0 switches to ON again, SM687 is OFF, D12 is reset to the setting value in D10, D13 is reset to 0, and the ramp calculation is restarted.

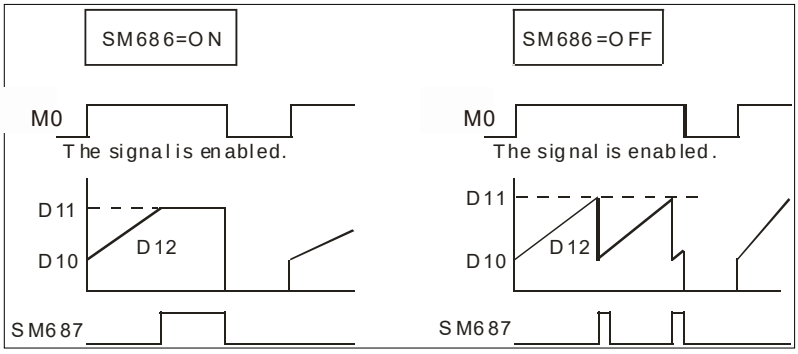
2. During execution, SM686 is OFF, and when D12 reaches the setting value in D11, SM687 switches to ON as a scan cycle. When D12 resets to the setting value in D10, D13 resets to 0.





The number of scan cycle is stored in D13.

3. When SM686 is ON, and D12 reaches the setting value in D11, the value in D12 is not reset to 0, and SM687 is ON. As long as the conditional contact is closed (ON), the value in D12 resets to 0 and SM687 is OFF. When SM686 is ON or OFF, the value in D12 changes as shown below.



Additional remarks

1. If **D**+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **n** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. For the 16-bit instruction, if you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of WORD/INT.
4. For the 32-Bit instruction, if you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of DWORD/DINT.

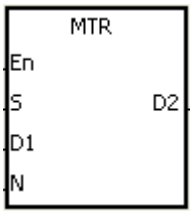
API	Instruction code	Operand	Function
0704	MTR	**S, D_1, D_2, n**	Matrix input

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	○															
D_1		○														
D_2		○	○	○				○								
n								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D_1	●												
D_2	●												
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



S : First input device in the matrix scan

D_1 : First output device in the matrix scan

D_2 : First corresponding device in the matrix scan

n : Number of rows to scan

Explanation

1. This instruction scans and stores the states of eight sequential input devices. **S** specifies the first input device in the matrix scan.

2. **D_1** specifies the transistor output device Y as the first device in the matrix scan. When the conditional contact is OFF, the states of the **n** devices starting from **D_1** are OFF.

3. One row of inputs is refreshed every scan cycle. There are 16 inputs in a row, and the scan starts from the first row and goes to the **n^{th}** row.

4. The eight input devices starting from the device specified by **S** are connected to the **n** output devices starting from the device specified by **D_1** to form the **n** rows of switches. The matrix scan reads the states of the **n** rows of switches, and stores the states in the devices starting from the device specified by **D_2**.

5. You can connect up to 8 rows of input switches in parallel to get 64 inputs (8×8=64).

6. The interval between executions of this instruction should be longer than the time it takes for the states of the I/O

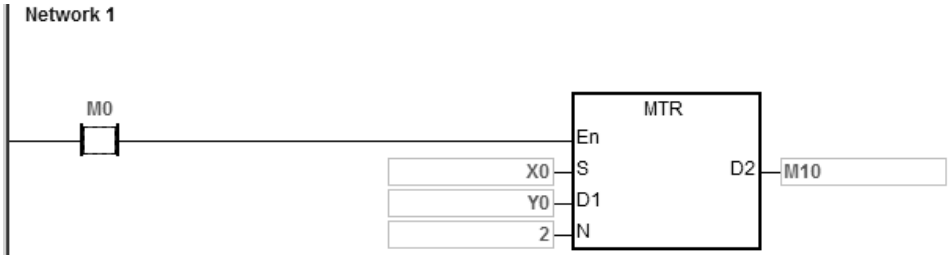
points on the module to be refreshed. Otherwise, the instruction cannot read the correct states of the inputs. See Additional remarks, below.

7. In general, the conditional contact used in the instruction is SM400: the flag is always ON when CPU runs.

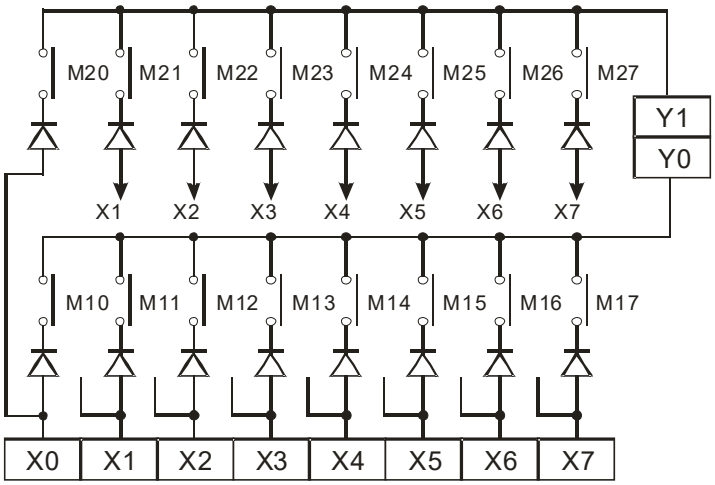
8. The value in **n** must be between 2–8.

Example 1

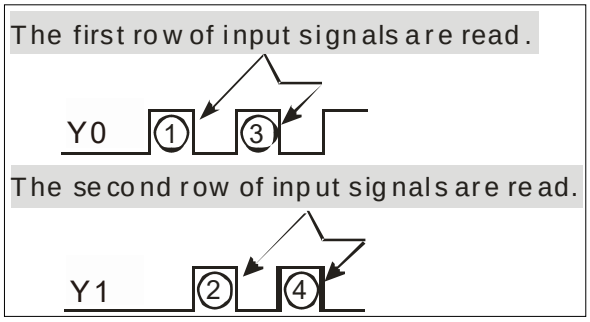
1. When M0 is ON, the MTR instruction is executed. The instruction reads the states of the two rows of switches in order, and stores them in the internal relays M10–M17 and M20–M27 respectively.



2. The diagram below is the external wiring diagram of the 2-by-8 matrix input circuit for X0–X7 and Y0–Y7. The corresponding internal relays of the 16 switches are M10–M17 and M20–M27.



3. The instruction connects eight input devices starting from M0 to the two output devices starting from M1 to form the two rows of switches. The matrix scan reads the states of the two rows of switches, and stores the states in the devices starting from M10 (specified by $\mathbf{D_2}$). That is, it stores the states of the first row of switches in M10–M17, and stores the states of the second row of switches in M20–M27.



Additional remarks

1. When this instruction is executed, a cycle time that is too long or a too short causes the state of the switches to be read incorrectly. Use the following tips to solve this issue.

   - When the scan cycle is too short, the I/O may not be able to respond in time and the correct states of the inputs cannot be read. You can set a fixed scan time to solve this issue.
   - When the scan cycle is too long, the switch may be slow to react. You can write this instruction in a timer interrupt task to set a fixed time to execute this instruction.

2. If **S**+7, $\mathbf{D_1}$+**n**-1, or $\mathbf{D_2}$+(**n***8)-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is not between 2–8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If you declare the operand **S** in ISPSoft, the data type is ARRAY [8] of BOOL.

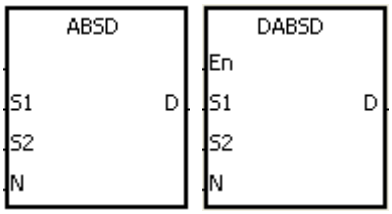
API	Instruction code			Operand	Function
0705	D	ABSD		**S_1, S_2, D, n**	Absolute drum sequencer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							
S_2					●	●	●	●	●							
D		●	●	●				●								
n								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D	●												
n		●	●		●	●	●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	ES3

Symbol



S_1 : Initial device for the comparison

S_2 : Comparison value

D : Comparison result

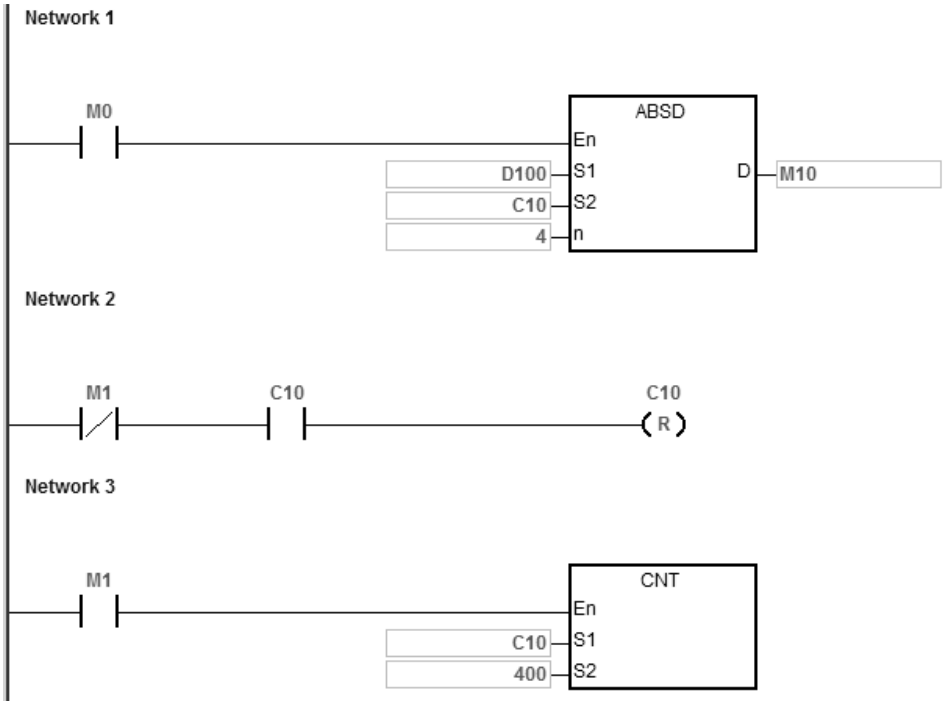
n : Number of comparison groups

Explanation

1. Use this instruction to generate multiple pulses corresponding to the current values of the counter.
2. Only the DABSD instruction can use the 32-Bit counter, but not the device E.
3. When using the ABSD instruction, **n** must be between 1–256.

Example 1

1. Before the ABSD instruction is executed, the MOV instruction writes the setting values in D100–D107. The values in the even devices are minimum values, and the values in the odd devices are maximum values.
2. When M0 is ON, the instruction compares the current value of the counter C10 with the maximum values and the minimum values in D100–D107, and stores the comparison results in M10–M13.
3. When M0 is OFF, the original states of M10–M13 are unchanged.

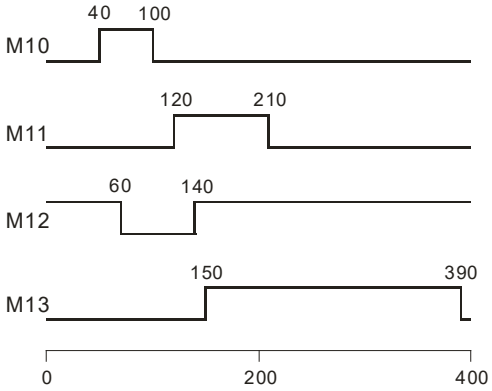


4. When the current value of C10 is between the minimum value and the maximum value, M10–M13 are ON. Otherwise, M10–M13 are OFF.

Minimum value	Maximum value	Current value of C10	Output
D100=40	D101=100	40≦C10≦100	M10=ON
D102=120	D103=210	120≦C10≦210	M11=ON
D104=140	D105=170	140≦C10≦170	M12=ON
D106=150	D107=390	150≦C10≦390	M13=ON

5. Suppose the minimum value is larger than the maximum value. When the current value of C10 is less than the maximum value (C10 < 60), or when the current value of C10 is larger than the minimum value (C10 > 140), M12 is ON. Otherwise, M12 is OFF.

Minimum value	Maximum value	Current value of C10	Output
D100=40	D101=100	40≦C10≦100	M10=ON
D102=120	D103=210	120≦C10≦210	M11=ON
D104=140	D105=60	60≦C10≦140	M12=OFF
D106=150	D107=390	150≦C10≦390	M13=ON



Additional remarks

1. For the 16-bit instruction, if **S**+2*__n__-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. For the 32-bit instruction, if **S**+4*__n__-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. For the 16-bit instruction, if **D**+__n__-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. For the 32-bit instruction, if **D**+2*__n__-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. For both the 16-bit instruction and the 32-bit instruction, if **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

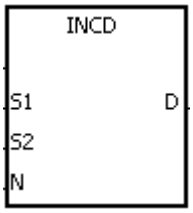
API	Instruction code	Operand	Function
0706	INCD	**S_1, S_2, n, D**	Incremental drum sequencer

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
D		●	●	●				●								
n								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												
n		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



S_1 : Initial device for comparison

S_2 : Counter number

D : Comparison result

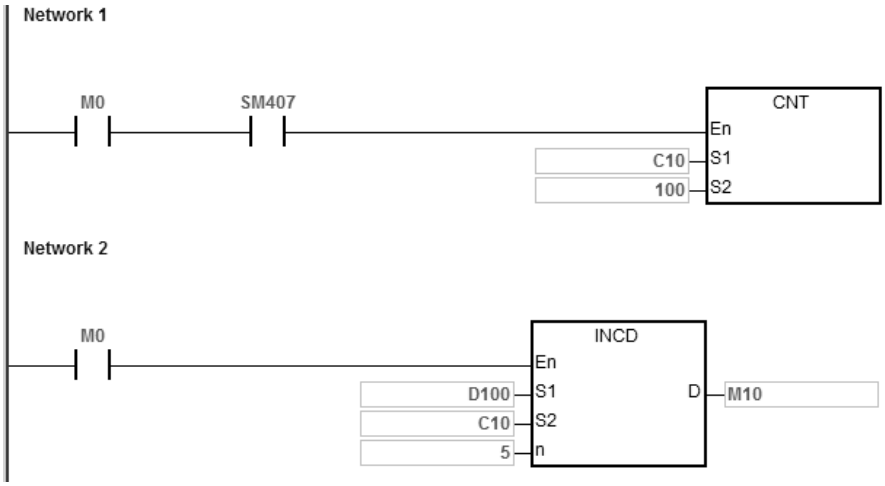
n : Number of comparison groups

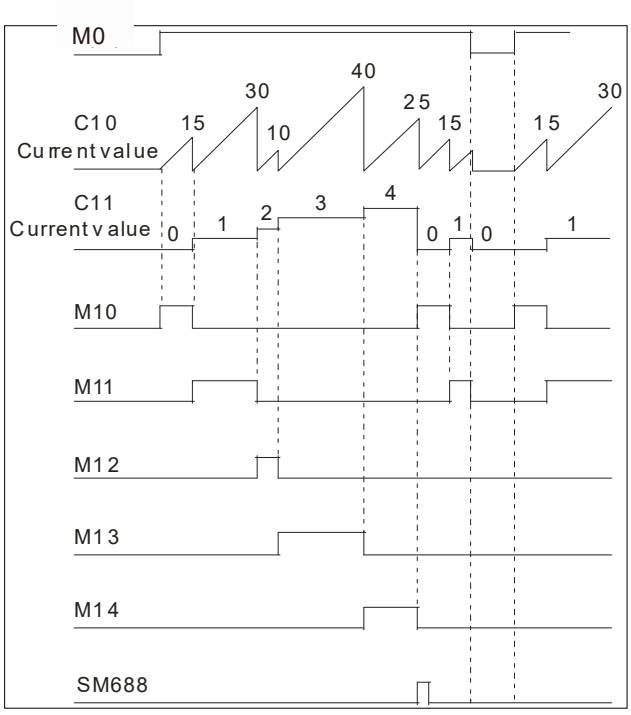
Explanation

1. This instruction generates multiple pulses for a pair of counters.
2. The instruction compares the current value of **S_2** with the setting value in **S_1**. When the current value matches the setting value, the instruction resets the current value of **S_2** to 0, and the stores the current comparison group number in **S_2**+1.
3. After the comparison between the current values of **S_2** and the **n** groups of values is complete, SM688 is ON for a scan cycle.
4. When the conditional contact is not enabled, the value in **S_2** is 0, the value in **S_2**+1 is 0, **D**-**D**+**n**-1 are OFF, and SM688 is OFF.
5. When using on-line editing, Reset the conditional contact to initialize the instruction.
6. The value in **n** must be between 1–256.

Example

1. Before the INCD instruction is executed, the MOV instruction writes the setting values in D100–D104. The values in D100–D104 are 15, 30, 10, 40, and 25 respectively.
2. The instruction compares the current values in C10 with the setting values in D100–D104. When the current value matches the setting value, the instruction resets C10 to 0, and counts again.
3. The instruction stores the current comparison group number in C11.
4. When the value in C11 changes by one, M10–M14 act correspondingly. Referto the following timing diagram.
5. When the comparison between the current values in C10 and the values in D100–D104 is complete, SM688 is ON for a scan cycle.
6. When M0 is switched from ON to OFF, C10 and C11 are reset to 0, and M10–M14 are switched OFF. When M0 switches to ON again, the instruction execution starts from the beginning.





Additional remarks

1. If S_2+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If S_1+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
5. If you declare the operand S_2 in ISPSoft, the data type is ARRAY [2] of WORD/INT.

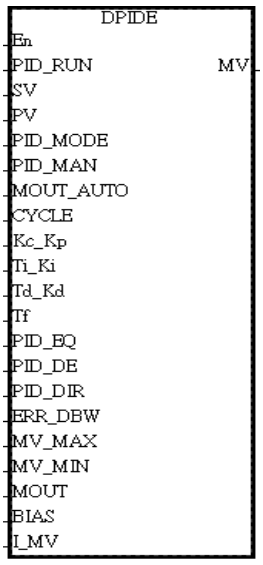
API	Instruction code			Operand	Function
0708	D	PIDE		**As shown in the following table**	PID algorithm

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
PID_RUN	●	●	●	●				●								
SV								●								○
PV								●								○
PID_MODE								●					○	○		
PID_MAN	●	●	●	●				●								
MOUT_AUTO	●	●	●	●				●								
CYCLE								●					○	○		
KC_Kp								●								
Ti_Ki								●								
Td_Kd								●								
Tf								●								○
PID_EQ	●	●	●	●				●								
PID_DE	●	●	●	●				●								
PID_DIR	●	●	●	●				●								
ERR_DBW								●	●							○
MV_MAX								●	●							○
MV_MIN								●	●							○
MOUT								●								
BIES3								●	●							○
I_MV								●								
MV								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING

Pulse instruction	16-Bit instruction	32-Bit instruction
-	-	ES3

Symbol



EN	:	Enable/Disable the instruction
PID_RUN	:	Enable the PID algorithm
SV	:	Target value (SV)
PV	:	Process value (PV)
PID_MODE	:	PID control mode
PID_MAN	:	PID Auto/Manual mode
MOUT_AUTO	:	Manual/Auto output value
CYCLE	:	Sampling time (CYCLE)
Kc_Kp	:	Proportional gain
Ti_Ki	:	Integral coefficient (sec. or 1/sec)
Td_Kd	:	Derivative coefficient (sec)
Tf	:	Derivate-action time constant (sec)
PID_EQ	:	PID formula types
PID_DE	:	Calculation of the PID derivative error
PID_DIR	:	PID forward/reverse direction (PID_DIR)
ERR_DBW	:	Range within which the error value is counted as 0
MV_MAX	:	Maximum output value (MV_MAX)
MV_MIN	:	Minimum output value (MV_MIN)
MOUT	:	Manual output value (MOUT)
BIES3	:	Feed forward output value
I_MV	:	Accumulated integral value
MV	:	Output value (MV)

Explanation

1. This instruction implements the PID algorithm. After the sampling time is reached, the instruction applies PID algorithm. PID stands for Proportional, Integral, Derivative. The PID control is widely applied to mechanical, pneumatic, and electronic equipment.
2. The parameter settings are listed in the following table.

Operand	Data type	Function	Setting range	Description
PID_RUN	BOOL	Enabling the PID algorithm	True: use the PID algorithm. False: reset the output value (MV) to 0, and stop using the PID algorithm.	
SV	REAL	SV	Range of single-precision floating-point numbers	Target value
PV	REAL	PV	Range of single-precision floating-point numbers	Process value
PID_MODE	DWORD/DINT	PID control mode	0: Automatic control When PID_MAN switches from True to False, invoke the output value (MV) in the automatic algorithm. 1: Auto tuning the parameters for the temperature control. After tuning is done, the system is in auto control mode (PID_MODE is set to 0) and fill in the appropriate parameters (Kc_Kp, Ti_Ki, Td_Kd, and Tf) Note: when the mode is set to 1, auto tuning the parameter, you cannot use numerical value to set up.	
PID_MAN	BOOL	PID A/M mode	True: Manual Output the MV according to	

Operand	Data type	Function	Setting range	Description
			MOUT, but it is still between MV_MIN and the MV_MAX. This setting has no effect when PID_MODE is set to 1. False: Automatic Output the MV according to the PID algorithm, and the output value is between MV_MIN and MV_MAX.	
MOUT_AUTO	BOOL	MOUT automatic change mode	True: Automatic MOUT varies with the MV. False: Normal MOUT does not vary with the MV.	
CYCLE	DWORD/DINT	Sampling time (T_S)	1–40,000 (unit: ms)	When the instruction is scanned, use the PID algorithm according to the sampling time, and refresh MV. The PLC requires that the instruction execute; it will not run the sampling time automatically. If T_S is less than 1, it is counted as 1. If T_S is larger than 40,000, it is counted as 40,000. When using the PID instruction in an interval interrupt task, the sampling time is the same as the interval between the timed

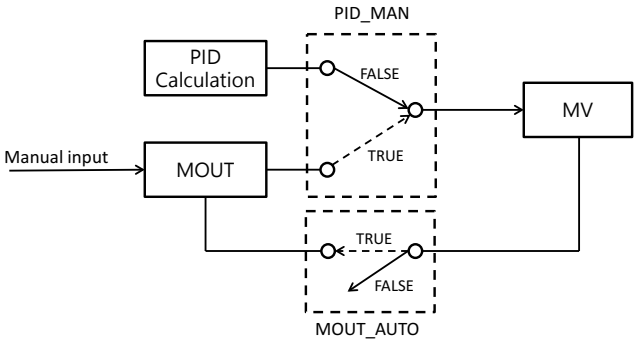
Operand	Data type	Function	Setting range	Description
				interrupt tasks. The sampling cycle setting of the sampling cycle is ignored here.
Kc_Kp	REAL	Calculated proportional coefficient (Kc or Kp, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers	Calculated proportional coefficient (Kc or Kp) If the P coefficient is less than 0, the Kc_Kp is 0. Independently, if Kc_Kp is 0, it is not controlled by P.
Ti_Ki	REAL	Integral coefficient (Ti or Ki, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers (unit: Ti = sec; Ki = 1/sec)	If the calculated coefficient I is less than 0, Ti_Ki is 0. If Ti_Ki is 0, it is not controlled by I.
Td_Kd	REAL	Derivative coefficient (Td or Kd, according to the settings in PID_EQ)	Range of positive single-precision floating-point numbers (unit: sec)	If the calculated coefficient D is less than 0, Td_Kd is 0. If Ti_Ki is 0, it is not controlled by D.

Operand	Data type	Function	Setting range	Description
Tf	REAL	Derivate-action time constant	Range of positive single-precision floating-point numbers (unit: sec)	If the derivate-action time constant is less than 0, Tf is 0 and it is not controlled by the derivate-action time constant (derivative smoothing).
PID_EQ	BOOL	PID formula types	TRUE: dependent formula FALSE: independent formula	
PID_DE	BOOL	The calculation of the PID derivative error	TRUE: use the variations in the PV to calculate the control value of the derivative (Derivative of the PV). FALSE: use the variations in the error (E) to calculate the control value of the derivative (derivative of the error).	
PID_DIR	BOOL	PID forward/reverse direction	True: reverse action (E=SV-PV) False: forward action (E=PV-SV)	
ERR_DBW	REAL	Range within which the error value is counted as 0.	Range of single-precision floating-point numbers	The error value (E) is the difference between the SV and the PV. When the setting value is 0, the function disabled; otherwise the CPU module checks whether the present error is less than the absolute value of ERR_DBW, and checks whether the present error meets the cross

Operand	Data type	Function	Setting range	Description
				status condition. If the present error is less than the absolute value of ERR_DBW, and meets the cross status condition, the present error is counted as 0, and the PLC applies the PID algorithm ; otherwise the present error is brought into the PID algorithm according to the normal processing.
MV_MAX	REAL	Maximum output value	Range of single-precision floating-point numbers	Suppose **MV_MAX** is set to 1,000. When MV is larger than 1,000, 1,000 is the output. The value in **MV_MAX** should be larger than that in **MV_MIN**. Otherwise, the maximum MV and the minimum MV are reversed.
MV_MIN	REAL	Minimum output value	Range of single-precision floating-point numbers	Suppose **MV_MIN** is set to -1,000. When the MV is less than -1,000, -1,000 is the output.
MOUT	REAL	MV	Range of single-precision	When set to PID Manual, the MV value is output as the setting

Operand	Data type	Function		Setting range	Description
				floating-point numbers	value for MOUNT, between MV_MAX and MV_MIN.
BIES3	REAL	Feed forward output value		Range of single-precision floating-point numbers	Feed forward output value, used for the PID feed forward.
I_MV (occupies 15 consecutive DWord devices)	REAL	I_MV	Accumulated integral value	Range of single-precision floating-point numbers	Accumulated integral value temporarily stored, and usually for reference. You can still clear or modify it according to your needs. When the MV is greater than the MV_MAX, or when the MV is less than MV_MIN, the accumulated integral value in I_MV is unchanged.
		I_MV+1	The previous error value is temporarily stored here.		
		I_MV+2–I_MV+5	For system use only		
		I_MV+6	The previous PV is temporarily stored here.		
		I_MV+7–I_MV+14	For system use only		
MV	REAL	MV	The MV is between the MV_MIN and the MV_MAX.		

The diagram of switching to PID_MAN / MOUT_AUTO:



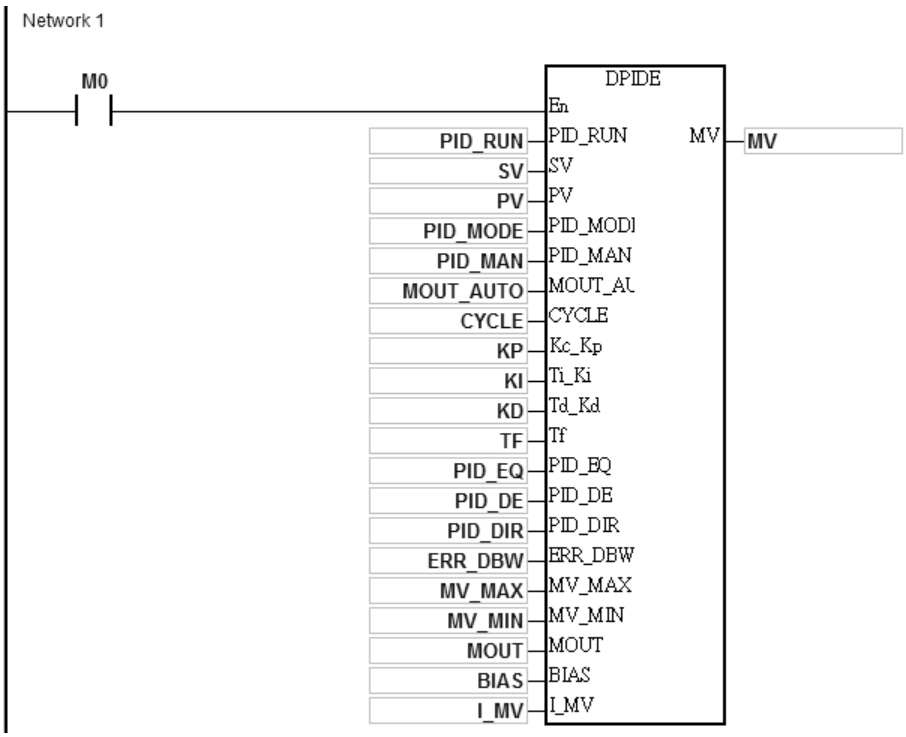
1. When switching the control mode **(PID_MAN=0)** from automatic to manual, you can set the flag MOUT_AUTO to 1 and the output value of MOUT goes along with the output value of MV. After switching to the manual mode **(PID_MAN=1)**, you can set the MOUT_AUTO to 0.

2. When **PID_RUN** changes from TRUE to FALSE, the PLC resets the value in MV to 0. When the value in MV is to be retained, you can set EN to FALSE to dismiss the instruction and to keep the output value in MV.

Example 1

1. Set all parameters before executing this instruction.

2. When M0 is ON, the instruction is executed. When PID_RUN is ON, the instruction applies the DPID algorithm. When PIC_RUN is OFF, MV is 0, and store the value in MV. When M0 switches to OFF, the instruction is not executed, and the previous data is unchanged.



Additional remarks

1. The instruction can be used several times, but the registers specified by **I_MV**–**I_MV**+14 cannot be the same.

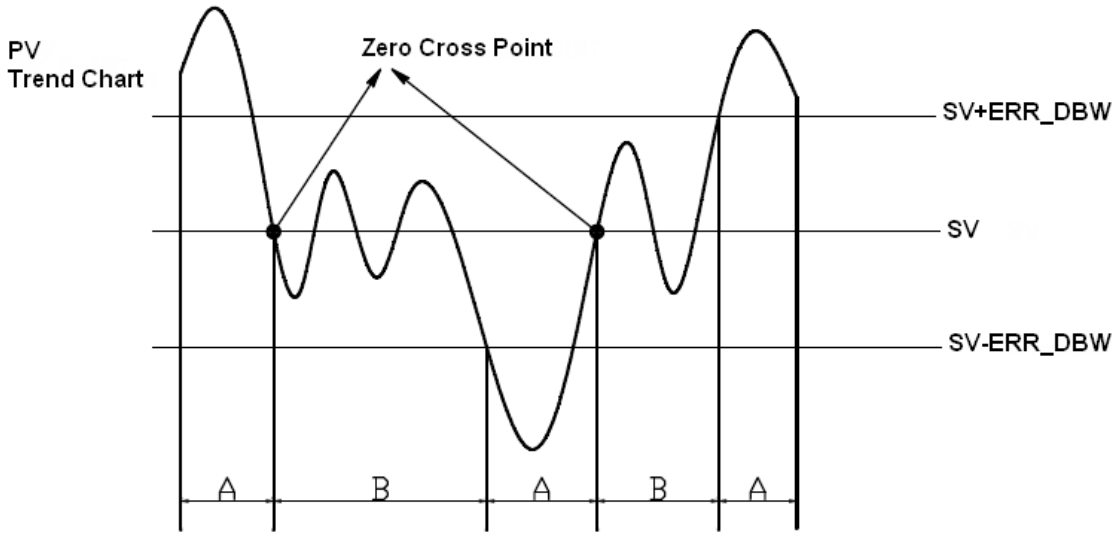
2. **I_MV** occupies 30 registers.

3. You can only use the 32-bit instruction in cyclic tasks and interval interrupt tasks. When using the 32-bit instruction in an interval interrupt task, the sampling time (Cycle) is the same as the interval between the timed interrupt tasks.

4. When the instruction is scanned, the 32-bit PID algorithm is applied according to the sampling time (Cycle), and it refreshes MV. When you use the instruction in an interrupt task, the sampling time (Cycle) is the same as the interval between the timed interrupt tasks. The PID algorithm is applied according to the interval between the timed interrupt tasks.

5. Before the 32-bit PID algorithm is applied, the process value used in the PID instruction has to be a stable value. When you need the input value in the module to implement the DPID algorithm, must note the time it takes for the analog input to be converted into the digital input.

6. When the PV (process value) is in the range of **ERR_DBW**, at the beginning, the present error is brought into the PID algorithm according to the normal processing, and then the CPU module checks whether the present error meets the cross status condition: PV (process value) goes beyond the SV (target value). Once the condition is met, the present error is counted as 0 when applying the PID algorithm. After the PV (process value) is out of the **ERR_DBW** range, the present error is brought into the PID algorithm again. If PID_DE is true, that means it uses the variations in the PV to calculate the control value of the derivative, and after the cross status condition is met, the PLC treats Δ **PV** as 0 to apply the PID algorithm. (Δ **PV**= current **PV** – previous **PV**). In the following example, the present error is brought into the PID algorithm according to the normal processing in section A ,and the present error or Δ **PV** is counted as 0 to apply the PID algorithm in the section B.



The PID algorithm:

1. When you set **PID_MODE** to 0, the PID control mode is the automatic control mode.

- **Independent Formula & Derivative of E** (**PID_EQ**=False & **PID_DE**=False)

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dE}{dt} + BIAS \quad E = SV - PV \text{ or } E = PV - SV$$

- **Independent Formula & Derivative of PV** (**PID_EQ=False & PID_DE=True**)

$$MV = K_P E + Ki\int_0^t Edt - K_d * \frac{dPV}{dt} + BIAS \quad E = SV - PV$$

Or

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dPV}{dt} + BIAS \quad E = PV - SV$$

- **Dependent Formula & Derivative of E** (**PID_EQ**=True & **PID_DE**=False)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS \quad E = SV - PV \text{ or } E = PV - SV$$

- **Dependent Formula & Derivative of PV** (**PID_EQ**=True & **PID_DE**=True)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt - T_d * \frac{dE}{dt}\right] + BIAS \quad E = SV - PV$$

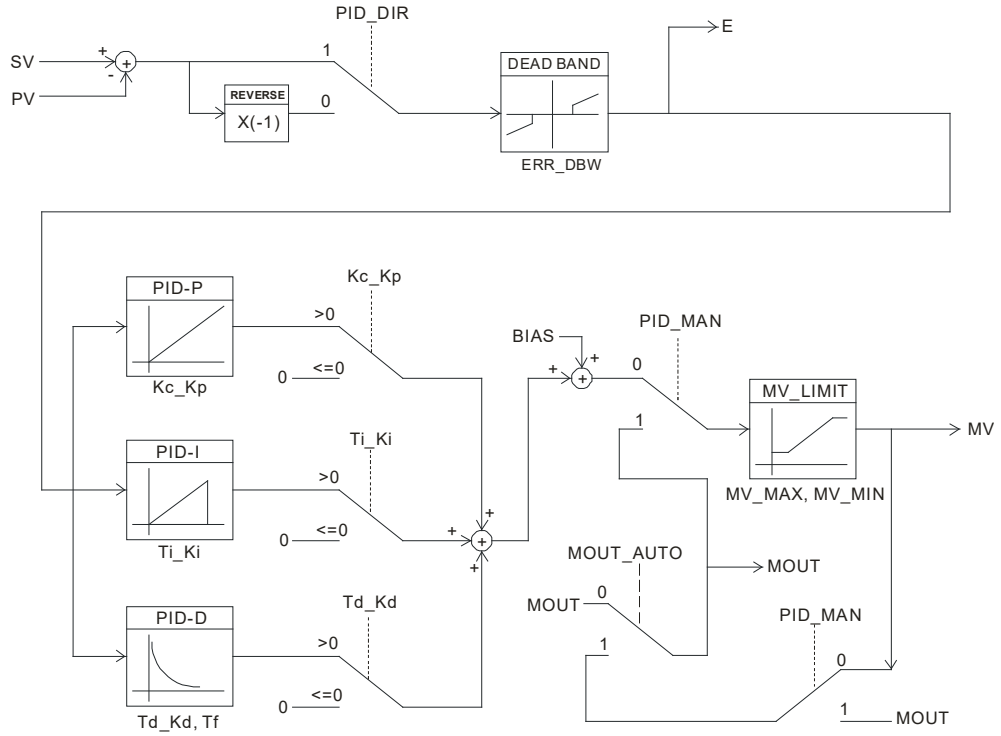
Or

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS \quad E = PV - SV$$

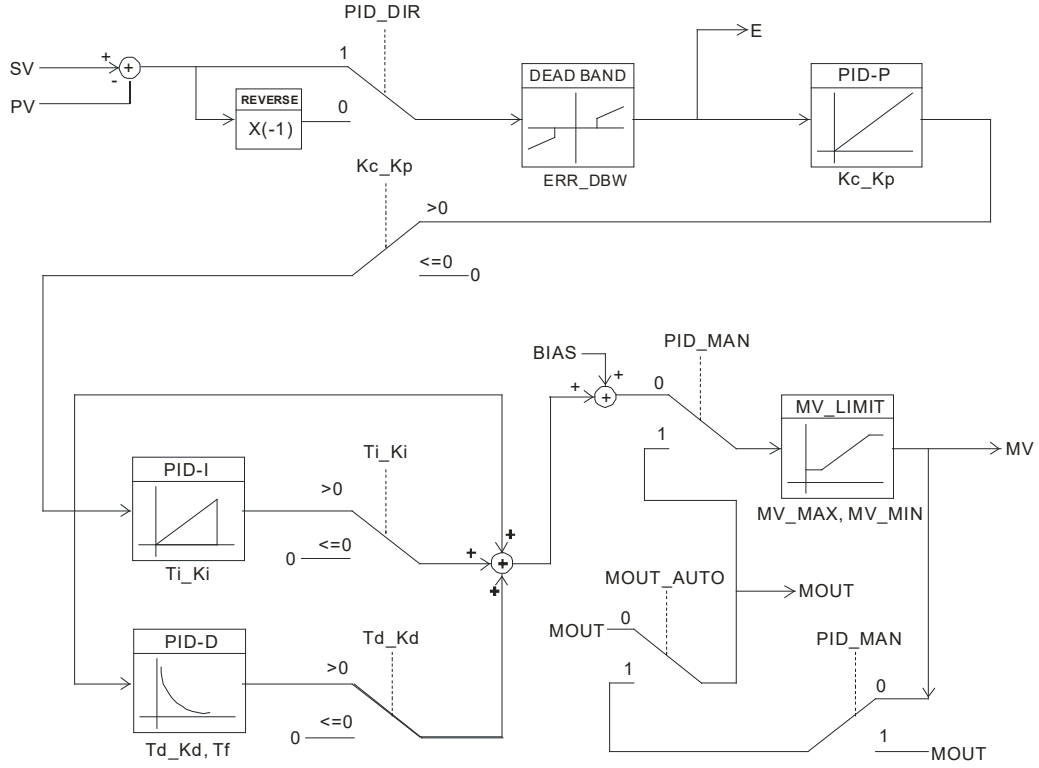
2. When you set **PID_MODE** to 1, the PID control mode is the automatic tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode then becomes the automatic control mode.

PID Block Diagram:

PID Block Diagram (Independent)



PID Block Diagram (Dependent)



Suggestions

1. Since you can use the 32-bit instruction in a lot of controlled environments, you must choose the appropriate control function. For example, to prevent improper control, do not use **PID_MODE** in the motor controlled environment when it is set to 1.

2. When you tune the parameters **Kc_Kp**, **Ti_Ki**, and **Td_Kd** (PID_MODE is set to 0), you must tune KP first (based on experience), and then set **Ti_Ki** and **Td_Kd** to 0. When you can handle the control, you can increase **Ti_Ki** and **Td_Kd**. When **Kc_Kp** is 1, it means that the proportional gain is 100%. That is, the error value is increased by a factor of one. When the proportional gain is less than 100%, the error value is decreased. When the proportional gain is larger than 100%, the error value is increased.

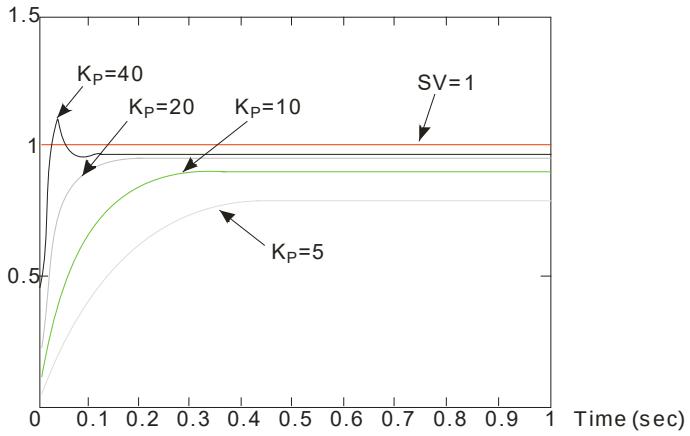
3. To prevent the parameters that have been tuned automatically from disappearing after a loss of power, you must store the parameters in the latched data registers when **PID_MODE** is set to 1. The parameters that have been automatically tuned are not necessarily suitable for every controlled environment. Therefore, you can modify the automatically tuned parameters; however, it is suggested that you only modify the **Ti_Ki** and the **Td_Kd.**

4. You can use this instruction with many parameters, but to prevent improper control, do not set the parameters randomly.

Example 2: Tuning the parameters used with the PID instruction

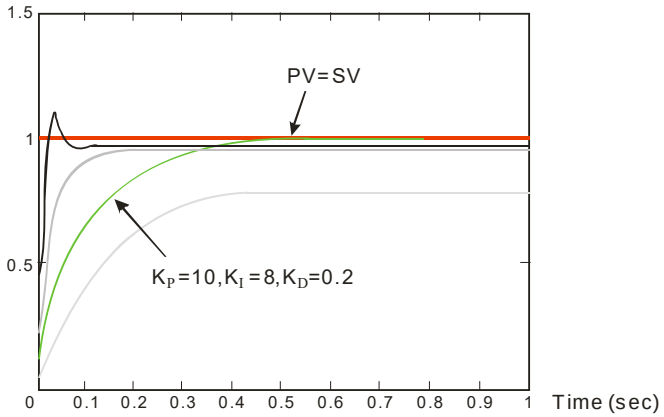
Suppose that the transfer function of the plant is the first-order function $G(s)=\frac{b}{s+a}$, the SV is 1, the sampling time Ts is 10 milliseconds. It is suggested that you follow these steps when tuning the parameters.

Step 1: First, set the K_I and the K_D to 0. Next, set the K_P to 5, 10, 20 and 40 successively, and record the target values and the process values. The results are shown in the following diagram.



Step 2: When the K_P is 40, there is overreaction. When the K_P is 20, the reaction curve of PV is close to SV, and there is no overreaction. However, due to the fast start-up, the transient output value (MV) is big. Neither 40 nor 20 is a suitable value. When the K_P is 10, the reaction curve of PV approaches SV smoothly. When K_P is 5, the reaction is too slow. Therefore, K_P = 10 is the best choice.

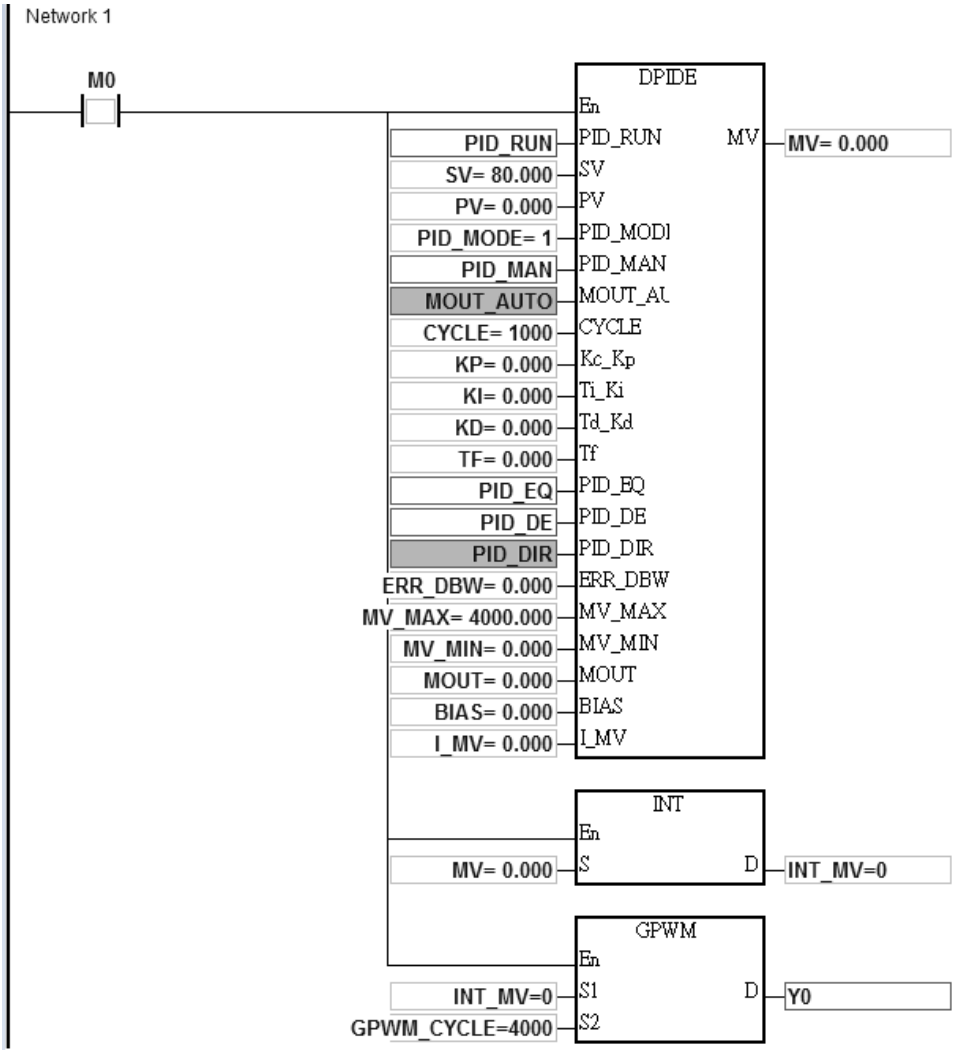
Step 3: After setting K_P to 10, increase K_I. For example, K_I is successively set to 1, 2, 4, and 8. K_I should not be larger than K_P. Then, increase K_D. For example, successively set K_D to 0.01, 0.05, 0.1, and 0.2. K_D should not be larger than ten percent of K_P. Finally, the relation between PV and SV is shown in the following diagram.



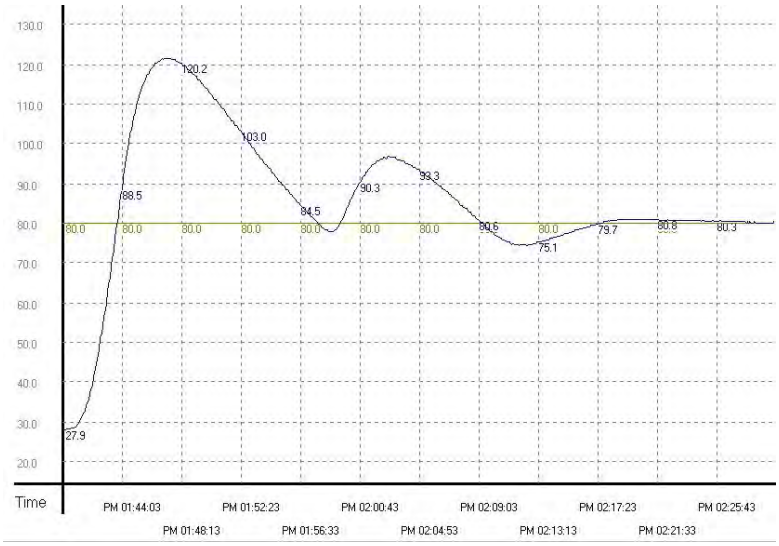
Note: This example is only for reference. You must tune the parameters properly according to the actual condition of the control system.

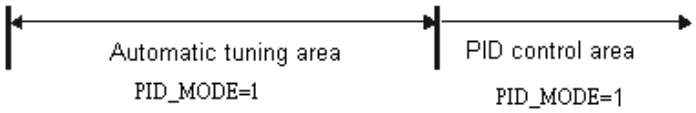
Example 3: Using the automatic tuning function to control the temperature

Because you may not be familiar with the characteristics of the temperature environment to be controlled, you can use the automatic tuning function to make an initial adjustment (**PID_MODE** is set to 1). After the automatic tuning of the parameter is complete, **PID_MODE** is set to 0. The controlled environment in this sample is an oven. The following example program shows the setting values for the instruction.

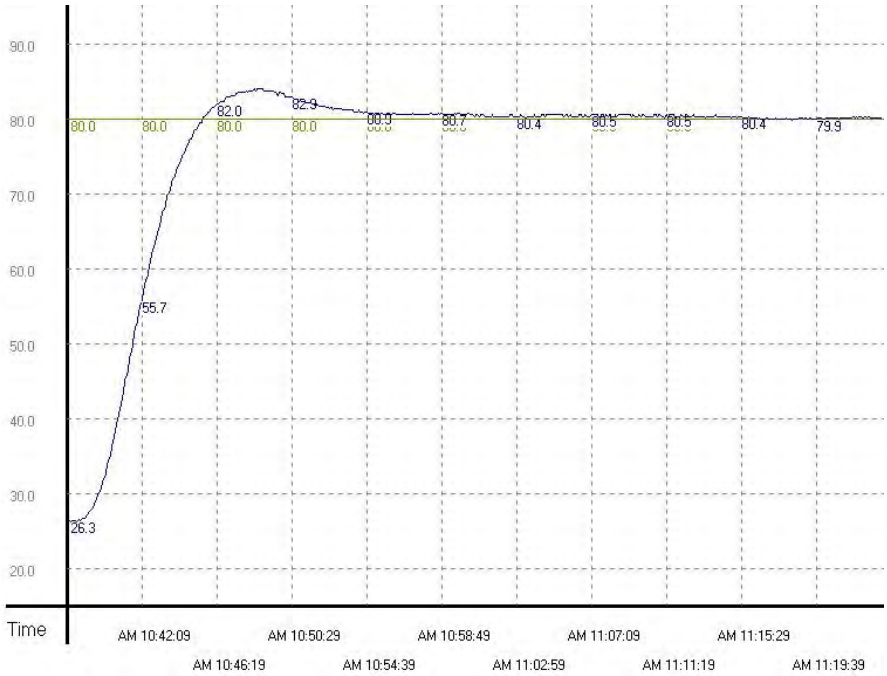


The experimental result of the automatic tuning function is shown in the following graph.

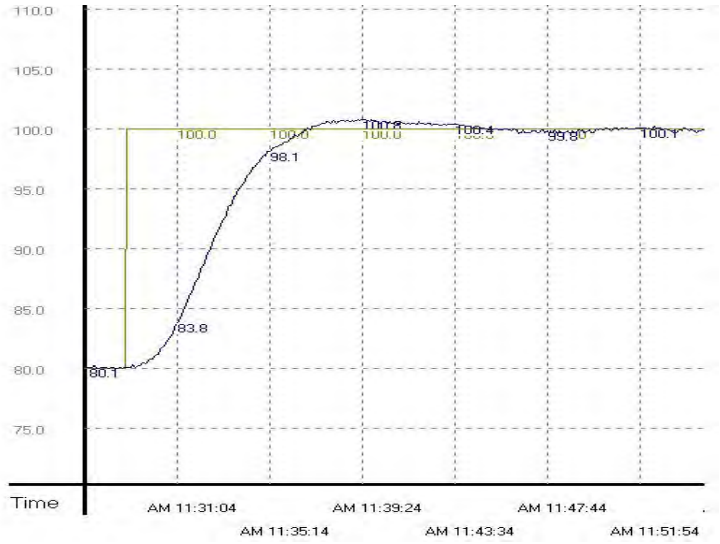




The following graph shows the result of using the automatically tuned parameters to control the temperature.



This graph shows that using automatically tuned parameters can result in a good temperature control result. It only takes about twenty minutes to control the temperature. The following graph shows the result of changing the target temperature from 80°C to 100°C.



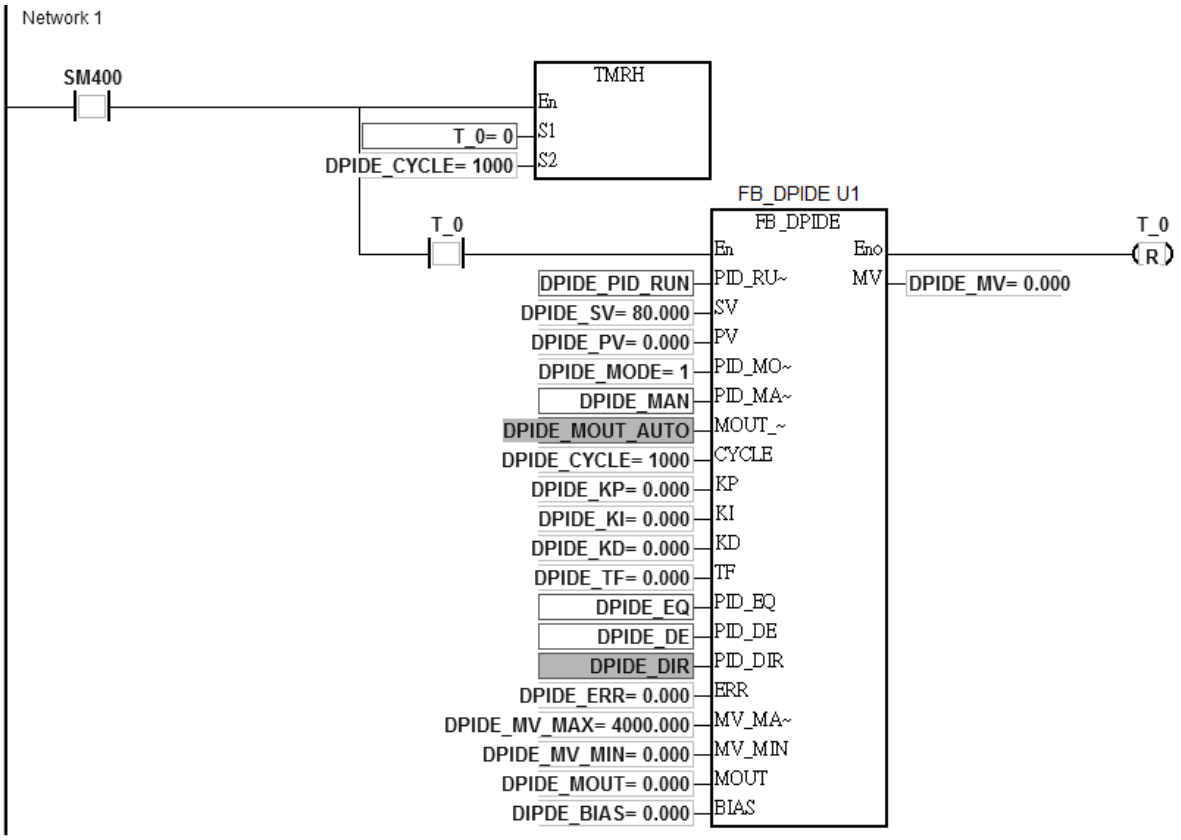
This graph shows that when the target temperature changes from 80°C to 100°C, the automatically tuned parameters still work to control the temperature in a reasonable amount of time.

Example 4: Creating a DPIDE instruction in a function block and setting to the cyclic task mode to read the function block written with a DPIDE instruction to control the temperature.

1. Set the value in DPIDE_CYCLE to 1000 ms, and execute the DPIDE instruction by reading the function block written with a DPIDE instruction. Whenever the function block is scanned, the PID algorithm is applied according to the sampling time (Cycle), and it refreshes the output value (DPIDE_MV).

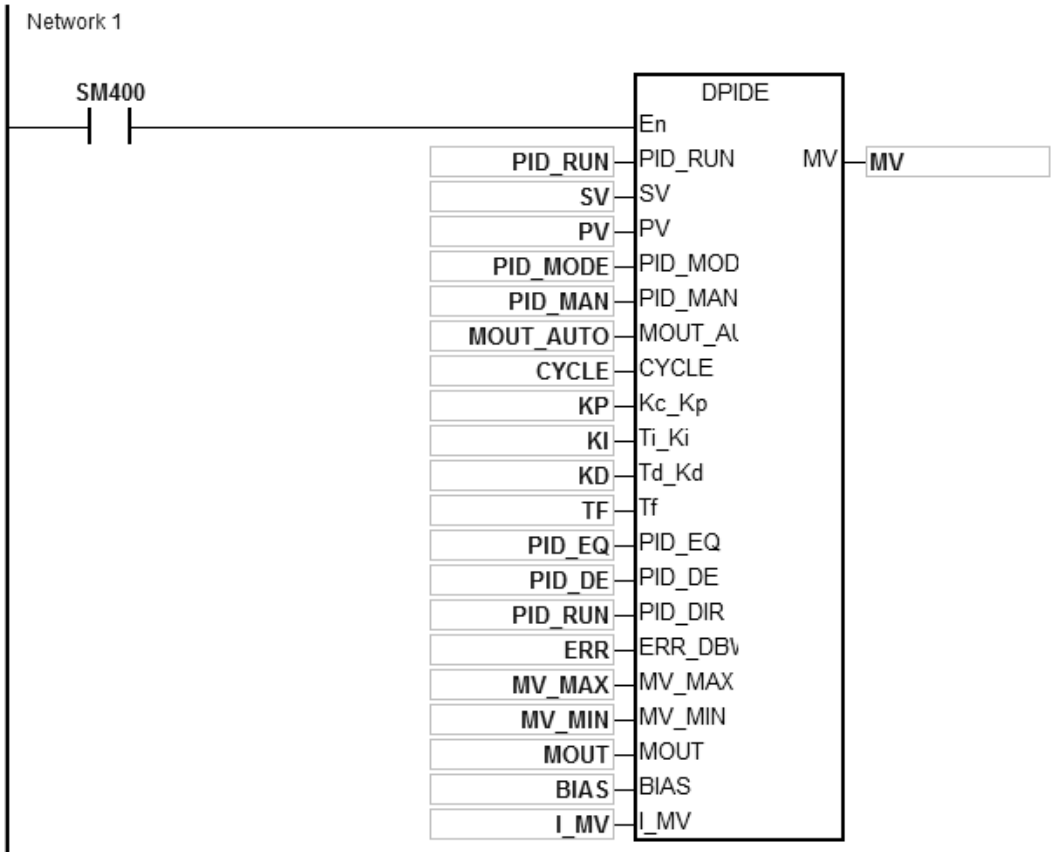
2. Set the DPIDE_MODE =1 for auto tuning the parameters for the temperature control. After tuning is done, the system is in auto control mode (PID_MODE is set to 0) and fill in the appropriate parameters (Kc_Kp, Ti_Ki, Td_Kd, and Tf).

3. Main program (cyclic task): Since PLC only executes the DPIDE instruction when it is scanned. If we use TMRH to work with the DPIDE instruction, for example, set the TMRH to 1000 ms, the system calls the function block written with a DPIDE instruction (ES3_DPIDE) every 1000 ms. See the example program below.



4. Function block (FB_DPIDE): Execute the DPIDE instruction by reading the function block written with a DPIDE instruction. Whenever the function block is scanned, the PID algorithm is applied according to the sampling time (Cycle), and it refreshes the output value (DPIDE_MV). (Refer to ISPSoft Manual for more details on how to create a function block.)

NOTE: The parameters PID_MODE, Kc_Kp, Ti_Ki, Td_Kd, and Tf in the function block written with a DPIDE instruction should be declared as VAR_IN_OUT.



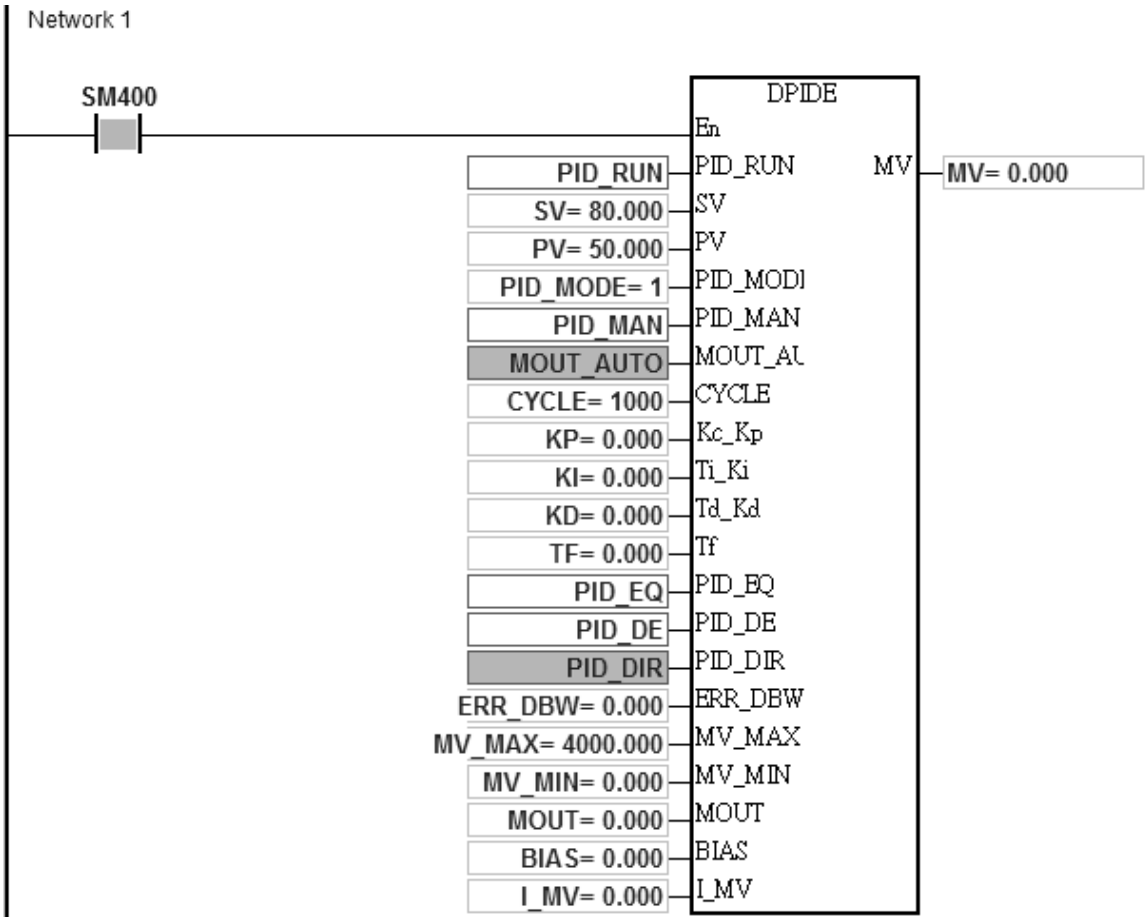
Example 5: Creating a DPIDE instruction in a time interrupt program to control the temperature. (Note: use the time interrupt as the cycle time of DPIDE.)

6

1. Set the time interrupt to 1000 ms in HWCONFIG.
2. Create a DPIDE instruction in a time interrupt program. Whenever a time interrupt occurs, the PID algorithm is applied. The setting in DPIDE_CYCLE is invalid here.
3. Set the DPIDE_MODE =1 for auto tuning the parameters for the temperature control. After tuning is done, the system is in auto control mode (PID_MODE is set to 0) and fill in the appropriate parameters (Kc_Kp, Ti_Ki, Td_Kd, and Tf).

Main program (cyclic task)

Time interrupt program I601 and the setting parameters



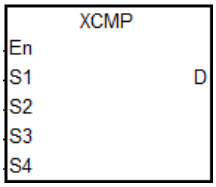
API	Instruction code			Operand	Function
0709		XCMP		**S_1, S_2, S_3, S_4, D,**	Setting up to compare the inputs of multiple work stations

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2							○									
S_3								○								
S_4								○								
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2												●	
S_3		●			●	●							
S_4			●				●						
D			●				●						

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



S_1 : Trigger input point

S_2 : High-speed counter number

S_3 : Setting for the numbers for work station and objects

S_4 : Reference value for comparison and the observational error

D : First corresponding device for the comparison result in the stack area

Explanation

1. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

2. Use **S_1** for setting the trigger input points; the high-speed inputs are X0–X7 & X10–X17 and the other inputs are general type. Executing the instruction enables the external interrupts for the inputs (M0–X0.15). Therefore do NOT use the inputs with interrupt tasks; otherwise, when the instruction is executed, the interrupts are disabled and resumed only after the instruction completes. The general type inputs are affected by the scan time though they are suitable for the environments where the inputs are not as stable.

3. **S_2** works with 32-bit counters (HC0–HC255). When the inputs are the high-speed trigger input type, implement the hardware high-speed counter and use the DCNT instruction to enable the counter. When you need high-speed output, use the DMOV instruction to copy the output current position; for example copying the axis of SR460 to HC0, (DMOV SR460 HC0).

4. **S_3** occupies seven consecutive 16-bit devices. **S_3**+0 is n (the work station number) and **S_3**+1 is m (the maximum object number). **S_3**+2 is the result of the object being filtered. The range for n and m is between 1–64. When this value is out of range, the value used is treated as the maximum (32) or the minimum (1). The range for **S_3**+2 (the number of filter) is between 0–32767. Zero is used for any value less than 0 ; and a value of 0 disables the filtering function. **S_3**+3 (Low Word) and **S_3**+4 (High Word) are for rising-edge values. **S_3**+5 (Low Word) and **S_3**+6 (High Word) are for falling-edge values (32-bit). Be sure to declare an array of 7 words or 7 consecutive word type variables.

5. Set the maximum number for **S_3**+1 (m). If m < n, note the objects and make sure they are sufficient on the production line.

6. **S_4** occupies 3xn consecutive 32-bit devices (6xn 16-bit devices). If the required space exceeds the range of device D, the instruction is not executed. The value of n is the work station number set in the operand **S_3**. The following table lists the functions for each device and the corresponding number for **S_4**. Be sure to declare an array of 3n double words or 3 consecutive double word type variables for **S**.

Function	Work station 1	Work station 2	• • •	Work station n
Reference value for comparison (32-bit)	**S_4**+0	**S_4**+2	• • •	**S_4**+(n-1)x2
Observational error of the compared stack area for the entering-work-station ones (32-bit)	**S_4**+2xn	**S_4**+2xn+2	• • •	**S_4**+(2xn-1)x2
Observational error of the compared stack area for the leaving-work-station ones (32-bit)	**S_4**+4xn	**S_4**+4xn+2	• • •	**S_4**+(4xn-1)x2

When you set the reference value to 0 for a specific work station, the specific work station stops working. You can use this technique to manage work stations.

7. **D** is the first corresponding device for the comparison result in the stack area. **D** occupies 2xn consecutive 16-bit devices and 2xmxn consecutive 32-bit devices (or 4xmxn consecutive 16-bit devices). If the required space exceeds the range of device D, the instruction is not executed. The following table lists the functions for each device and the corresponding number for **D**.

Function	Work station 1	Work station 2	• • •	Work station n
Value of the head index (16-bit)	**D**+0	**D**+1	• • •	**D**+(n-1)
Value of the tail index (16-bit)	**D**+n	**D**+(n+1)	• • •	**D**+(2xn-1)

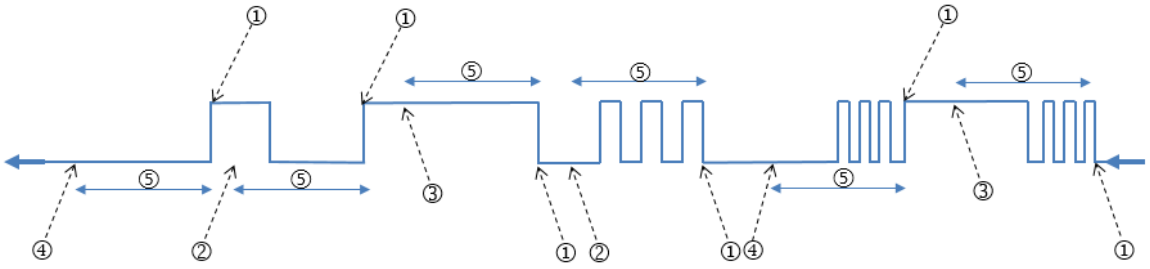
Function	Work station 1	Work station 2	・・・	Work station n
Compared stack area 1 for the entering -work-station ones (32-bit)	**D**+2xn	**D**+2xn+2	・・・	**D**+2xn+2(n-1)
Compared stack area 1 for the leaving-work-station ones (32-bit)	**D**+4xn	**D**+4xn+2	・・・	**D**+4xn+2(n-1)
:	:	:	:	:
Compared stack area m for the entering -work-station ones (32-bit)	**D**+4xmxn-2xn		・・・	**D**+4xmxn-2
Compared stack area m for the leaving-work-station ones (32-bit)	**D**+4xmxn		・・・	**D**+4xmxn+2 (n-1)

D tends to occupy more space in the stack area. If the required space exceeds the range of device **D**, the PLC only executes what is valid in the storage and does not show a no warning. It is suggested that you declare an array of 2xn+4xmxn words for **D**.

8. There is no limit on the number of times you can execute the instruction but only one execution can be done at a time.

9. It is suggested to use this instruction with the YOUT instruction (API 0710), and use the same first corresponding device for the comparison result in the stack area (**D**).

10. The following timing diagram shows executing the high-speed counter and filter (reading from right to left).



① PLC reads the current counter value and stores the value in a register.

② Drop the counter value: the number of filters read is less than the number of filters set.

③ Record the counter value to the compared stack area for the entering-work-station ones, when the signal is stable.

④ Record the counter value to the compared stack area for the leaving-work-station ones, when the signal is stable.

⑤ The number of filters

11. When the signal is rising- or falling-edge triggered, and the PLC completes processing the filters, the PLC reads the high-speed counter value and adds one in the value of the head index. The PLC then records the entering and leaving counter results for each work station. The compared counter result is the current counter value + reference value + observational error. For either rising- or falling-edge triggered, the value of the head index is incremented. The maximum value for the head index mx2 (the maximum number of objects).

12. The value of the head index is cyclically incremented, when the signal is rising- or falling-edge triggered and completes processing the number of filters (the default for trigger input is OFF). The maximum value for the head index is mx2 (the maximum number of objects). For example, if you set the number of objects to 10, the value of the head index (default: 0) is incremented to 1, 2, 3 to 20 and then 1, 2, 3 to 20 repeatedly. When the value of the head index is 0, it means no object has entered after executing the instruction. The PLC adds one to the value of the head index, and then checks the value of the tail index. If the value (after adding one) in the value of the head index equals the value of the tail index, the PLC cancels the addition and records the counter result.

13. When the instruction is executed and the state of the initial input is **OFF**, the **rising-edge trigger** corresponds to the **odd numbers** of the head index value, and the ***falling-edge trigger*** corresponds to the ***even numbers*** of the head index value.

14. When the PLC executes the instruction and the state of the initial input is **ON**, the **falling-edge trigger** corresponds to the **odd numbers** of the head index value, and the ***rising-edge trigger*** corresponds to the ***even numbers*** of the head index value.

15. When the PLC executes the instruction, it does not clear the values in the accumulated area and the index areas. If the data is in a latched area and needs to be enabled again, use the ZRST instruction to clear the values in the head and tail indexes.

Example

Refer to the example in the YOUT instruction (API 0710) for more information.

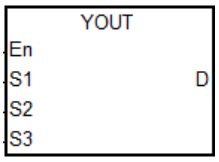
API	Instruction code			Operand	Function
0710		YOUT		**S_1 · S_2 · S_3 · D**	Comparing the outputs of multiple work stations

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								○								
S_3								○								
D		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2		●			●	●							
S_3			●				●						
D	●												

Pulse instruction	16-Bit instruction	32-Bit instruction
-	ES3	-

Symbol



S_1 : High-speed counter number

S_2 : Setting for the number for work stations and objects

S_3 : First corresponding device for the comparison result in the stack area

D : First corresponding device for the output work station

Explanation

1. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

2. **S_1** is for the setting of the high-speed counter. Use the same settings for the high-speed counter as for the high-speed counter for the XCMP instruction.

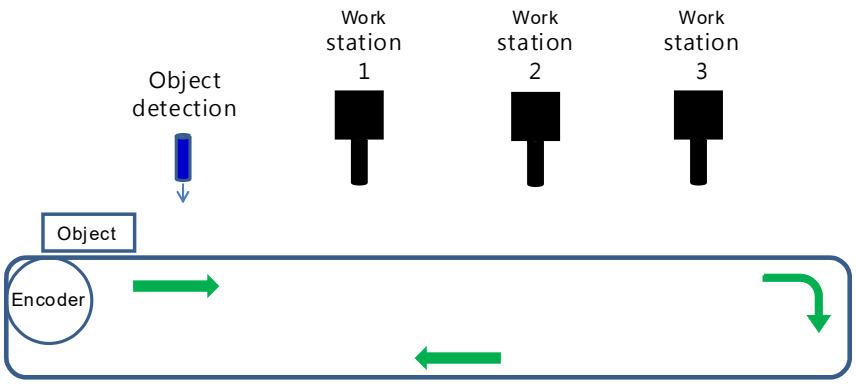
3. **S_2** occupies two consecutive 16-bit devices. **S_2**+0 is n (the work station number) and **S_2**+1 is m (the maximum number of objects). The range for n and m is between 1–64. When the value is out of range, the value used is the maximum (32) or the minimum (1). The settings for the operands should be the same as for the XCMP instruction.

4. **S_3** is first corresponding device for the comparison result in the stack area. **S_3** occupies 2xn consecutive 16-bit devices and 2xmxn consecutive 32-bit devices (or 4xmxn consecutive 16-bit devices). For information on the functions of each device and the corresponding number for **D**, refer to the XCMP instruction (API 0709). It is suggested that you use the same variable as you use for the XCMP instruction.

5. There is no limit on the number of times you can execute the instruction but only one execution can be done at a time.

6. It is suggested that you use with the XCMP instruction, and use the same first corresponding device for the comparison result in the stack area (**S_3**).
7. **D** is only for the outputs of Y and M devices; Y and M should be the BOOL data type. It occupies a consecutive number of work stations Xn. When used as the output point of Y or the M device, the instruction refreshes the output states.
8. The odd numbered head index values (for example 1, 3, 5,…) are the compared counter results for the object when entering (Compared stack areas for the entering -work-station ones). The even numbered head index values (for example 2, 4, 6,…) are the compared counter result of the object when leaving (Compared stack areas for the entering -work-station ones).
9. When the compared counter result for entering and leaving in the stack area are 0, the actions in this area are not executed and the state of the corresponding output work station is OFF. Add 2 to the value of the tail index and the added value in the tail index should not exceed the value of the head index.
10. When the YOUT instruction is executed, each work station checks the compared value for entering and leaving in the tail index. When the counter value is larger or the same as the compared value for entering, the corresponding output point is ON and adds 1 to the value of the tail index. When the counter value is larger or the same as the compared value for leaving, the corresponding output is OFF and adds 1 to the value of the tail index; but the value of the tail index (after adding 1) does not exceed the value of the head index.

Example: three work stations and up to four objects



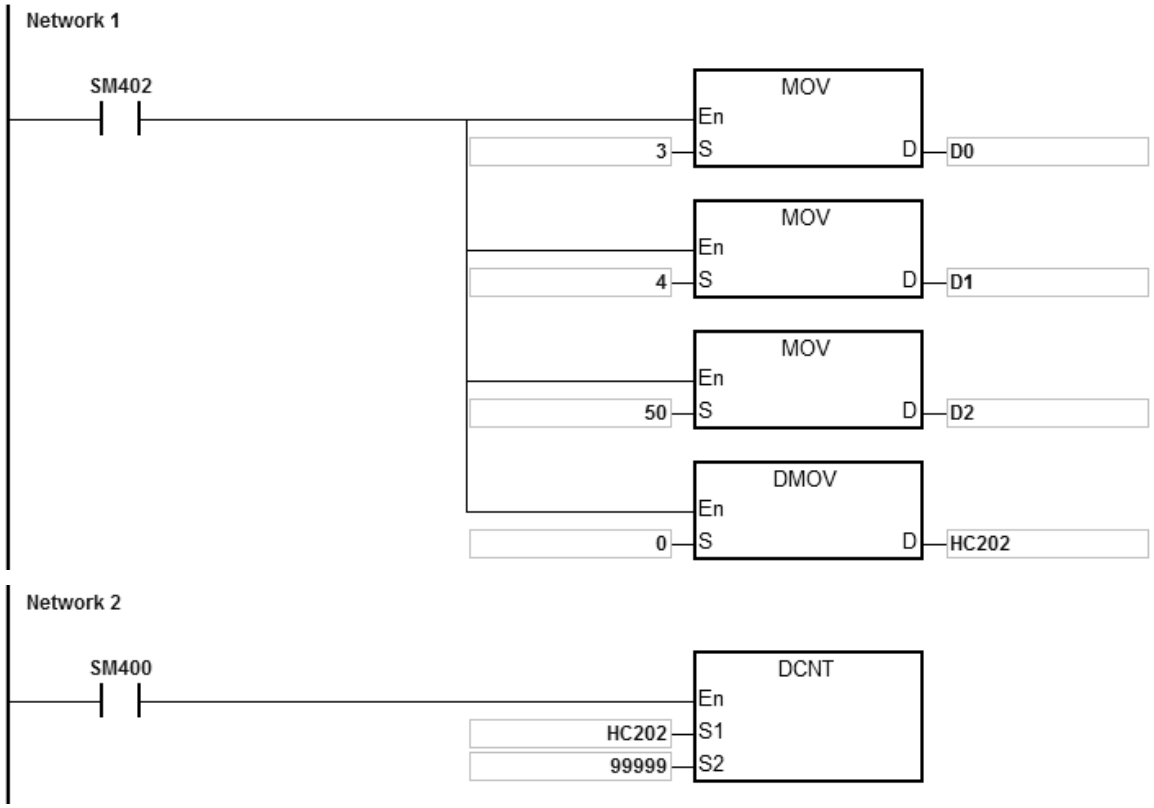
Step 1: use the input point X4 as the object detection interrupt, HC202 as the high-speed counter for the encoder and output point Y0 as the first output point for the work station.

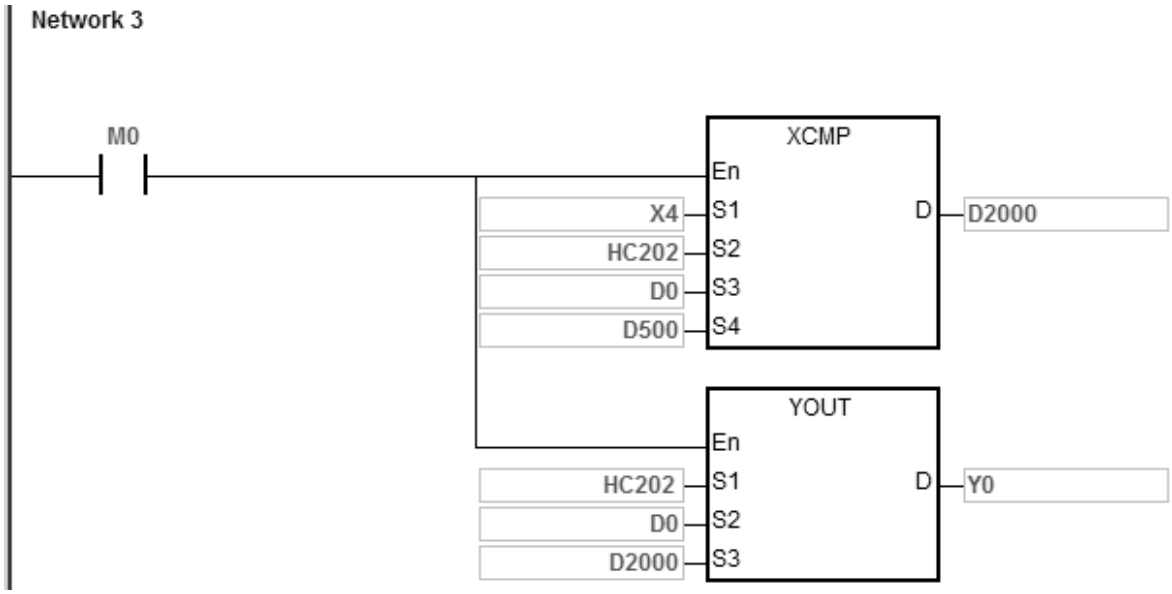
Step 2: edit the register to set up the reference values, and the observational error when entering and leaving.

Device D	D500	D502	D504
Reference value for comparison (32-bit)	K2000	K3000	K4000
Device D	D506	D508	D510
Observational error when entering (32-bit)	K100	K120	K130

Device D	D512	D514	D516
Observational error when leaving (32-bit)	K50	K-20	K20
Device D	D2000	D2001	D2002
Value of the head index (16-bit)	K0	K0	K0
Device D	D2003	D2004	D2005
Value of the tail index (16-bit)	K0	K0	K0

Step 3: set up the initial values and write the programs.





Set up three work stations for D0, 4 objects for D1 and 50 filters for D2. After the contact M0 is activated, the system sets the object detection, the compared values, the compared counter result of the object entering and leaving, and the output controls for each work station. For example, the system detects two objects have entered and then four triggers to read the compared counter results: 3000, 3500, 4500, and 5000 in HC202 (HC202=K5060). The following table shows the compared value and the head/tail index in the stack area.

Device D	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K1	K1	K1
Device D number	D2006	D2008	D2010
Compared stack area 1 for the entering -work-station ones (32-bit)	K5100	K6120	K7130
Device D number	D2012	D2014	D2016
Compared stack area 1 for the leaving-work-station ones (32-bit)	K5550	K6480	K7520
Device D number	D2018	D2020	D2022
Compared stack area 1 for the entering -work-station ones (32-bit)	K6600	K7620	K8630
Device D number	D2024	D2026	D2028
Compared stack area 2 for the leaving-work-station ones (32-bit)	K7050	K7980	K9020

Device D number	D2030	D2032	D2034
Compared stack area 3 for the entering -work-station ones (32-bit)	K0	K0	K0
Device D number	D2036	D2038	D2040
Compared stack area 3 for the leaving-work-station ones (32-bit)	K0	K0	K0

The following table shows the state of the output point Y when the high-speed counter HC202 reaches 5200.

Output point Y number	Y0	Y1	Y2
16-bit value	ON	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K2	K1	K1

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 6200.

Output point Y number	Y0	Y1	Y2
16-bit value	OFF	ON	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K3	K2	K1

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 6800.

Output point Y number	Y0	Y1	Y2
16-bit value	ON	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K3	K1

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 7300.

Output point Y number	Y0	Y1	Y2
16-bit value	OFF	OFF	ON
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K3	K2

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 7700.

Output point Y number	Y0	Y1	Y2
16-bit value	OFF	ON	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K3

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 8000.

Output point Y number	Y0	Y1	Y2
Output state	OFF	OFF	OFF
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K3

The following table shows the state of the output point Y when the high-speed counter HC202 reaching 8700.

Output point Y number	Y0	Y1	Y2
Output state	OFF	OFF	ON
Device D number	D2000	D2001	D2002
Value of the head index (16-bit)	K4	K4	K4
Device D number	D2003	D2004	D2005
Value of the tail index (16-bit)	K4	K4	K4

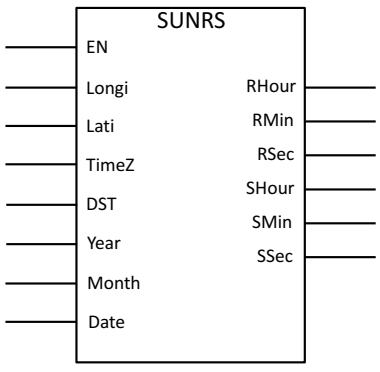
API	Instruction code			Operand	Function
0711	D	SUNRS	P	**Longi ~ SSec**	Sunrise and sunset times

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Longi								●								○
Lati								●								○
TimeZ								●					○			
DST	●	●	●	●												
Year								●					○			
Month								●					○			
Date								●					○			
RHour								●								
RMin								●								
RSec								●								
SHour								●								
SMin								●								
SSec								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Longi									●				
Lati									●				
TimeZ		●				●							
DST	●												
Year		●			●	●							
Month		●			●	●							
Date		●			●	●							
RHour		●			●	●							
RMin		●			●	●							
RSec		●			●	●							
SHour		●			●	●							
SMin		●			●	●							
SSec		●			●	●							

Pulse instruction	16-Bit instruction	32-Bit instruction
ES3		ES3

Symbol



Longi : Longitude (REAL type)

Lati : Latitude (REAL type)

TimeZ : Time zone(integer) (-12 ~ +14)

DST : Daylight saving time

Year Year

Month Month

Date	Date
RHour	The hour to sunrise on the set date (24 hour time format)
RMin	The minute to sunrise on the set date
RSec	The second to sunrise on the set date
SHour	The hour to sunset on the set date (24 hour time format)
SMin	The minute to sunset on the set date
SSec	The second to sunset on the set date

Explanation

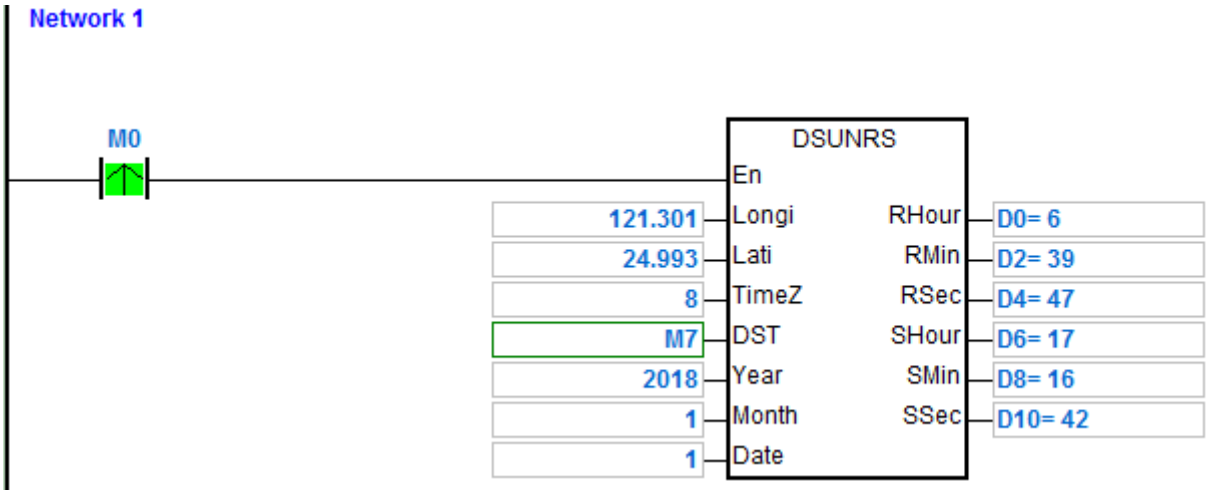
1. The sunrise and sunset times may not be as accurate as the local weather report publishes because the values that you have entered may be incorrect or the altitude of where the device is installed may interfere with the accuracy. When the result is not as accurate, you can adjust the values manually. After self-evaluation, the error range of this instruction is less than 5 minutes.
2. Enter values for the local longitude and latitude in numbers. For example, the longitude and latitude of Taoyuan, Taiwan is 121.30098 and 24.99363. Latitudes north of the Equator are denoted by a positive sign. Latitudes south of the Equator are given negative values.
3. Enter values for the local time zone, ranging from -12 to +14. The time zone cannot be calculated through the set longitude and latitude; if the setting is out of range or the value is incorrect, no error message will be shown.
4. When the daylight saving time is enabled (ON), the instruction checks if the daylight saving time on the PLC is enabled. When the daylight saving time is enabled on the PLC, DST time will be added on the sunrise and sunset times.
5. Enter values for the local date, month, and year in decimal format. Make sure you have entered correct values. The instruction does not check if the values are correctly entered.
6. After calculation, the instruction output the hour, minute and second to sunrise and sunset in integer, in decimal format and 24 hour time format.

Example

Calculate the time to sunrise and sunset in Taoyuan, Taiwan on January 1st, 2018. The official time zone in Taiwan is defined by an UTC offset of +08:00. And daily saving time is NOT implemented in Taiwan.

After calculation, the time to sunrise is at 06:39:47 and the time to sunset is at 17:16:42. See the example program below.

Checked with the official weather website, the actual sunrise occurred at 06:39:44 and the actual sunset occurred at 17:16:45. The difference between the PLC calculation and the actual occurrence is ±3 seconds.


Network 1
M0
DSUNRS
En
121.301
Longi
RHour
D0= 6
24.993
Lati
RMin
D2= 39
8
TimeZ
RSec
D4= 47
M7
DST
SHour
D6= 17
2018
Year
SMin
D8= 16
1
Month
SSec
D10= 42
1
Date

6.9 Logic Instructions

6.9.1 List of Logic Instructions

The following table lists the Logic instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0800	WAND	DAND	✓	Logical AND operation
0801	MAND	–	✓	Matrix AND operation
0802	WOR	DOR	✓	Logical OR operation
0803	MOR	–	✓	Matrix OR operation
0804	WXOR	DXOR	✓	Logical exclusive OR operation
0805	MXOR	–	✓	Matrix exclusive OR operation
0808	WINV	DINV	✓	Logical reversed INV operation
0809	LD&	DLD&	–	S_1&S_2
0810	LD|	DLD|	–	S_1|S_2
0811	LD^	DLD^	–	S_1^S_2
0812	AND&	DAND&	–	S_1&S_2
0813	AND|	DAND|	–	S_1|S_2
0814	AND^	DAND^	–	S_1^S_2
0815	OR&	DOR&	–	S_1&S_2
0816	OR|	DOR|	–	S_1|S_2
0817	OR^	DOR^	–	S_1^S_2

6.9.2 Explanation of Logic Instructions

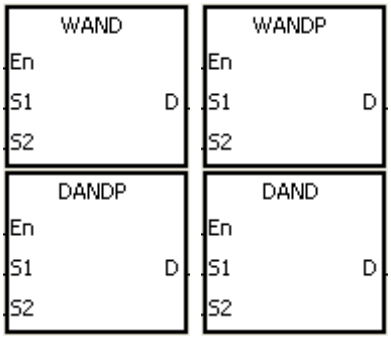
API	Instruction code			Operand	Function
0800	D	WAND	P	**S_1, S_2, D**	Logical AND operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source 1

S_2 : Data source 2

D : Operation result

Explanation

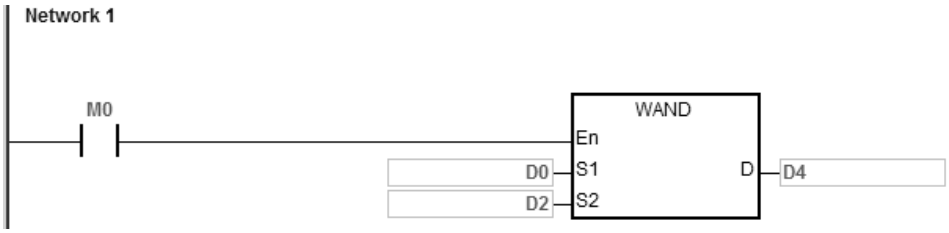
1. This instruction applies the logical operator AND to the binary representations in **S_1** and **S_2** It performs the logical AND operation on each pair of corresponding bits and stores the result in **D**.

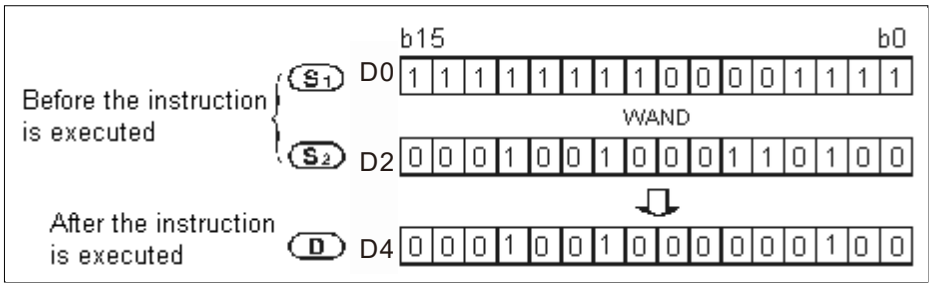
2. Only the DAND instruction can use the 32-bit counter.

3. The result in each position is 1 if the first bit is 1 and the second bit is 1; otherwise, the result is 0.

Example 1

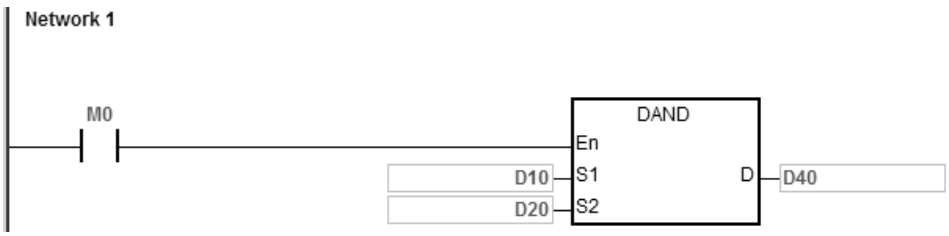
When M0 is ON, the instruction performs the logical operation AND on each pair of corresponding bits in the 16-bit device D0 and the 16-bit device D2. It stores the result in D4.

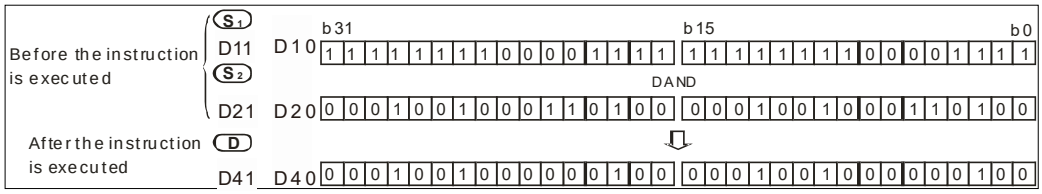




Example 2

When M0 is ON, the instruction performs the logical operation AND on each pair of corresponding bits in the 32-bit device (D11, D10) and the 32-bit device (D21, D20). It stores the result in (D41, D40).





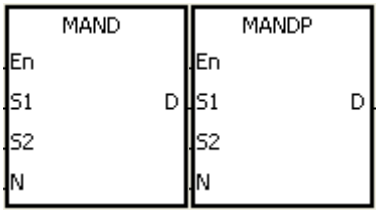
API	Instruction code			Operand	Function
0801		MAND	P	**S_1, S_2, D, n**	Matrix AND operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
D					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

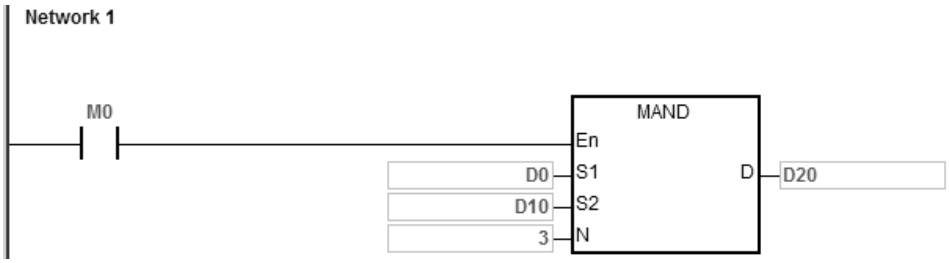
n : Length of the array

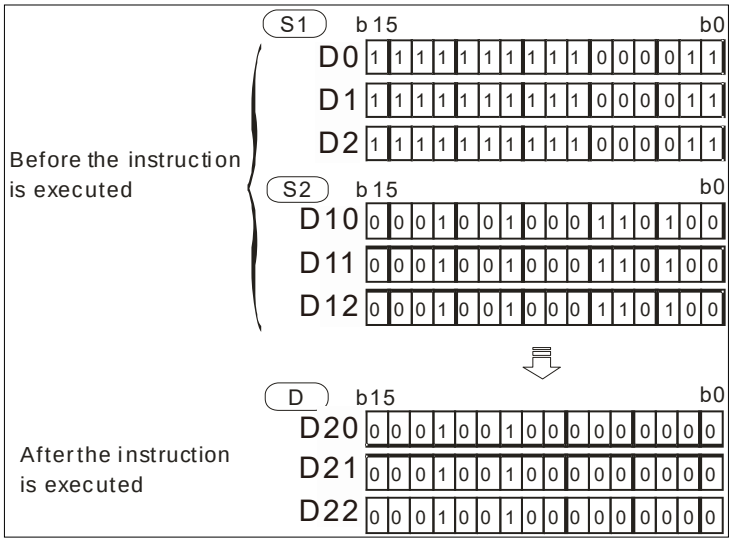
Explanation

1. This instruction applies the logical operator MAND to the **n** rows of binary representations in **S_1** and the **n** rows of binary representations in **S_2**. It performs the matrix operation AND on each pair of corresponding bits, and stores the operation result in **D**.
2. The result in each position is 1 if the first bit is 1 and the second bit is 1; otherwise, the result is 0.
3. The operand **n** must be between 1–256.

Example

When M0 is ON, this instruction performs the matrix operation AND on each pair of corresponding bits on the data in the 16-bit devices D0–D2 and the data in 16-bit devices D10–D12. It stores the result in the 16-bit devices D20–D22.





Additional remarks

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Explanation of matrix instructions:

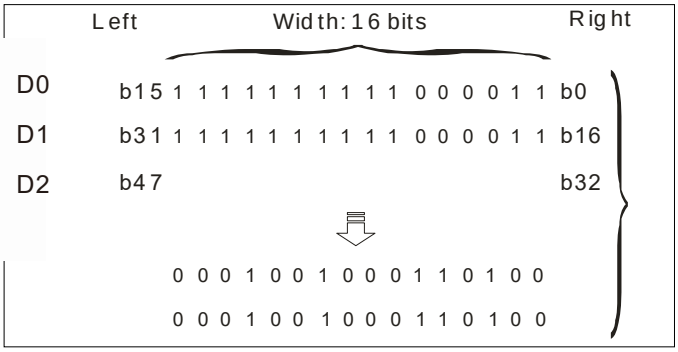
   - A matrix is composed of more than one 16-bit register. The number of registers in a matrix is the length of the array **n**. There are 16×**n** bits in a matrix, and the matrix operation is performed on one bit at a time.

   - The matrix instruction takes the 16×**n** bits in a matrix as a string of bits, rather than as values. The matrix operation is performed on each bit.

   - Matrix instructions mainly process the one-to-many or many-to-many status, such as moving, copying,

comparing, and searching by bit.

- You must specify a 16-bit register for the matrix instruction. The 16-bit register specifies a certain bit among the 16**n** bits in the matrix for the operation, and the 16-bit register is called the pointer. The value in the register is between 0–16**n**-1, and corresponds to the bit between b0–b16**n**-1.
- Shifting or rotating of the specified data can be involved in the matrix operation. Note that the bit number decreases from the left to the right, as illustrated below.



- The width of the matrix is 16 bits.
- Pr represents the pointer. When the value in Pr is 15, it specifies b15.

Example: The following matrix is composed of the three 16-bit devices D30, D31, and D32. The data in D30 is 16#AAAA, the data in D31 is 16#5555, and the data in D32 is 16#AAFF.

b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	D30
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	D31
1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	D32

Example: The following matrix is composed of the three 16-bit devices D40, D41, and D42. The data in D40 is 16#37, the data in D41 is 16#68, and the data in D42 is 16#45.

b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	D40
0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	D41
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	D42

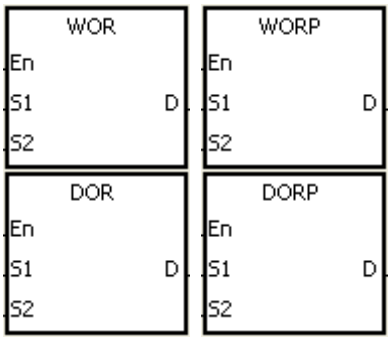
API	Instruction code			Operand	Function
0802	D	WOR	P	**S_1, S_2, D**	Logical OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source 1

S_2 : Data source 2

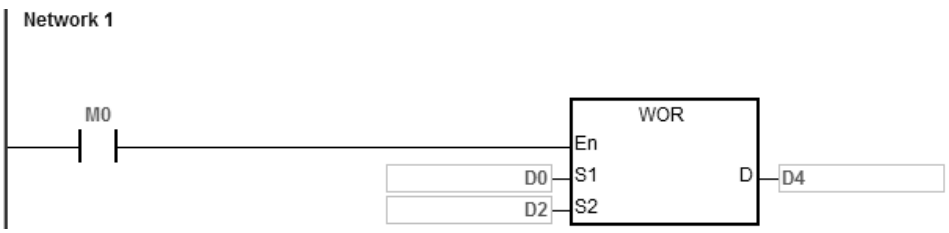
D : Operation result

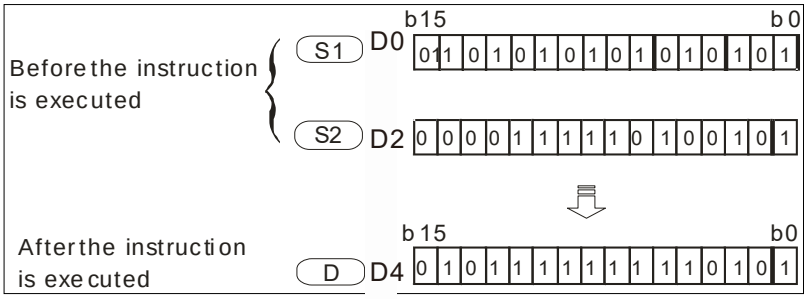
Explanation

1. This instruction applies the logical operator OR to the binary representations in **S_1** and **S_2**. It performs the logical inclusive operation OR on each pair of corresponding bits, and stores the operation result in **D**.
2. Only the DOR instruction can use the 32-bit counter but not the device E.
3. The result in each position is 1 if the first bit is 1, the second bit is 1, or both bits are 1; otherwise, the result is 0.

Example 1

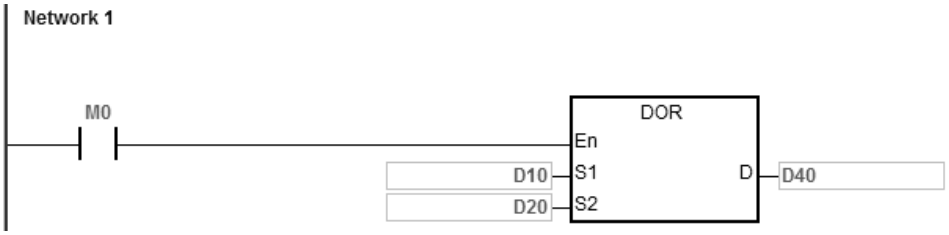
When M0 is ON, This instruction performs the logical inclusive operation OR on each pair of corresponding bits in the 16-bit device D0 and the 16-bit device D2. It stores the operation result in D4.

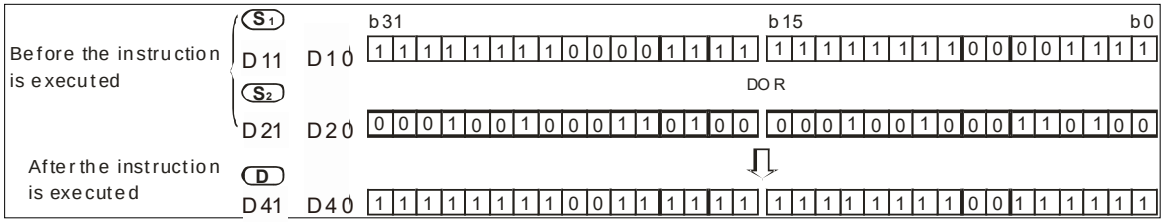




Example 2

When M0 is ON, the instruction performs the logical inclusive operation OR on each pair of corresponding bits in the 32-bit device (D11, D10) and the 32-bit device (D21, D20). It stores the operation result in (D41, D40).





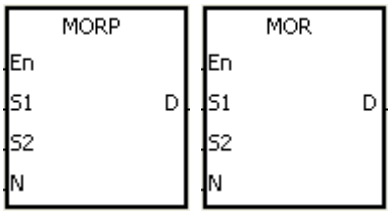
API	Instruction code			Operand	Function
0803		MOR	P	**S_1, S_2, D, n**	Matrix OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
D					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

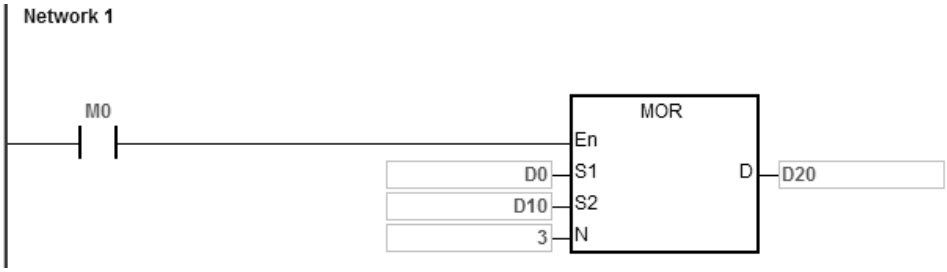
n : Length of the array

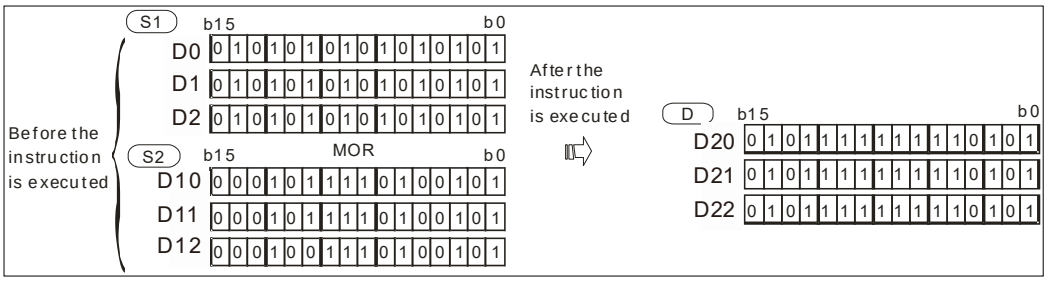
Explanation

1. This instruction applies the logical operator OR to the **n** rows of binary representations in **S_1** and the **n** rows of binary representations in **S_2**. It performs the matrix operation OR on each pair of corresponding bits and stores the operation result in **D**.
2. The result in each position is 1 if the first bit is 1, the second bit is 1, or both bits are 1; otherwise, the result is 0.
3. The operand **n** must be between 1–256.

Example

When M0 is ON, the instruction performs the matrix operation OR on each pair of corresponding bits in the 16-bit devices D0–D2 and the data in 16-bit devices D10–D12. It stores the operation result in the 16-bit devices D20–D22.





Additional remarks

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

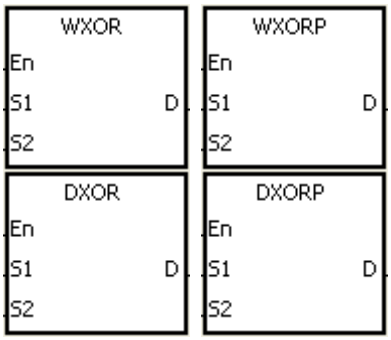
API	Instruction code			Operand	Function
0804	D	WXOR	P	**S_1, S_2, D**	Logical exclusive OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Data source 1

S_2 : Data source 2

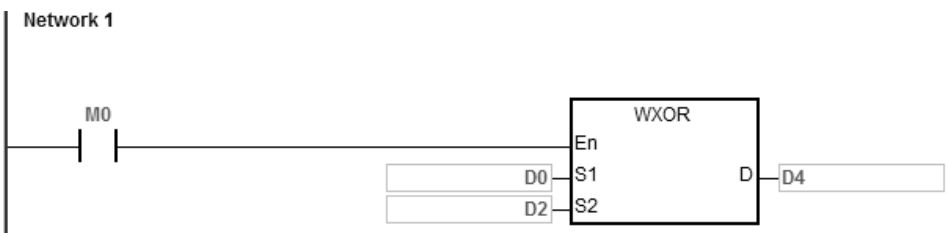
D : Operation result

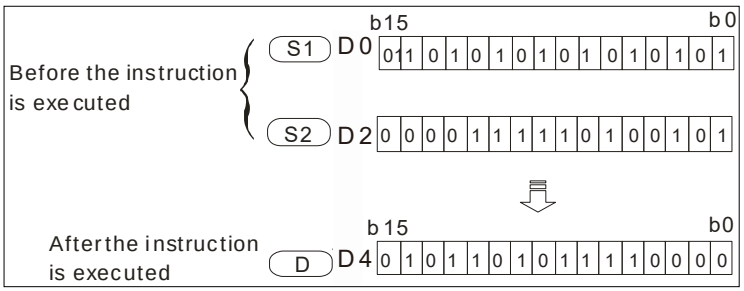
Explanation

1. This instruction applies the logical operator XOR to the binary representations in **S_1** and **S_2**. It performs the logical exclusive operation OR on each pair of corresponding bits, and stores the operation result in **D**.
2. Only the DXOR instruction can use the 32-bit counter, but not the device E.
3. The result in each position is 1 if the two bits are different, and 0 if they are the same.

Example 1

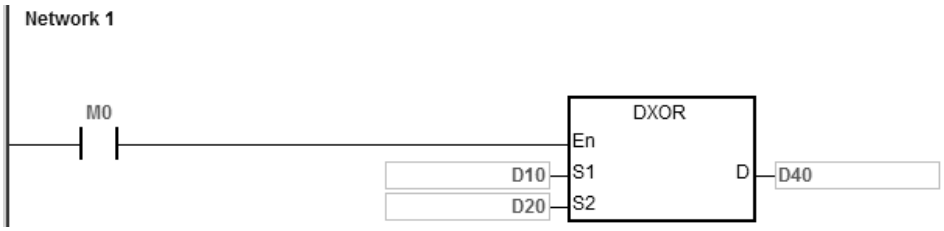
When M0 is ON, the instruction performs the exclusive operation OR on each pair of corresponding bits in the 16-bit device D0 and the 16-bit device D2. It stores the operation result in D4.

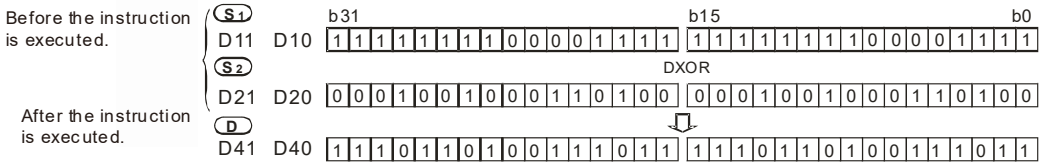




Example 2

When M0 is ON, the instruction performs the logical exclusive operation OR on each pair of corresponding bits in the 32-bit device (D11, D10) and the 32-bit device (D21, D20). It stores the operation result in (D41, D40).





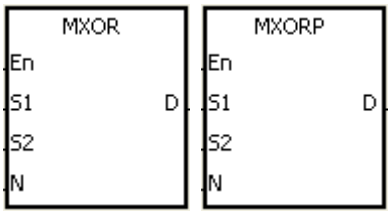
API	Instruction code			Operand	Function
0805		MXOR	P	**S_1, S_2, D, n**	Matrix exclusive OR operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
D					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Matrix source 1

S_2 : Matrix source 2

D : Operation result

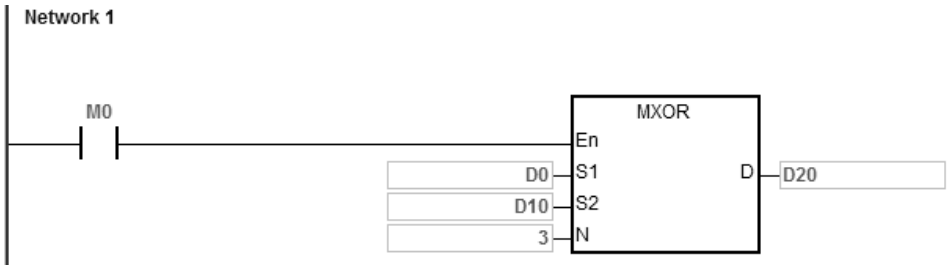
n : Length of the array

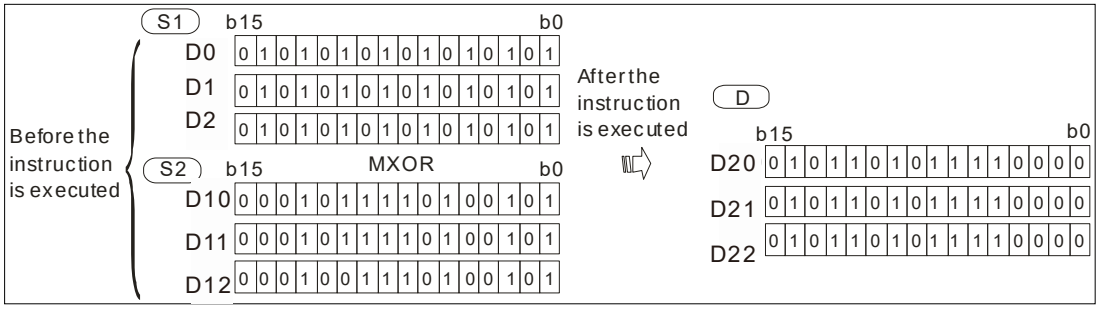
Explanation

1. This instruction applies the logical operator XOR to the **n** rows of binary representations in **S_1** and the **n** rows of binary representations in of **S_2**. It performs the matrix exclusive operation OR on each pair of corresponding bits and stores the operation result in **D**.
2. The result in each position is 1 if the two bits are different, and 0 if they are the same.
3. The operand **n** must be between 1–256.

Example

When M0 is ON, the instruction performs the matrix exclusive operation OR on each pair of corresponding bits in the 16-bit devices D0–D2 and the data in 16-bit devices D10–D12. It stores the operation result in the 16-bit devices D20–D22.





Additional remarks

1. If **S_1+n-1**, **S_2+n-1**, or **D+n-1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

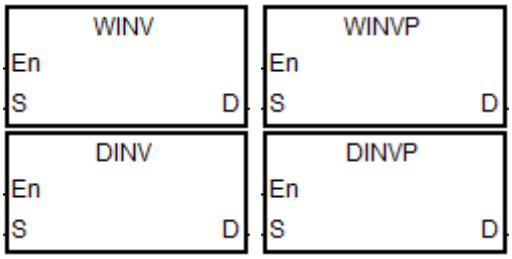
API	Instruction code			Operand	Function
0808	D	WINV	P	**S, D**	Logical reversed INV operation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Data source

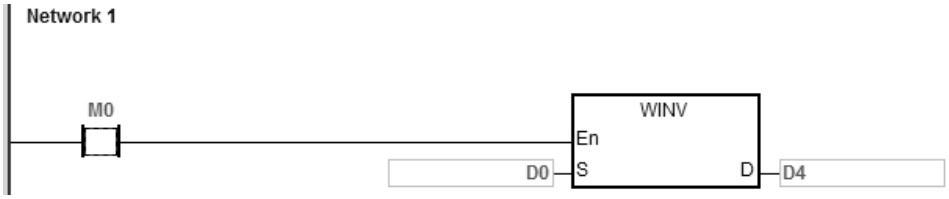
D : Operation result

Explanation

1. This instruction applies the INV instruction to the data in **S** , and stores the operation result in **D**.
2. Only the DINV instruction can use the 32-bit counter but not the device E.
3. This instruction performs reverse processing on the data in S. If the state of S is 0 before executing the INV instruction, the state changes to 1 as a result of the INV instruction.

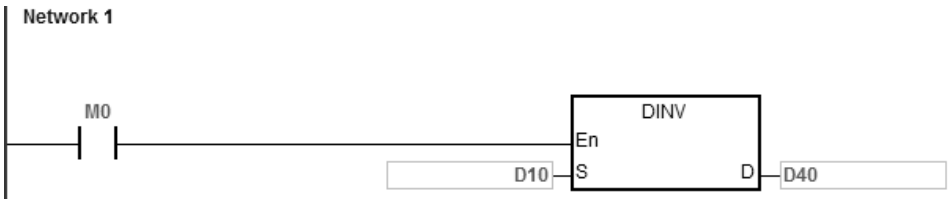
Example 1

When M0 is ON, the instruction performs the INV operation on the corresponding bits in the 16-bit device D0. It stores the operation result in the 16-bit device D4.



Example 2

When M0 is ON, the instruction performs the INV operation on each pair of corresponding bits in the 32-bit devices D11–D10. It stores the operation result in the 32-bit device D41–D40.



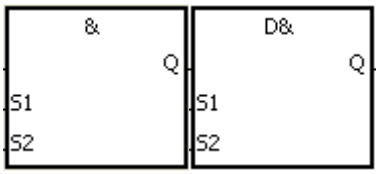
API	Instruction code			Operand	Function
0809- 0811	D	LD #		S_1, S_2	Contact type of logical operation LD #

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking LD& and DLD& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation

1. The instruction is used to compare the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.
2. Only the instruction DLD # can use the 32-bit counter but not the device E.
3. The instruction LD # can be connected to the mother line directly.

API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0809	LD&	**D**LD&	$S_1 \& S_2 \neq 0$	$S_1 \& S_2 = 0$
0810	LD|	**D**LD|	$S_1 \mid S_2 \neq 0$	$S_1 \mid S_2 = 0$
0811	LD^	**D**LD^	$S_1 \wedge S_2 \neq 0$	$S_1 \wedge S_2 = 0$

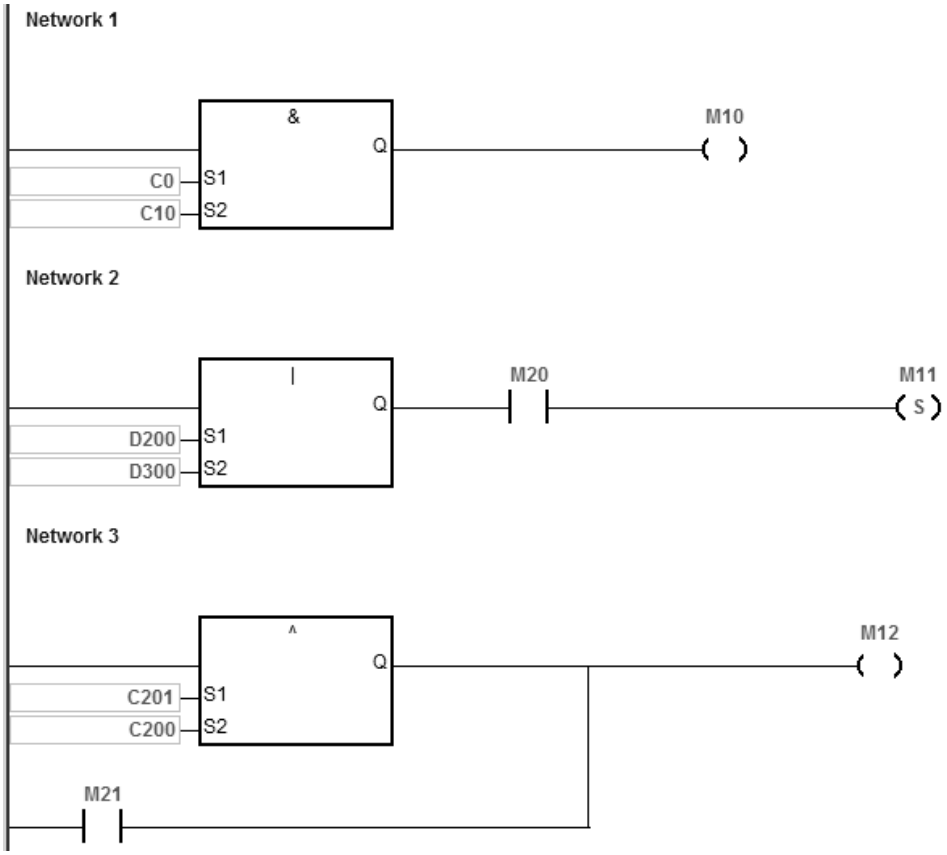
4. &: Logical AND operation
5. |: Logical OR operation
6. ^: Logical exclusive OR operation

Example

1. The logical operator AND takes the data in C0 and C1, and performs the logical AND operation on each pair of corresponding bits. When the operation result is not 0, M10 is ON.

2. The instruction performs the logical operation OR on each pair of corresponding bits in D200 and D300, when the operation result is not 0 and M20 is ON, M11 is ON and is retentive.

3. The instruction performs logical exclusive operation XOR on each pair of corresponding bits in C201 and C200, when the operation result is not 0, or when M21 is ON, M12 is ON.



Additional remarks

If S_1 or S_2 is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

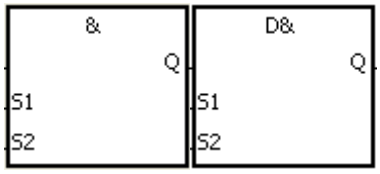
API	Instruction code			Operand	Function
0812- 0814	D	AND #		S_1, S_2	Contact type of logical operation AND #

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking AND& and DAND& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation

1. This instruction compares the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.
2. Only the DAND # instruction can use the 32-bit counter, but not the device E.
3. Connect the AND # instruction and the contact in series.

API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0812	AND&	**D**AND&	$S_1 \& S_2 \neq 0$	$S_1 \& S_2 = 0$
0813	AND|	**D**AND|	$S_1 \mid S \neq 0$	$S_1 \mid S_2 = 0$
0814	AND^	**D**AND^	$S_1 \wedge S_2 \neq 0$	$S_1 \wedge S = 0$

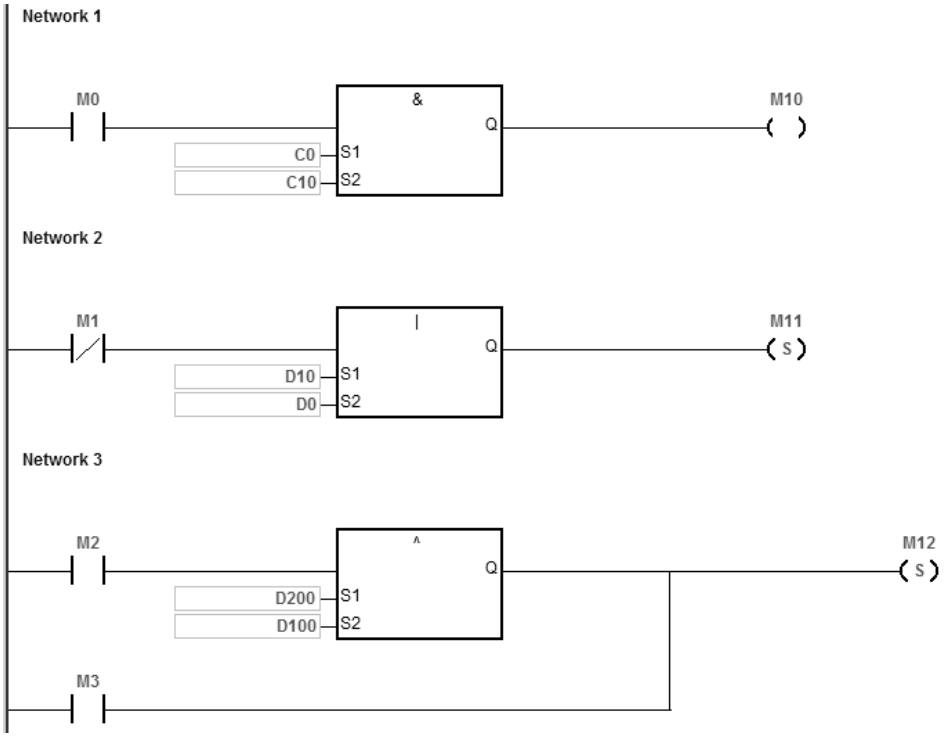
4. &: Logical AND operation
5. |: Logical OR operation
6. ^: Logical exclusive OR operation

Example

1. When M0 is ON, the instruction performs the logical operation AND on each pair of corresponding bits in C0 and C10. When the operation result is not 0, M10 is ON.

2. When M1 is OFF, the instruction performs the logical operation OR on each pair of corresponding bits in D10 and D0. When the operation result is not 0, M11 is ON.

3. When M2 is ON, the instruction performs the logical exclusive operation OR on each pair of corresponding bits in the 32-bit register (D200, D201) and the data in the 32-bit register (D100, D101). When the operation result is not 0, or when M3 is ON, M12 is ON.



Additional remarks

If the value in S_1 or S_2 is not valid, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

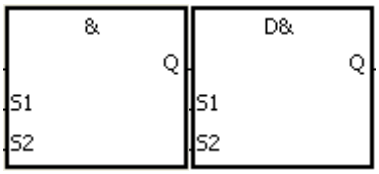
API	Instruction code			Operand	Function
0815- 0817	D	OR#		S_1, S_2	Contact type of logical operation OR#

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●				●	●	
S_2		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



Taking OR& and DOR& for example

S_1 : Data source 1

S_2 : Data source 2

Explanation

1. This instruction compares the data in S_1 with that in S_2. When the comparison result is not 0, the condition of the instruction is met. When the comparison result is 0, the condition of the instruction is not met.
2. Only the DOR# instruction can use the 32-bit counter.
3. Connect the OR# instruction and the contact in parallel.

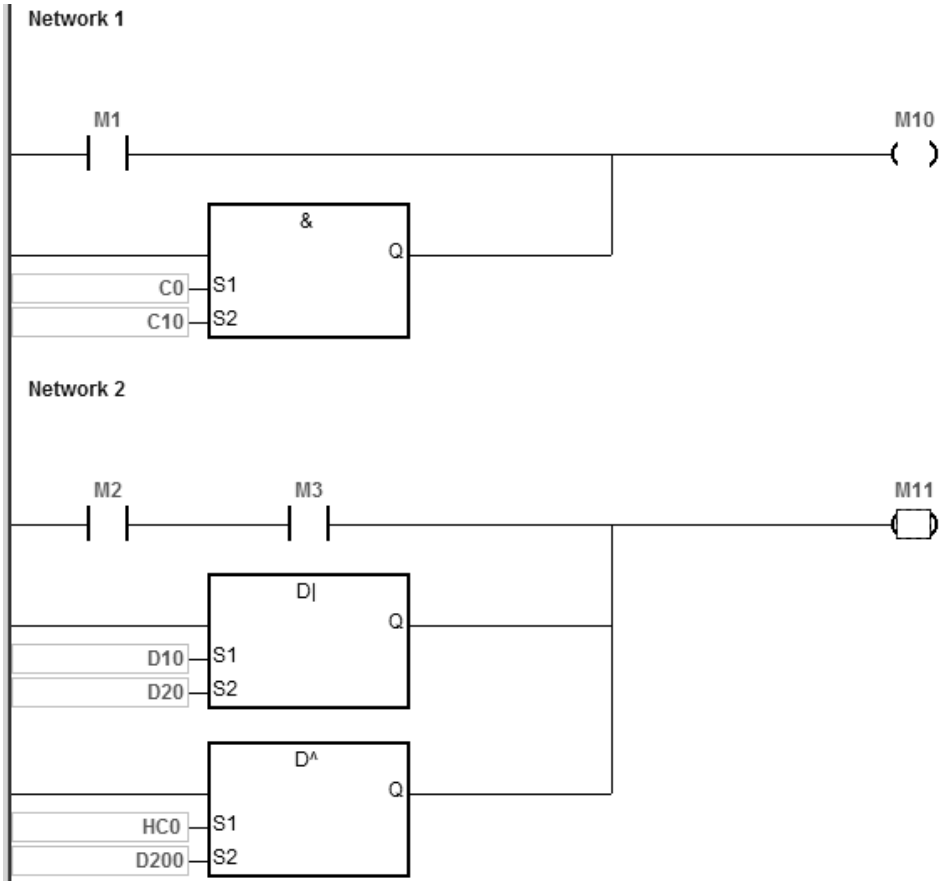
API No.	16-bit instruction	32-bit instruction	Comparison operation result	
			ON	OFF
0815	OR&	**D**OR&	S_1&$S_2 \neq 0$	S_1&$S_2 = 0$
0816	OR|	**D**OR|	S_1|$S_2 \neq 0$	S_1|$S_2 = 0$
0817	OR^	**D**OR^	S_1^$S_2 \neq 0$	S_1^$S_2 = 0$

4. &: Logical AND operation
5. |: Logical OR operation
6. ^: Logical exclusive OR operation

Example

1. When M1 is ON, M10 is ON. The instruction performs the logical operation AND on each pair of corresponding bits in C0 and C10. When the operation result is not 0, M10 is ON.

2. When M2 and M3 are ON, M11 is ON. The instruction performs the logical operation OR on each pair of corresponding bits in the 32-bit register (D10, D11) and the 32-bit register (D20, D21). When the operation result is not 0, M11 is ON. The instruction performs the logical exclusive operation OR on each pair of corresponding bits in the 32-bit counter HC0. and the 32-bit register (D200, D201). When the operation result is not 0, M11 is ON.



Additional remarks

If the value in S_1 or S_2 is not valid, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

6.10 Rotation Instructions

6.10.1 List of Rotation Instructions

The following table lists the Rotation instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
0900	ROR	DROR	✓	Rotating bits in a group to the right
0901	RCR	DRCR	✓	Rotating bits in a group to the right with the carry flag
0902	ROL	DROL	✓	Rotating bits in a group to the left
0903	RCL	DRCL	✓	Rotating bits in a group to the left with the carry flag
0904	MBR	–	✓	Rotating bits to the right or left in a matrix

6.10.2 Explanation of Rotation Instructions

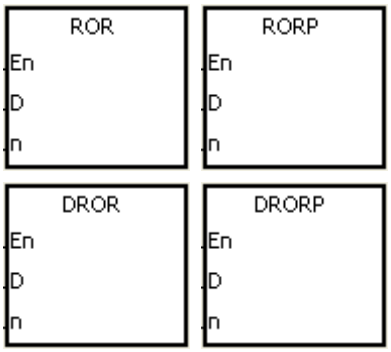
API	Instruction code			Operand	Function
0900	D	ROR	P	**D, n**	Rotating bits in a group to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Device to rotate

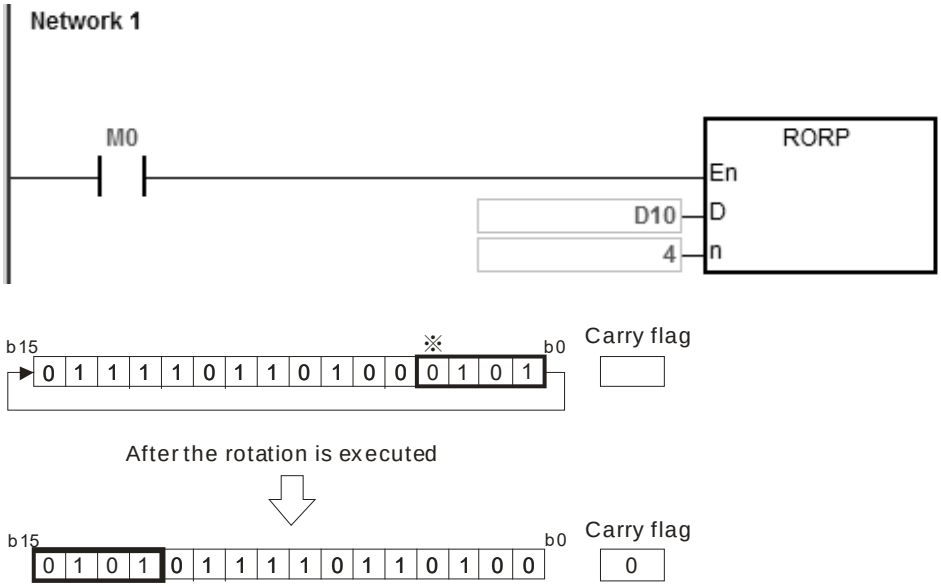
n : Number of bits in a group

Explanation

1. This instruction divides the bits in the device specified by **D** into groups (**n** bits in a group), and then rotates these groups to the right without the carry flag.
2. Only the DROR instruction can use the 32-bit counter, but not the device E.
3. For the 16-bit instruction, the value of **n** used must be between 1–6. For the 32-bit instruction, the value of **n** must be between 1–32. When **n** is less than 0, the instruction is not executed. When **n** exceeds the range, the instruction is executed with **n** at the maximum value (32) of the range.
4. In general, the RORP and DRORP pulse instructions are used.

Example

When M0 switches from OFF to ON, the instruction divides the values of the bits in D10 into groups (four bits in a group), and rotates these groups to the right. The value of the bit marked ※ is transmitted to the carry flag SM602.



Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

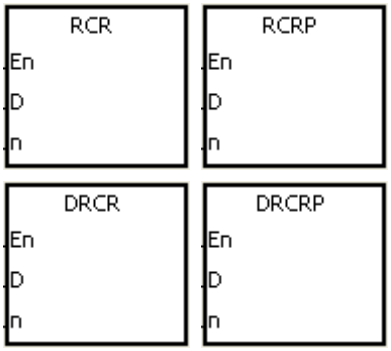
API	Instruction code			Operand	Function
0901	D	RCR	P	**D, n**	Rotating bits in a group to the right with the carry flag

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Device to rotate

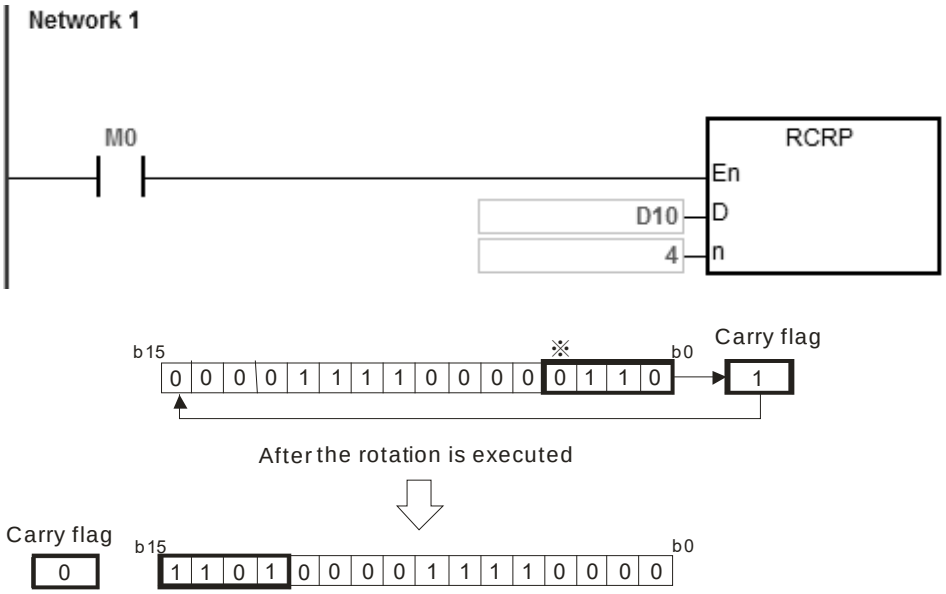
n : Number of bits in a group

Explanation

1. This instruction divides the bits in the device specified by **D** into groups (**n** bits in a group), and then rotates these groups to the right with the carry flag SM602.
2. Only the DRCR instruction can use the 32-bit counter, but not the device E.
3. For 16-bit instructions, the value of **n** used must be between 1–16. For 32-bit instructions, the value of **n** must be between 1–32. When **n** is less than 0, the instruction is not executed. When **n** exceeds the range, the instruction is executed with **n** at the maximum value (32) of the range.
4. In general, the RCRP and DRCRP pulse instructions are used.

Example

When M0 switches from OFF to ON, the instruction divides the values of the bits in D10 into groups (four bits as a group), and then rotates these groups to the right with the carry flag SM602. The value of the bit marked ※ is transmitted to the carry flag SM602.



Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

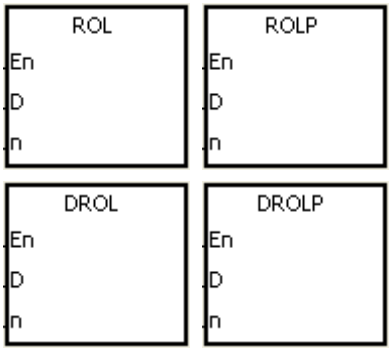
API	Instruction code			Operand	Function
0902	D	ROL	P	**D, n**	Rotating bits in a group to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Device to rotate

n : Number of bits in a group

Explanation

1. This instruction divides the bits in the device specified by **D** into groups (**n** bits in a group), and then rotates these groups to the left.

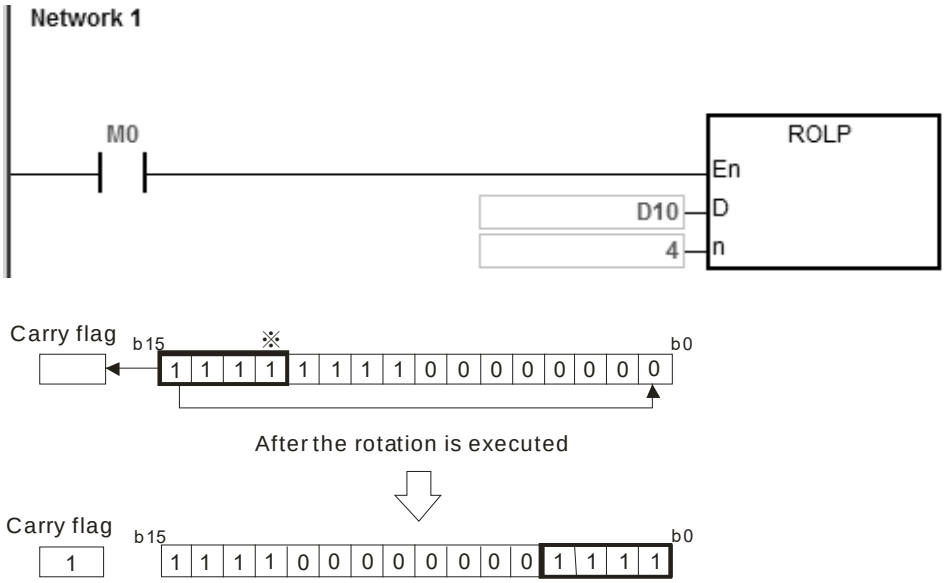
2. Only the DROL instruction can use the 32-bit counter, but not the device E.

3. For 16-bit instructions, the value of **n** must be between 1–16. For 32-bit instructions, the value of **n** must be between 1–32. When **n** is less than 0, the instruction is not executed. When **n** exceeds the range, the instruction is executed with **n** at the maximum value (32) of the range.

4. In general, the ROLP and DROLP pulse instructions are used.

Example

When M0 switches from OFF to ON, the instruction divides the values of the bits in D10 into groups (four bits as a group), and then rotates these groups to the left. The value of the bit marked ※ is transmitted to the carry flag SM602.



Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

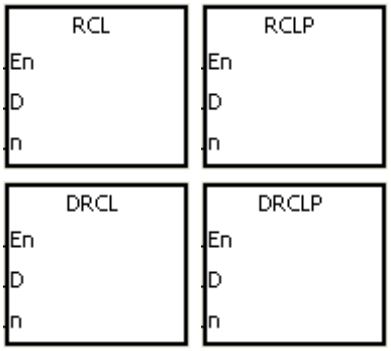
API	Instruction code			Operand	Function
0903	D	RCL	P	**D, n**	Rotating bits in a group to the left with the carry flag

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



D : Device to rotate

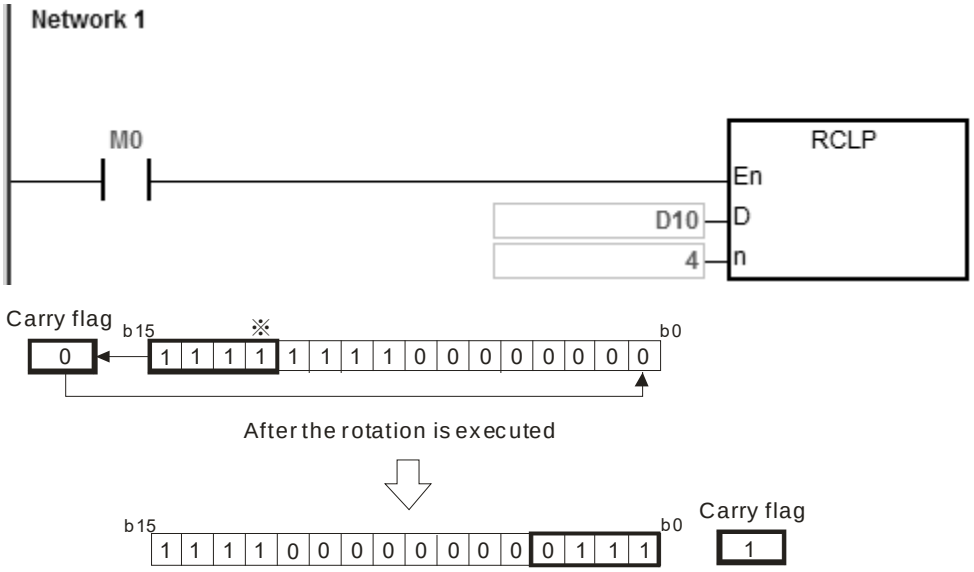
n : Number of bits in a group

Explanation

1. This instruction divides the bits in the device specified by **D** into groups (**n** bits in a group) , and then rotates these groups to the left with the carry flag SM602.
2. Only the DRCL instruction can use the 32-bit counter, but not the device E.
3. For 16-bit instructions, the value in **n** must be between 1–16. For 32-bit instructions, the value of **n** must be between 1–32. When **n** is less than 0, the instruction is not executed. When **n** exceeds the range, the instruction is executed with **n** at the maximum value (32) of the range.
4. In general, the RCLP and DRCLP pulse instructions are used.

Example

When M0 switches from OFF to ON, the instruction divides the values of the bits in D10 into groups (four bits as a group), and then rotates these groups to the left with the carry flag SM602. The value of the bit marked ※ is transmitted to the carry flag SM602.



Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

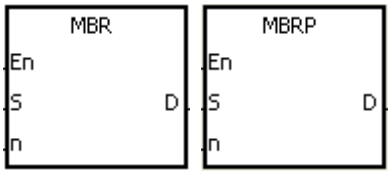
API	Instruction code			Operand	Function
0904		MBR	P	**S, D, n**	Rotating bits in a group to the right or the left in a matrix

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
D					●	●		●								
n					●	●		●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Matrix source

D : Operation result

n : Length of the array

Explanation

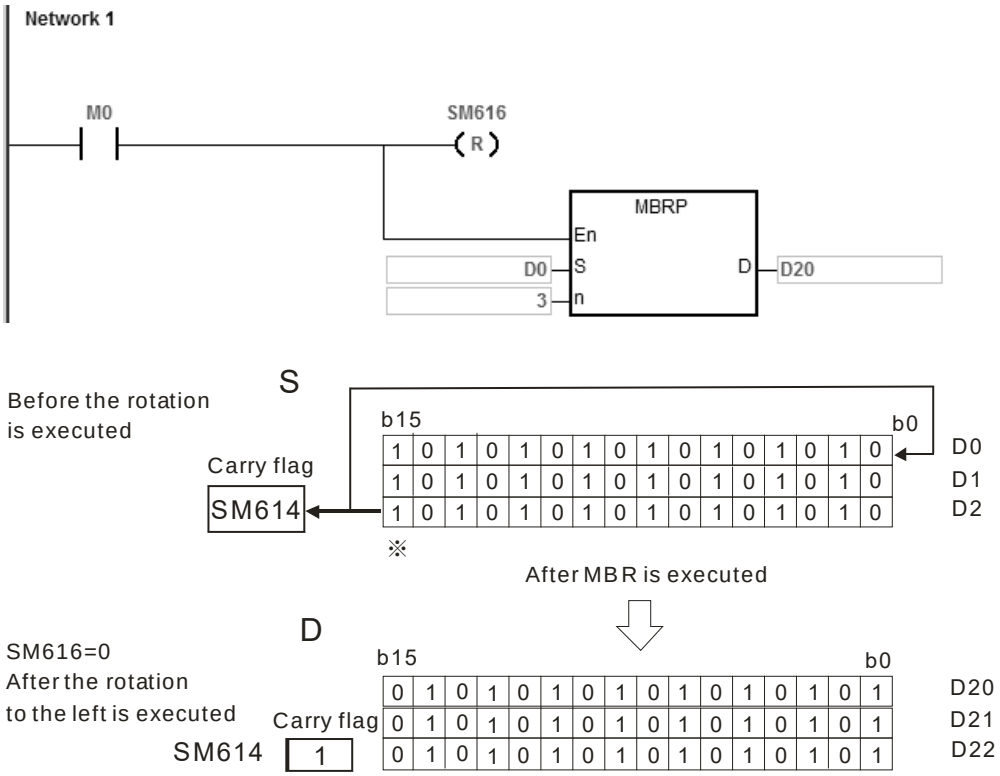
1. This instruction rotates the values of **n** rows of bits in **S** to the right or to the left. When SM616 is OFF, the instruction rotates the values of the bits to the left. When SM616 is ON, the instruction rotates the values of the bits to the right. The instruction fills the vacancy resulting from the rotation with the value of the bit rotated last, and stores the operation result in **D**. The value of the bit rotated last not only fills the vacancy, but also is transmitted to the carry flag SM614.

2. For 16-bit instructions, the value of **n** must be between 1–16. For 32-bit instructions, the value of **n** must be between 1–32. When **n** is less than 0, the instruction is not executed. When **n** exceeds the range, the instruction is executed with **n** at the maximum value (32) of the range.

3. In general, the MBRP pulse instruction is used.

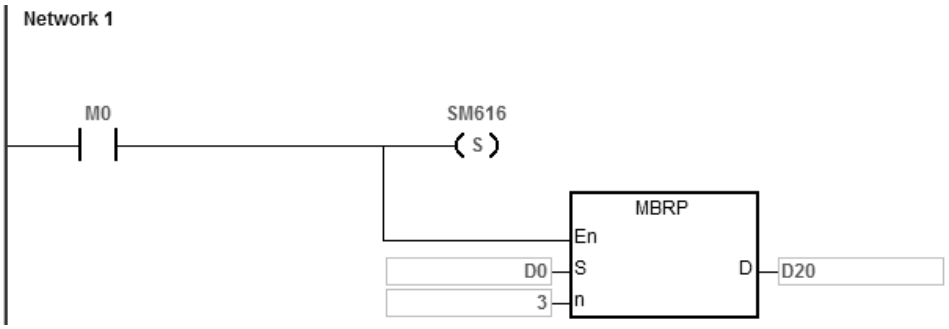
Example 1:

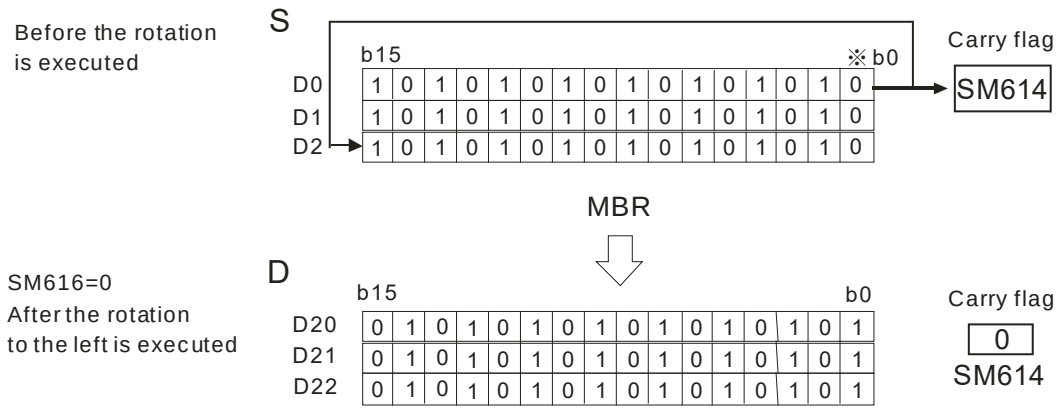
When M0 is ON and SM616 is OFF, the instruction rotates the values of the bits in the 16-bit registers D0–D2 to the left, and stores the operation result in the 16-bit registers D20–D22. The value of the bit marked ※ is transmitted to the carry flag SM614.



Example 2:

When M0 is ON and SM616 is ON, the instruction rotates the values of the bits in the 16-bit registers D0–D2 to the right, and stores the operation result in the 16-bit registers D20–D22. The value of the bit marked ※ is transmitted to the carry flag SM614.





Additional remarks

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. Instruction flags:

   SM614: The carry flag for the matrix rotation/shift/output

   SM616: The direction flag for the matrix rotation/shift

6.11 Timer and Counter Instructions

6.11.1 List of Timer and Counter Instructions

The following table lists the Timer and Counter instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1000	RST	DRST	–	Resetting a contact to OFF or clearing the value in a register.
1001	TMR	–	–	16-bit timer (Unit: 100ms)
1002	TMRH	–	–	16-bit timer (Unit: 1ms)
1003	CNT	–	–	16-bit counter
1004	–	DCNT	–	32-bit counter (including the use of high-speed counters)
1005	–	DHSCS	–	Setting high-speed comparison
1006	–	DHSCR	–	Resetting high-speed comparison
1007	–	DHSZ	–	High-speed input zone comparison
1008	–	DSPD	–	Detecting speed
1009	PWD	–	–	Detecting pulse width
1010	–	DCAP	–	Capturing the high-speed count value in an external input interrupt
1011	TMRM	–	–	16-bit timer (Unit: 10ms)
1012	IETS	–	✓	The start of the instruction execution time measurement
1013	IETE	–	✓	The end of the instruction execution time measurement

6.11.2 Explanation of Timer and Counter Instructions

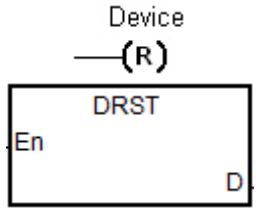
API	Instruction code			Operand	Function
1000	D	RST		**D**	Resetting a contact or clearing a register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		○	○	○	○	○	○	○		○	○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●	●			●	●		●		●	●	

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



D : Device to reset

Explanation

1. This instruction clears the values in a 32-bit HC device or two consecutive 16-bit **D** devices. For other devices, use the RST instruction to clear the values.

2. The following table shows the actions of the RST instruction.

Device	State
Bit	Sets the coil and contact to OFF.
T · C	Set the current timer value and counter value to 0, and sets the coil and contact to OFF.
Word	Clears the 16-bit content value to 0.
DWord · HC · Real	Clears the 32-bit content value, including floating point numbers, to 0.

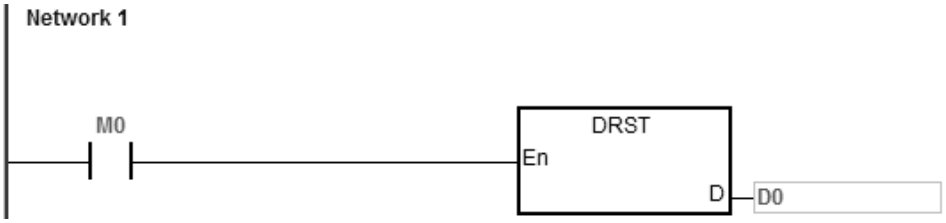
3. If the RST instruction is not executed, the state of the device specified by **D** is unchanged.

4. The instruction supports direct output.

Example

When M0 is ON, the instruction sets M1 to OFF.

The instruction clears the 32-bit D1 and D0 to zero when M0 is ON.



API	Instruction code	Operand	Function
1001	TMR	S_1, S_2	16-bit timer (100ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

S_1 : Timer number

S_2 : Setting value of the timer

Explanation

Refer to the explanation of the TMRH instruction (API 1002) for details.

API	Instruction code	Operand	Function
1002	TMRH	**S_1, S_2**	16-bit timer (1ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

S_1 : Timer number

S_2 : Setting value for the timer

Explanation

1. The TMR instruction uses 100ms as the timing unit in the timer, while, the TMRH instruction uses 1ms as the timing unit in the timer.
2. The value of **S_2** for both the TMR and TMRH instructions is between 0–32767.
3. If you use the same timer repeatedly in the program, including using it in different TMR and TMRH instructions, the timer that completes the measurement first will be the only one that counts.
4. The T timer resets to zero automatically when the conditional contact changes from ON to OFF.
5. When you add the letter S in front of the device T, the timer in the instruction TMR is an accumulative timer. When the conditional contact is OFF, the value of the accumulative timer is not cleared. When the conditional contact is ON, the timer counts from the current value. Use the RST instruction with the ST accumulative timer when you want to clear the value of the timer.
6. If you use the same T timer in the program, it is OFF when one of the conditional contacts is OFF.
7. If you use the same T timer for T and ST in the program, T is OFF when one of the conditional contacts is OFF.
8. When the instruction TMR is executed, the specified timer coil is ON and the timer begins to count. As the value of the timer matches the setting value, the contact is ON.

9. The timers T0–T411 are defined as general timers, and T412–T511 are subroutine timers by default. Use the hardware configuration software HWCONFIG if you need to change the ranges of the two types of timers.

10. The general timers compare the timing values when the TMR instruction is scanned. The system applies the timer to the condition every time the TMR instruction status is scanned.

    For the subroutine timers, the system counts the time and compares the timing values after the END instruction is executed. Use subroutine timers when the TMR instruction is executed not in every scan, but you need longer lasting timing and comparing.

Example 1

When M0 is ON, the instruction loads the setting value 50 to the timer T0, and T0 counts from 0 to 50. When the value of T0 matches 50, the contact of T0 is ON.

Example 2

When M0 is ON, the instruction loads the setting value 50 to the timer ST0. When the value of T0 is 25 and M0 switches from OFF to ON, then T0 counts up from 25 to 50, and the contact of T0 is ON (accumulative).

Example 3

When M0 is ON, the instruction loads the setting value 1000 to the timer T5, and T5 counts up from 0 to 1000, and the contact of T5 is ON.

Example 4

When M0 is ON, the instruction loads the setting value 1000 to the timer T5. When the value of T5 is 500 and M0 switches from OFF to ON, T5 counts up from 500 to 1000, and the contact of T5 is ON (accumulative).

Additional remarks

When you declare the operand **S1** in ISPSoft, select the data type TIMER for the general T timer. For an accumulative ST timer, specify the ST device.

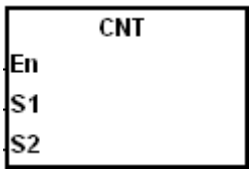
API	Instruction code	Operand	Function
1003	CNT	S_1, S_2	16-bit counter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1						○										
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Counter number

S_2 : Setting value for the counter

Explanation

1. This instruction changes a specified counter coil from OFF to ON, and then increments the value of the counter by 1. When the value of the counter matches the setting value, the contact of the counter is ON.

2. When the value of the counter matches the setting value, the instruction does not change the state of the contact and value of the counter if any more counting pulses are input. Use the RST instruction (API 1000) to reset the counter and enable counting again.

Example

When SM408 is ON for the first time, the instruction loads the setting value 10 to the counter C0 and the counter begins counting. After SM408 switches from OFF to ON ten times, the value in C0 is 10 and the contact of C0 is ON. After C0 is ON, if SM408 continues to switch from OFF to ON, the instruction does not increase the value in C0 after it reaches the setting value for C0.

Additional remarks

When you declare the operand S_1 in ISPSoft, select the data type COUNTER.

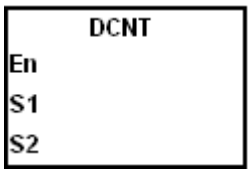
API	Instruction code	Operand	Function
1004	DCNT	S_1, S_2	32-bit counter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								○					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Counter value

S_2 : Setting value for the counter

Explanation

1. This instruction enables the 32-bit counter between HC0–HC255.
2. If you declare the operand S_1 in ISPSoft, you cannot select the CNT data type; instead, specify an HC device number.
3. For the count-up/count-down counters HC0–HC63, when the conditional contact of this instruction switches from OFF to ON, the counters count up by incrementing the values by 1 when SM621–SM684 are OFF or count down by decrementing the values by 1 when SM621–SM684 are ON.
4. Count-up counters HC64–HC199 count up by increasing the values by 1 when the conditional contact of the DCNT instruction switches from OFF to ON.
5. The counter stops counting when the DCNT instruction is OFF, but the instruction does not clear the original count value. Use the RST instruction to clear the count value and reset the contact to OFF.
6. Refer to the following pages for details on the high-speed counter HC200–HC255.

7. Since ES3 Series PLC is with various built-in input types, different maximum input frequency are suggested. See the table below for reference.

Model Name	Input Type	Input Number	Maximum Input Frequency
ES3 Series	Sinking / Sourcing	X0 ~ X7	200K Hz
		X10 ~ X17	10K Hz

Example 1

NETWORK1:

When PLC runs, the value of the counter HC0 is cleared and the counter counts because SM621 is OFF. At this time, SM408 is ON for the first time. So the instruction loads the setting value 10 to the counter HC0 and the counter begins counting.

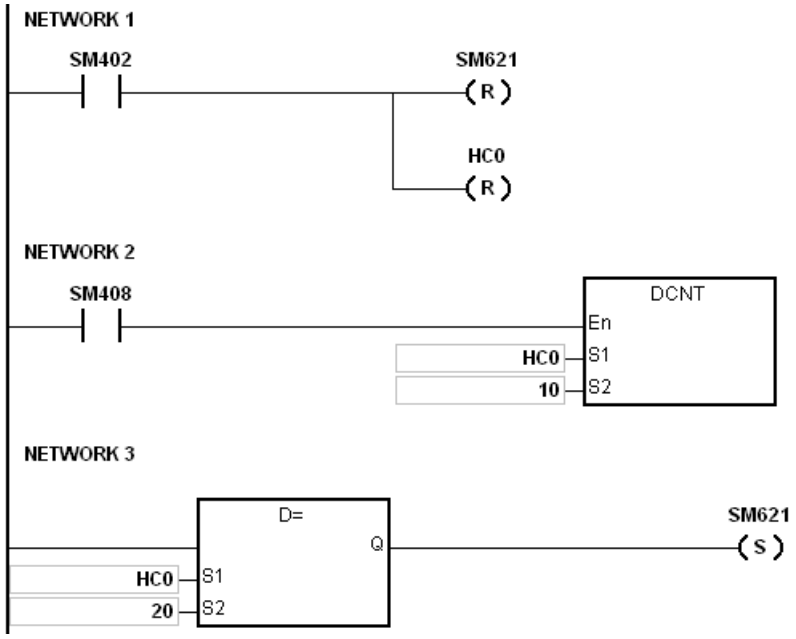
NETWORK2:

After SM408 switches from OFF to ON ten times, the value of the counter HC0 matches the setting value 10, and the contact of HC0 is ON. After HC0 is ON, the value of the counter keeps increasing because SM408 continues to change from OFF to ON even though the value of HC0 has reached the setting value.

NETWORK3:

When HC0 continues to count up and the value reaches the setting value 20, the counter counts down because SM621 is ON in the program. After SM408 switches from OFF to ON ten times and the value of HC0 decreases from 10 to 9, the contact of HC0 is OFF.

After the contact of HC0 is OFF, the value of HC0 still continues to decrease because SM408 continues to change from OFF to ON.



Additional remarks

For setting the mode of SM621–SM684, refer to the explanation of the 32-bit counter HC in Chapter 2.

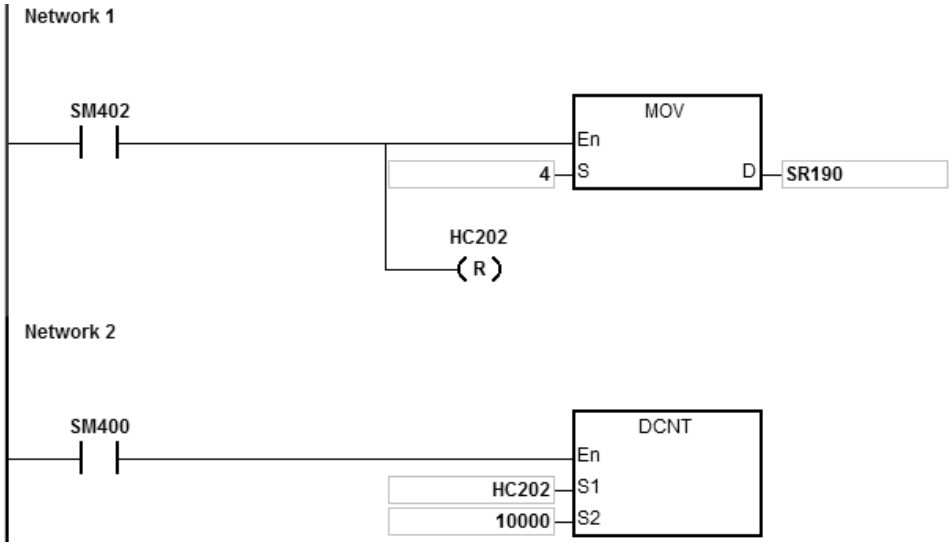
Example 2

NETWORK1:

When PLC runs, set the value of the counter HC202 to four time frequency (mode setting should be set before executing the DCNT instruction). And then the value of the counter HC202 is cleared.

NETWORK2:

After the value of the counter HC202 reached the setting value 1000, the contact of HC202 is ON.



Explanation of the high-speed counter:

ES3 Series high-speed counters can be divided into hardware counters (up to a maximum of 200KHz or 10KHz) and software counters (up to a maximum of 10KHz). Refer to hardware specification for more details on the input limit.

Hardware counter

Input / HC No.	X															
	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
HC200	P#1												R#4			
HC201	P	D#1											R			
HC202	A#1	B#1											R			
HC203	--#2	--											--			
HC204			P											R		
HC205			P	D										R		
HC206			A	B										R		
HC207			--	--										--		
HC208					P										R	
HC209					P	D									R	
HC210					A	B									R	
HC211					--	--									--	
HC212							P									R
HC213							P	D								R
HC214							A	B								R
HC215							--	--								--
HC216									P							
HC217									P	D						
HC218									A	B						
HC219									--	--						
HC220											P					
HC221											P	D				
HC222											A	B				
HC223											--	--				

Note 1: P: single-phase pulse input, D: Direction signal input, A and B: two phase two input, R: Reset signal input. Only

one out of four input modes can be used in PLC programing. For example, if HC200 is edited, the HC20-HC203 can no longer be edited.

Note 2: -- indicates that the counting mode is reserved and not available now. An empty box indicates no function.

Note3: refer to the SM/SR table for count up/down state selection and the number of times for frequency input.

Note 4: the R function (reset input) is disabled by default. Refer to the SM/SR comparison table for how to use R.

Take HC200 for example. SM291 switches to ON to start the R function and then the rising edge of X14 triggers clearing the value of HC200. If there is no built-in input points, including X14 ~ X17, in the PLC you are using, you can use external interrupts or use counter eraser instructions to clear the counters.

Software counter:

Input / HC No.	X															
	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
HC232													P			
HC233													P	D		
HC234													A	B		
HC235													UP#5	DN#5		
HC236															P	
HC237															P	D
HC238															A	B
HC239															UP	DN
HC240		P														
HC241	UP	DN														
HC242				P												
HC243			UP	DN												
HC244						P										
HC245					UP	DN										
HC246								P								
HC247							UP	DN								
HC248										P						
HC249									UP	DN						
HC250												P				

HC251											UP	DN				
HC252														P		
HC253																P

Note 5: UP: single phase count-up input (same as CW), DN: single phase count-down input (same as CCW)

The high-speed counters between HC200–HC255, are reserved devices inside PLC and are not listed in this table. It is not recommended to use these counters in a program.

The following table lists the high-speed counter, function, reset, reversing, and counting mode.

	Count-up/count-down function			**Starting the Reset function**	**Reversing the direction**	**Counting mode**
HC No.	SM No.	Attribute	Explanation	SM No.	SM No.	SR No.
HC200	SM300	R/W	Show/ set	SM291	SM281（Applicable for HC201）	SR190
HC201	SM301	R	Show			
HC202	SM302	R	Show			
HC204	SM304	R/W	Show/ set	SM292	SM282（Applicable for HC205）	SR191
HC205	SM305	R	Show			
HC206	SM306	R	Show			
HC208	SM308	R/W	Show/ set	SM293	SM283（Applicable for HC209）	SR192
HC209	SM309	R	Show			
HC210	SM310	R	Show			
HC212	SM312	R/W	Show/ set	SM294	SM284（Applicable for HC213）	SR193
HC213	SM313	R	Show			
HC214	SM314	R	Show			
HC216	SM316	R/W	Show/ set	--	SM285（Applicable for HC217）	SR194
HC217	SM317	R	Show			
HC218	SM318	R	Show			
HC220	SM320	R/W	Show/ set	--	SM286（Applicable for HC221）	SR195
HC221	SM321	R	Show			
HC222	SM322	R	Show			
HC232	SM332	R/W	Show/ set	--	SM287（Applicable for	SR196
HC233	SM333	R	Show			

	Count-up/count-down function			Starting the Reset function	Reversing the direction	Counting mode
HC No.	SM No.	Attribute	Explanation	SM No.	SM No.	SR No.
HC234	SM334	R	Show		HC233）	
HC235	SM335	R	Show			
HC236	SM336	R/W	Show/ set	--	SM288（Applicable for HC237）	SR197
HC237	SM337	R	Show			
HC238	SM338	R	Show			
HC239	SM339	R	Show			
HC240	SM340	R/W	Show/ set	--	--	Supports one time frequency and rising edge –triggered counting only
HC241	SM341	R	Show			
HC242	SM342	R/W	Show/ set			
HC243	SM343	R	Show			
HC244	SM344	R/W	Show/ set			
HC245	SM345	R	Show			
HC246	SM346	R/W	Show/ set	--	--	Supports one time frequency and rising edge –triggered counting only
HC247	SM347	R	Show	--	--	Supports one time frequency and rising edge –triggered counting only
HC248	SM348	R/W	Show/ set			
HC249	SM349	R	Show			
HC250	SM350	R/W	Show/ set			
HC251	SM351	R	Show			
HC252	SM352	R/W	Show/ set			
HC253	SM353	R	Show			

Note 1: All SM special flags in the above table are OFF by default.

Note 2: When SM under “Count-up/count-down function” is OFF, it indicates that the corresponding counter counts up or displays that it is counting up. If SM is ON, it indicates that the corresponding counter counts down or displays that it is counting down.

Note 3: The “ under Attribute indicates “Read only” and R/W indicates “Read/Write”.

Note 4: The SR special registers under “Counting mode” are 1 time frequency input by default. Use 2 for the input value for double frequency and 4 for four times frequency. Four times frequency is only applicable to the A/B 2-phase input counter. If the input value is not 1, 2 or 4 in SR, this indicates that the PLC uses one times frequency.

Note 5: All single-phase counters in the table count using one times frequency, and the rising-edge counting mode changes the input point from OFF to ON.

Note 6: P (Pulse input) and D (Direction) counters can reverse direction. When SM is ON, the counting direction (up/down) is reversed. For example, when the preset direction input is OFF, the counter counts up. When SM switches to ON, the counter changes to count down.

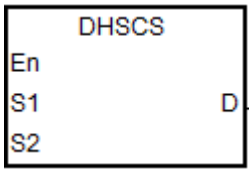
API	Instruction code			Operand	Function
1005	D	HSCS		**S_1, S_2, D**	Setting high-speed comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
D		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Counter number

S_2 : Comparative value

D : Comparison result

6

Explanation

1. Use this instruction with high-speed counters with numbers HC200 and above. If the value in the high-speed counter specified by **S_1** changes by increasing or decreasing by 1, the DHSCS instruction makes the comparison immediately. When the current value of the high-speed counter is equal to the comparative value specified in **S_2**, the device specified by **D** changes to ON. After that, the device specified by **D** remains ON even if the comparison result is that the current value and the comparative value are not equal.

2. If the device specified by **D** is Y0–Y7 and Y10–Y13, and the value of **S_2** is equal to the current value , the comparison result of the high-speed counter is output to the output terminals Y0–Y7 and Y10–Y13. Other Y devices are affected by the scan cycle, but this instruction updates all devices immediately and is not affected by the scan cycle.

3. The **D** operand can also specify an I interrupt device between 1200–1267.

4. The high-speed counters are divided into software counters and hardware counters. The available high-speed comparators and interrupt device numbers are listed in the following table.

Type	Range of counter numbers	High-speed comparator number	High-speed interrupt device number
Hardware counter	HC200 - HC203	Comparator: HCC00-HCC03	I200-I203
	HC204 - HC207	Comparator: HCC04-HCC07	I210-I213
	HC208 - HC211	Comparator: HCC08-HCC11	I220-I223
	HC212 - HC215	Comparator: HCC12-HCC15	I230-I233
	HC216 - HC219	Comparator: HCC16-HCC19	I240-I243
	HC220 - HC223	Comparator: HCC20-HC223	I250-I253
Software counter	HC232 - HC253	-	I260-I267

5. Explanation of the hardware comparators for DHSCS, DHSCR, DHSCY and DHSZ instructions:

   - Every one group of hardware counters shares 4 high-speed comparators. One DHSCS, DHSCR or DHSCY instruction occupies 1 high-speed comparator. One DHSZ instruction uses 2 high-speed comparators.
   - During program editing, every group of hardware counters can use 4 high-speed comparators at most for DHSCS, DHSCR, DHSCY or DHSZ instructions; otherwise, a syntax error occurs.

6. Explanation of the software comparators for DHSCS, DHSCR and DHSCY instructions:

   - There are 8 software comparators to compare the Set or Reset function. Each DHSCS, DHSCR or DHSCY instruction uses one high-speed comparator.
   - The software comparators compare the interrupt by assigning a corresponding software comparator according to the interrupt numbers. Note that the same interrupt number cannot be used repeatedly.
   - For DHSCS, DHSCR or DHSCY instructions, the number of Set or Reset comparators cannot exceed eight occurrences in the program; otherwise, a syntax error occurs.

7. Explanation of the software comparators for DHSZ instruction:

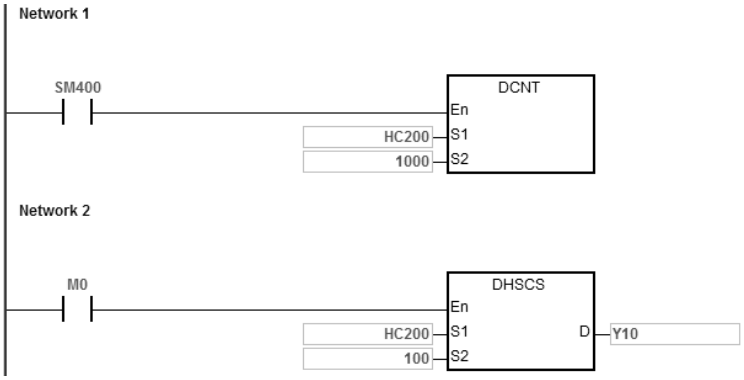
   - There are eight software comparators for the zone comparison. One DHSZ instruction uses one comparator.
   - DHSZ instruction can use a maximum of eight software comparators; otherwise, a syntax error occurs if more than eight comparators are used.

8. The instructions DHSCS, DHSCR, DHSCY and DHSZ cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example 1

When M0 is ON, the DHSCS instruction is executed.

When the current value of HC200 changes from 99 to 100 or from 101 to 100, Y10 is ON, which outputs to the external output terminal Y10 in real time, and remains ON.

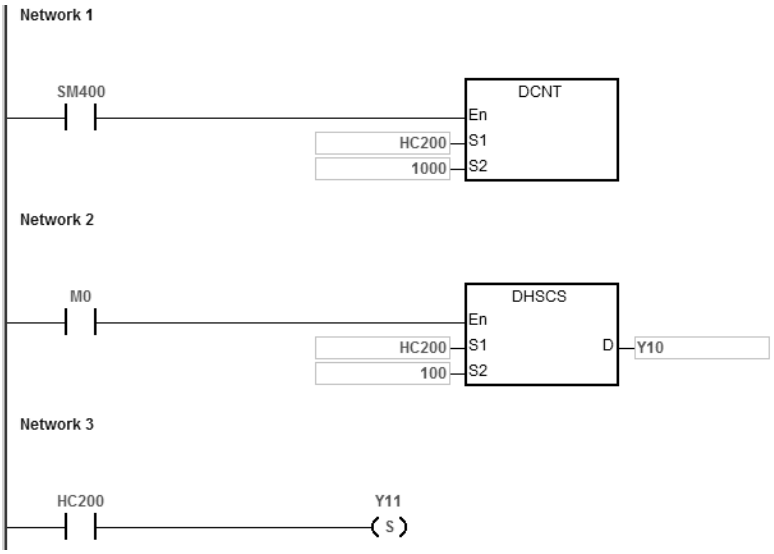


Example 2

The Y output of DHSCS instruction is different from the general Y output.

1. When M0 is ON, the DHSCS instruction is executed. When the current value of HC200 changes from 99 to 100 or from 101 to 100, Y10 outputs its state to the external output terminal immediately, and is not affected by the program scan time.

2. When the current value of HC200 changes from 99 to100, the contact of HC200 is ON immediately. When SET Y0.11 is executed, Y11 is still affected by the scan time, and outputs its state only after END is passed.

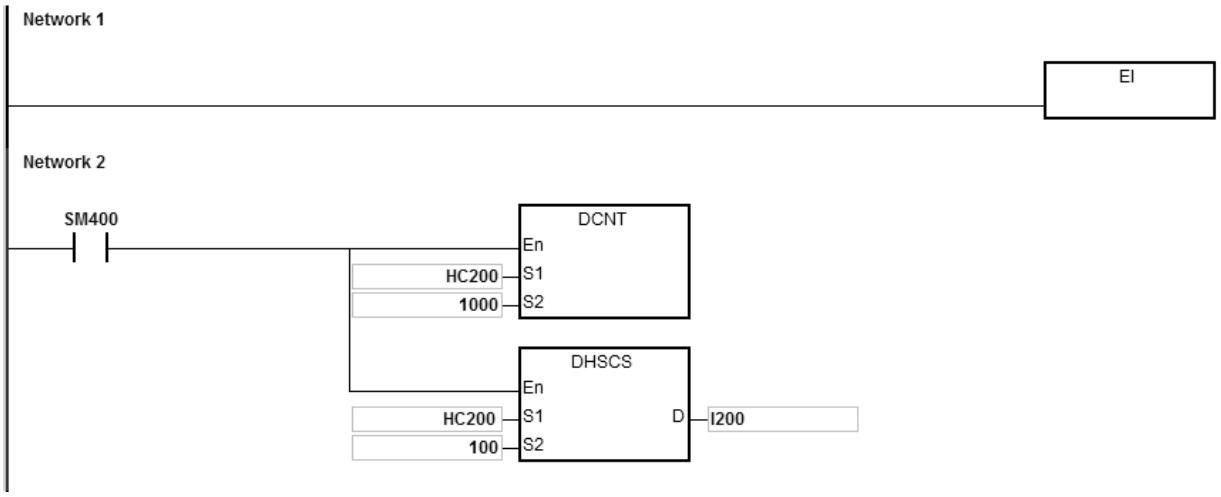


Example 3

Using an interrupt in hardware high-speed comparison.

When the current value of HC200 changes from 99 to100 or 101 to100, the program jumps to the interrupt pointer to execute the interrupt program, and Y10 is ON.

Main program:



I200 interrupt program:

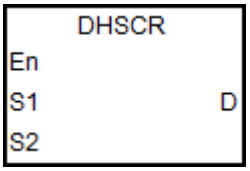
API	Instruction code			Operand	Function
1006	D	HSCR		**S_1, S_2, D**	Resetting high-speed input comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
D		○	○	○			○	○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Counter number

S_2 : Comparative value

D : Comparison result

6

Explanation

1. Use this instruction with the high-speed counter numbered HC200 and above. If the value in the high-speed counter specified by **S_1** changes by increasing or decreasing, the DHSCR instruction makes the comparison immediately. When the current value of the high-speed counter is equal to the comparative value specified in **S_2**, the device specified by **D** changes to OFF. After that, the device specified by **D** remains OFF even if the comparison result is that the current value and the comparative value are not equal.

2. If the device specified by **D** is Y0–Y7 and Y10–Y13, and the comparative value of **S_2** is equal to the current value of the counter, the comparison result is output to the external output terminals Y0–Y7 and Y10–Y13. Other Y devices are affected by the scan cycle, but this instruction updates all devices immediately and is not affected by the scan cycle.

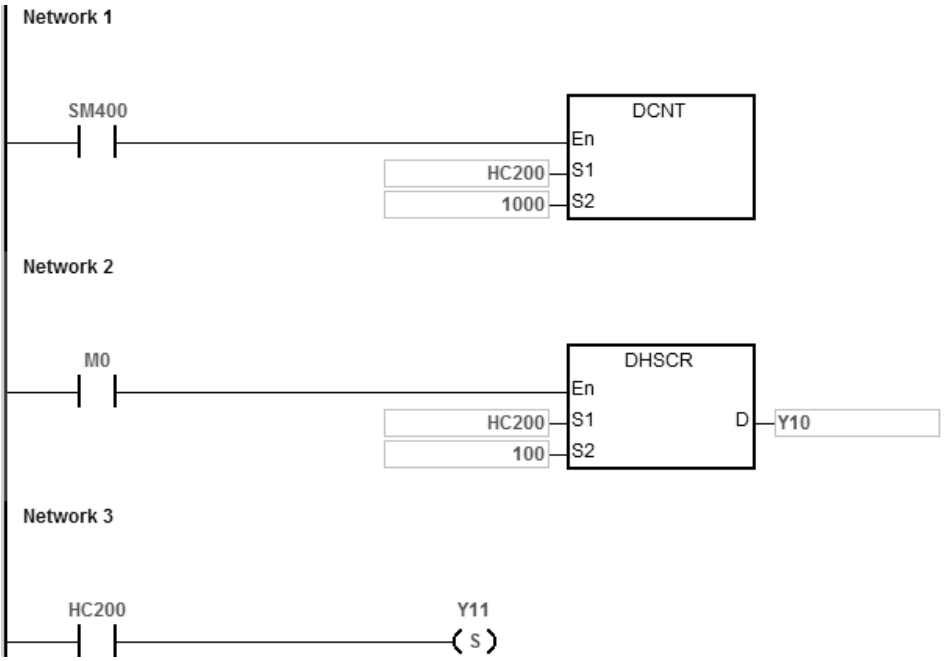
3. The **D** operand can also specify the HC device to reset, and is limited to the condition in which the high-speed counter number is the same as that of **S_1**.

4. Refer to the DHSCS instruction (API 1005) for more information.

Example 1

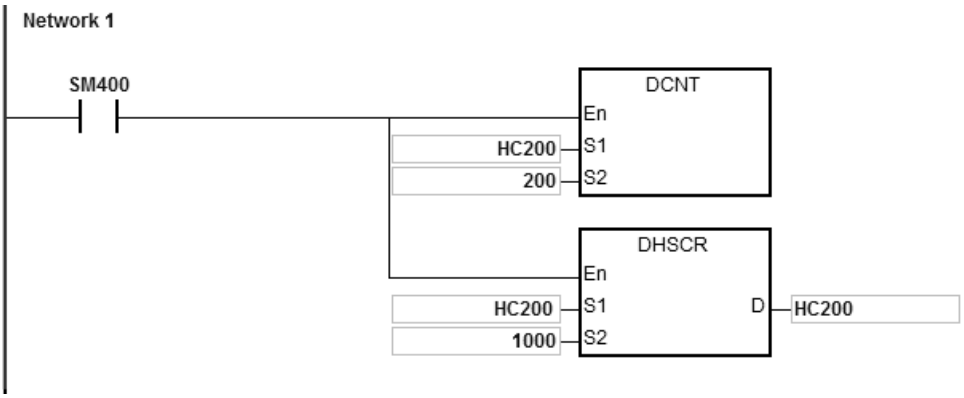
1. When M0 is ON and HC200 changes its current value from 99 to 100 or from 101 to 100, Y10 is reset to OFF.

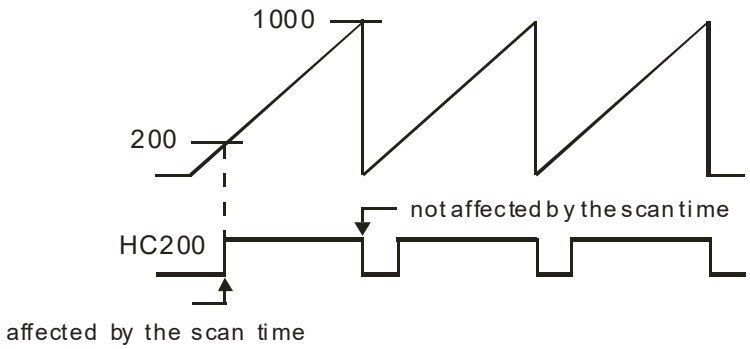
2. When HC200 changes its current value from 199 to 200, the contact of HC200 is ON and Y0.11 is ON, but the output is delayed by the program scan time.



Example 2

If you specify HC200 as the hardware high-speed counter of the same number, the contact of HC200 is reset to OFF when HC200 changes its current value from 999 to 1000 or from 1001 to 1000.




1000
200
not affected by the scan time
HC200
affected by the scan time

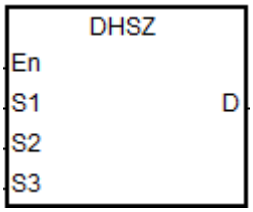
API	Instruction code			Operand	Function
1007	D	HSZ		**S_1, S_2, S_3, D**	High-speed input zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								●					○	○		
S_3								●					○	○		
D		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
S_3			●				●						
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Counter number

S_2 : Lower bound of the comparison zone

S_3 : Upper bound of the comparison zone

D : Comparison result (3 consecutive devices)

Explanation

1. Use this instruction with the high-speed counter numbers HC200 and above. The lower bound of **S_2** must be less than the upper bound of **S_3**. If you do not set the zone limit values properly, the PLC automatically adjusts them.

2. If **S_1** specifies a software counter and the specified counter changes by increasing or decreasing by 1 in value, the DHSZ instruction makes the comparison immediately. The comparison condition and output state are shown in the following table.

Comparison condition	**D**+0 state	**D**+1 state	**D**+2 state
The count value of **S_1** < the lower bound (**S_2**)	ON	OFF	OFF
The lower bound (**S_2**) <= the count value of **S_1** < the upper bound (**S_3**)	OFF	ON	OFF
The count value of **S_1** >= the upper bound (**S_3**)	OFF	OFF	ON

Note: You must set the lower bound (**S_2**) to be less than the upper bound (**S_3**). If you set the zone boundaries incorrectly, the PLC automatically makes the adjustment.

3. If **S_1** specifies a hardware counter and the value of the specified counter reaches the lower bound (**S_2**) or the upper bound (**S_3**), the DHSZ instruction makes the comparison immediately according to the count direction (up/down). The comparison condition and output state are shown in the following table.

Count direction	Comparison condition	**D**+0 state	**D**+1 state	**D**+2 state
Count up	The count value of **S_1** == the lower bound (**S_2**)	OFF	ON	OFF
	The count value of **S_1** == the upper bound (**S_3**)	OFF	OFF	ON
Count down	The count value of **S_1** == the lower bound (**S_2**)	ON	OFF	OFF
	The count value of **S_1** == the upper bound (**S_3**)	OFF	ON	OFF

4. If the device specified by **D** is Y0–Y7 and Y10–Y13, the comparison result is output to the external output terminals Y0–Y7 and Y10–Y13. Other Y devices are affected by the scan cycle, but this instruction updates all devices immediately and is not affected by the scan cycle.

5. Refer to the DHSCS instruction (API 1005) for more information on the high-speed zone comparison.

6

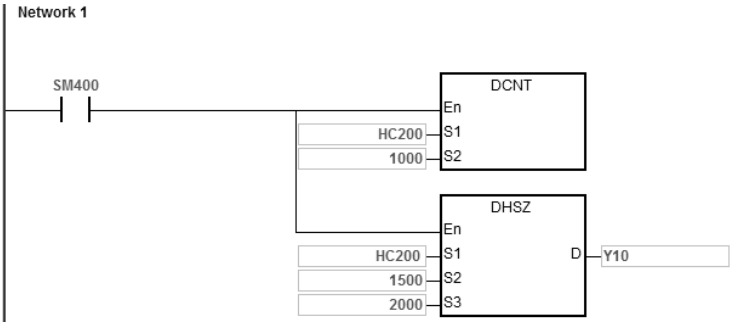
Example

1. When **D** is specified as Y10, Y11–Y12 are also specified automatically.

2. The instruction compares the current value in HC200 with the upper/lower bound (1500/2000) of the comparison zone, and one of Y10–Y12 is ON according to the comparison result.

3. When the current value in HC200 <1500, Y10 is ON. When 1500<= the current value in HC200<2000, Y11 is ON.

   When the current value in HC200>=2000, Y12 is ON.



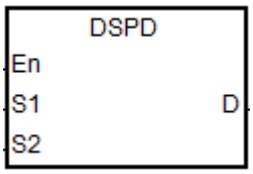
API	Instruction code			Operand	Function
1008	D	SPD		**S_1, S_2, D**	Detecting speed

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							○									
S_2								○	○				○	○		
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1												●	
S_2			●				●						
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : Counter value

S_2 : Setting value for the cycle time

D : Number of pulses from the previous scan cycle

Explanation

1. This instruction requires that you use **S_1** with the DCNT instruction (API 1004) to enable the high speed counter with counter numbers above HC200 (including HC200).

2. The time units for **S_2** (the setting value for the cycle time) are millisecond (ms). The setting must be between 10–1000. When the value is out of range, the PLC executes the instruction with **S_2** at the minimum value or the maximum value and there are no error messages.

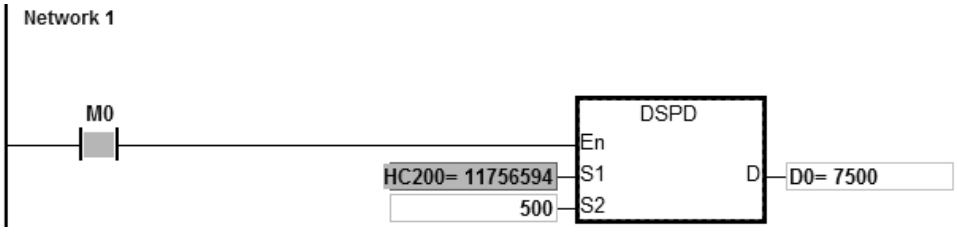
3. When the count reaches the setting value in **S_2** , this instruction stores the number of pulses in the device specified by **D**, and is not affected by the PLC scan cycle.

4. This instruction has no limitation when editing, but it only allows eight sets of speed detection instructions to run simultaneously. The system ignores the ninth set of the speed detection instruction and there are no error messages. When executing this instruction, the setting values for the operand are recorded, and during the execution of this instruction, you cannot edit the parameters.

Example

You can use the DSPD instruction for speed detection where there is an input pulse signal in M0. When M0 is ON, the instruction updates the number of pulses counted by HC200 in D0 every 500ms.

In the following example, the value in D0 is 7500 and the actual pulse input frequency of M0 is 15kHz (7500/500ms).



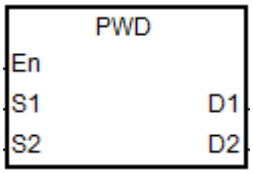
API	Instruction code	Operand	Function
1009	PWD	**S_1, S_2, D**	Detecting pulse width

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								○	○				○	○		
D_1								○								
D_2		○	○	○												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●				●							
D_1			●				●						
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Number of the input point

S_2 : Unit of measurement

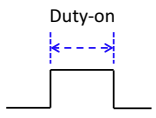
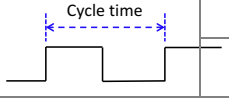
D_1 : Pulse width detection time (32-bit value)

D_2 : Update flag

Explanation

1. S_1 supports the following 12 inputs, X0~X13, but S_1 cannot share the same inputs with the high speed counter.

2. S_2 is the unit of measurement. The instruction is not executed if the setting value of S_2 is not a valid S_2 code from the following table.

S_2 code	Measurement Unit	Detection range	Frequency range	Remark
0	1us		1Hz - 10kHz	
1	1ms		0.02Hz - 100Hz	
2	10ns		10Hz - 1MHz	Odd-numbered inputs are NOT supported.
4	1us		1Hz - 10kHz	
5	1ms		0.02Hz - 100Hz	
Other values	The instruction is not executed.			

3. The instruction stores the pulse width detection time (32-bit value) in **D_1** and the detection range is 0–100,000,000. If the value is over the maximum value, it is processed as the maximum value. If the value is 0, that means is no input switched from ON to OFF during the execution of this instruction.

4. **D_2** is the update flag. Whenever the detection of the **S_1** input is completed and the instruction is scanned,, the updated flag switches to ON for one scan cycle time. You can check if the detection value has been updated with the update flag. When the system executes the instruction for the first time, the update flag resets to OFF.

5. When the value in **S_2** is 0, 1 or 2, refer to the timing diagram below for the procedures performed, such as storing detection values and updating flags during the execution of the instruction. The timer starts when the **S_1** input switches from OFF to ON as it is shown in the position ① of the following diagram. The instruction stores the detection time when the **S_1** input switches from ON to OFF as shown in the position ② of the following diagram.

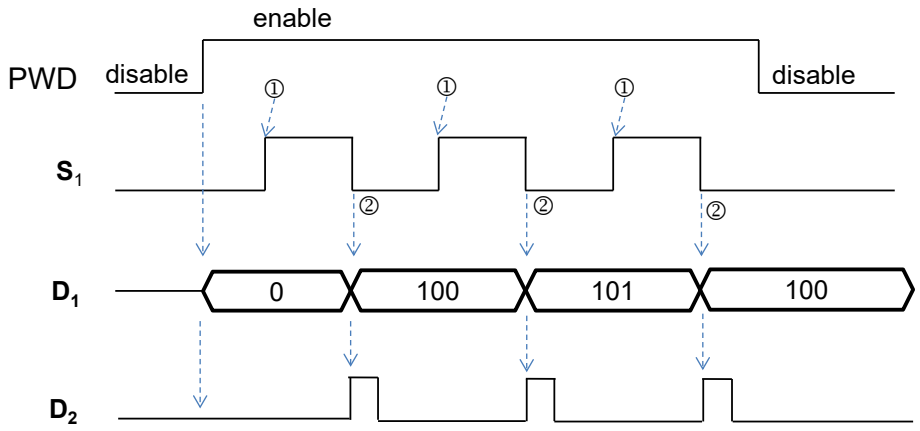


Figure 1 Detection mode when the value in **S_2** is 0, 1 or 2

6. When the value in **S_2** is 4 or 5, refer to the timing diagram below for the procedures performed, such as storing detection values and updating flags during the execution of the instruction. The timer starts when the **S_1** input switches from OFF to ON as it is shown in the position ① of the following diagram. The instruction stores the detection time when the **S_1** input switches from OFF to ON as shown in the position ② of the following diagram.

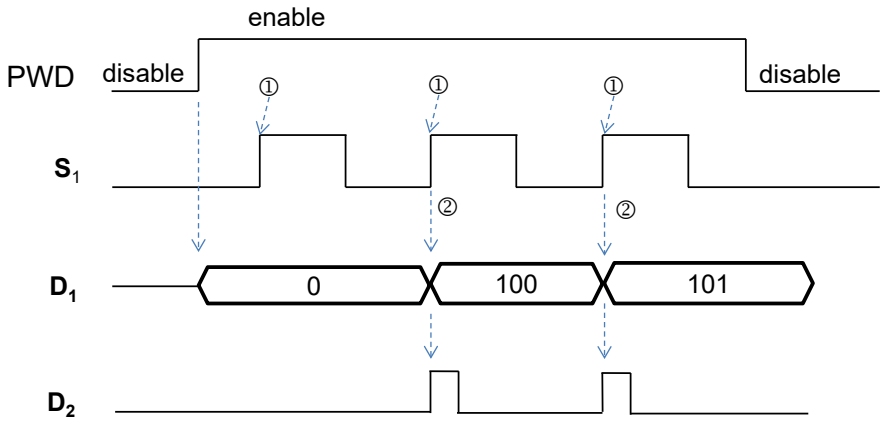


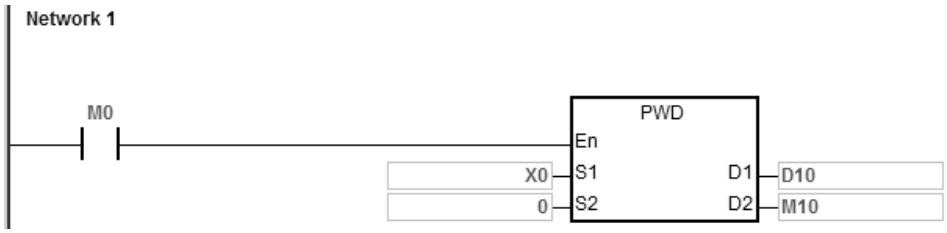
Figure 2 Detection mode when the value in **S_2** is 4 or 5

7. This instruction has no limitation during editing, but it only allows eight sets of pulse width detection instructions to run simultaneously. The system ignores the ninth or later sets of the pulse width detection instruction and there are no error messages. When executing this instruction, the setting values for the operands are recorded, and you cannot edit the parameters during execution.

8. Before executing this instruction, check the input hardware response time and the pulse time set in HWCONFIG. For example, when the value in **S_2** is set to 0 or 2, that means the unit of time measurement is microseconds (μs). Set the **S_1** input value to 0 to disable the Input Point Filter Time in HWCONFIG.

Example

Suppose there is a pulse signal of 10kHz in the input M0. When M0 is ON, the PWD instruction detects the input signal on X0 with the pulse width stored in D10/D11 (32-bit data), the time unit is set to 0, and the detected pulse width from D10 is 50μs.



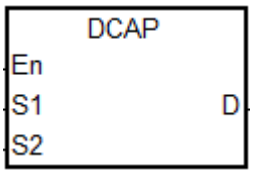
API	Instruction code			Operand	Function
1010	D	CAP		**S_1, S_2, D**	Capturing the high-speed count value in the external input interrupt

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2							○									
D								○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2												●	
D			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
-	-	ES3

Symbol



S_1 : External interrupt input point number

S_2 : High-speed counter number

D : Register for storing the captured value

Explanation

1. You can use only the 16 input points X0–X7 and X10–X17 of the PLC in **S_1**. Use one of these input points with the external interrupt service program to start the function. Note that **S_1** cannot share the same input point with the high-speed counter.

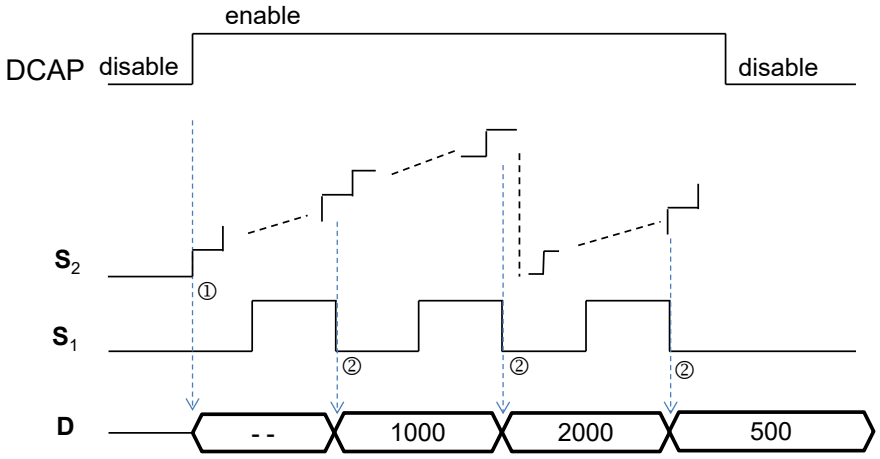
2. Select the high-speed counter HC device in **S_2**. You must use the HC device with the DCNT instruction (API 1004) to start the counting function.

3. The instruction stores the captured value from the high-speed counter (32-bit) in **D** when the interrupt occurs. The instruction stores data when the interrupt occurs, and is not affected by the PLC program scanning.

4. The instruction operation is shown below. The input interrupt is triggered by the falling edge.

   ① → When the execution of the instruction starts, the value in **D** does not change and you can enter the default setting value.

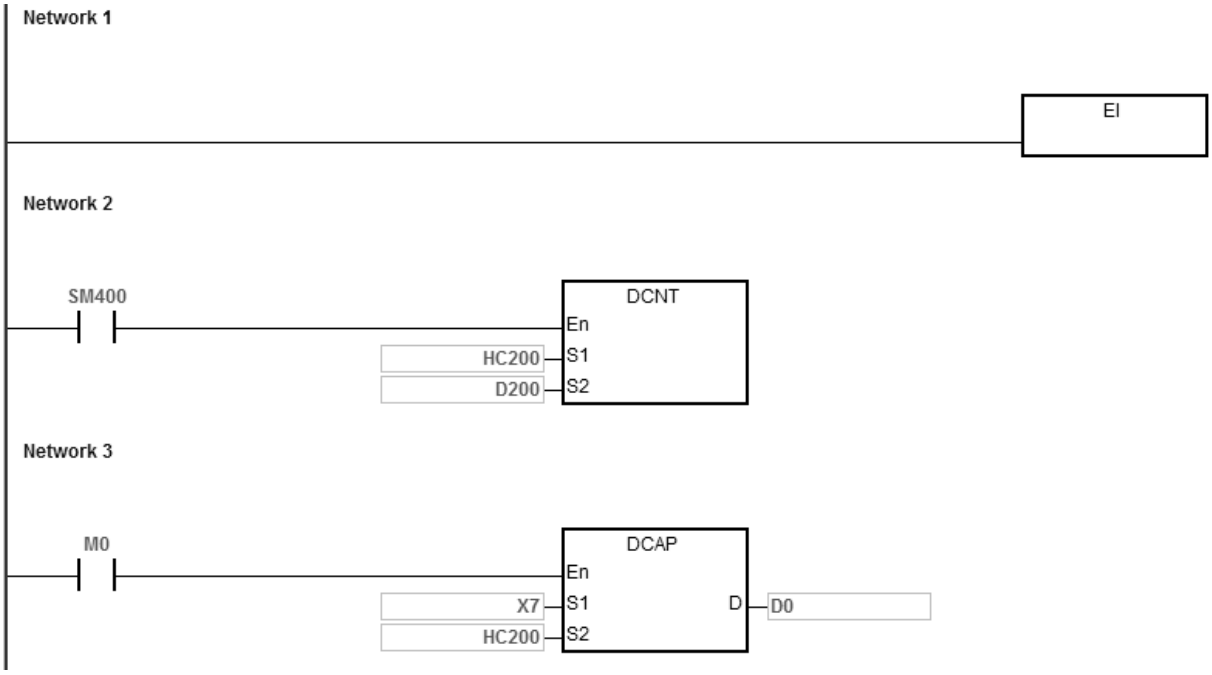
   ② → When the interrupt in **S_1** occurs, the instruction captures the value of the counter specified by **S_2** immediately and stores it in D.



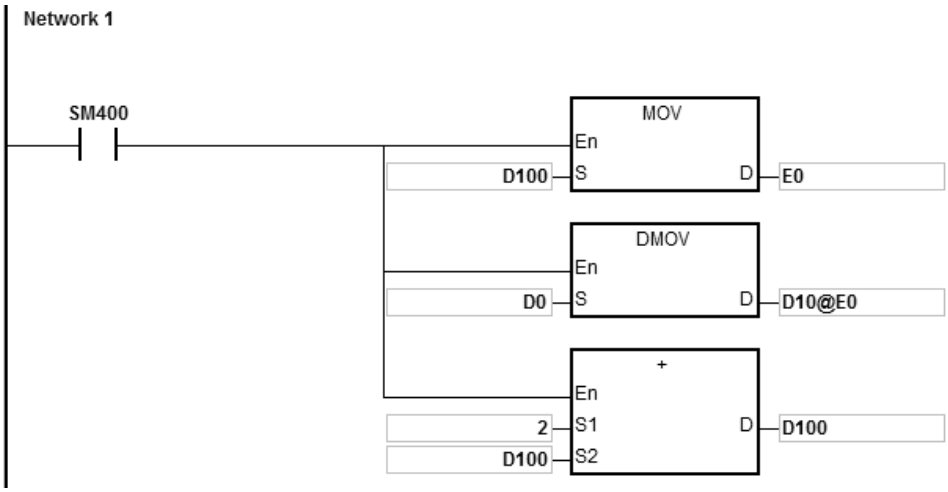
5. The instruction can start DCAP instructions for four different input points at most. If you set one input point as the external interrupt triggered by the rising edge and falling edge, the instruction captures the value when the input is triggered by the rising edge and by falling edge respectively, and stores the count value in the device specified by **D**. When two instructions specify the same interrupt input point, the one that starts first uses the interrupt input point first.

6. Set the HC device number in **S_2**. It is recommended that you use the high-speed counters between HC200–HC255. For details on the counters, refer to the explanation of the DCNT instruction (API 1004).

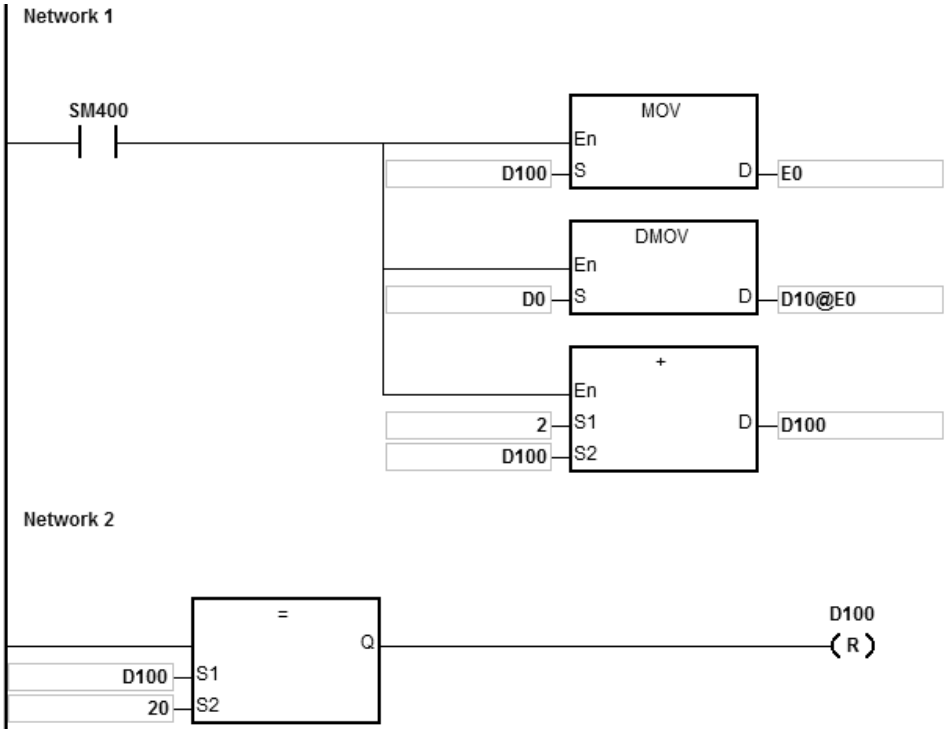
Example



External interrupt is triggered by the rising edge in X7.



External interrupt is triggered by the falling edge in X7.



Additional remarks

1. When M0 is ON, the DCAP instruction is enabled. When an external interrupt occurs in X0.7, the instruction captures the value in HC200 and stores it in (32-bit) D0.

2. When the external input interrupt is triggered by the rising edge once, the instruction modifies E0 to 0 by setting D100, stores the count value in D0 in D10 by modifying E0, and the value in D100 is 0+2.

3. When the external input interrupt is triggered by the falling edge one time, the instruction modifies E0 to 2 by setting D100, stores the count value (10+E0=12) in D0 in D12 by modifying E0, and the value in D100 is 0+2. When the value in D100 is 20, D100 is cleared to 0.

4. If the external interrupt is triggered by the rising edge and falling edge five times respectively, the instruction captures the value 10 times and stores the captured values in D10, D12…D28.

   The 1st captured value= D10

   The 2nd captured value= D12

   …

   …

   The 10th captured value=D28

   The 11th captured value=D10

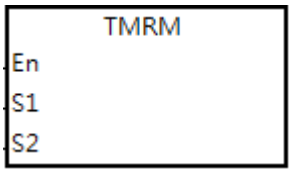
API	Instruction code	Operand	Function
1011	TMRM	S_1, S_2	16-bit timer (10ms)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					○											
S_2								○				○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1											●		
S_2		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Timer number

S_2 : Setting value for the timer

Explanation

This instruction uses 10ms as the unit of time.

Refer to the explanation of the TMRH instruction (API 1002) for details.

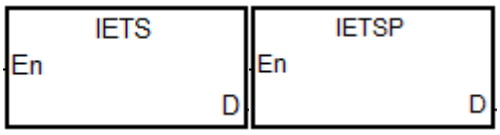
API	Instruction			Operand	Description
1012		IETS	P	**D**	The start of the instruction execution time measurement

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	—

Symbol



D : The time measurement result

Explanation

1. The IETS instruction need be used with the API1013 IETE instruction together in order to measure the time for the execution of the instruction in a PLC program which is specified to execute. The unit for the measured time is 1us.

2. When the IETS instruction is enabled, the timing starts immediately until the IETE instruction is also executed. The measurement result is stored in **D** device.

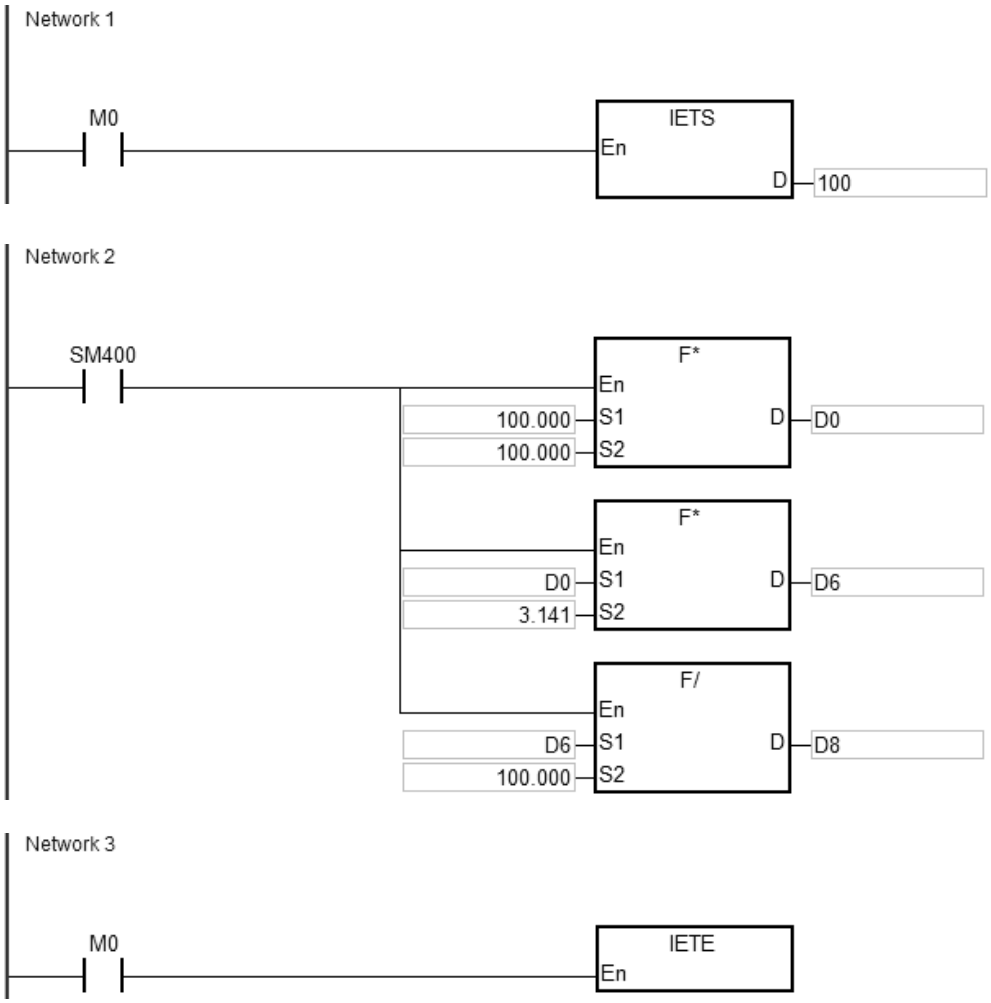
3. Minimum and maximum time measurement results are 0us and 32767us respectively. After the **IETS** instruction is enabled, the PLC will automatically finish the time measurement and store the measurement result in **D** device if no **IETE** instruction has been scanned and the PLC program scanning reaches the **END** instruction.

4. For the instructions IETS and IETE, there is no limit to how many of them are written in the program. But only one set of IETS and IETE can be enabled every time the scan is executed. If IETS is enabled repeatedly for measuring time, the timing of enabling the last IETS instruction is taken as the start of the time measurement. On the contrary, if the execution of multiple IETE instructions is completely finished, the PLC will see the point when the first IETE instruction is disabled as the end point when the time measurement is finished.

5. The IETS instruction is usually used to measure the running time of a PLC program such as interrupt service program function blocks and etc. Since PLC's time-measurement resource will be occupied as the time measurement function is enabled, we suggest the two instructions should be removed after the measuring is completed in order to avoid occupying the PLC resource during the normal execution.

Example

Calculate the instruction execution time based on the formula for the floating point number operation and the operation result is stored in D100.



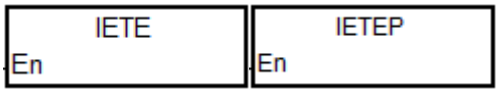
API	Instruction			Operand	Description
1013		IETE	P	—	The end of the instruction execution time measurement

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	—

Symbol



Explanation

The IETE instruction should be used with the API1012 IETS instruction together. Refer to the explanation of the API 1012 instruction for more information.

6.12 Shift Instructions

6.12.1 The List of Shift Instructions

The following table lists the Shift instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1100	SFTR	–	✓	Shifting the states of devices to the right
1101	SFTL	–	✓	Shifting the states of devices to the left
1102	WSFR	–	✓	Shifting the data in word devices to the right
1103	WSFL	–	✓	Shifting the data in word devices to the left
1104	SFWR	–	✓	Shifting the data and writing it into a word device
1105	SFRD	–	✓	Shifting the data and reading it from a word device
1106	SFPO	–	✓	Reading the latest data from the data list
1107	SFDEL	–	✓	Deleting data from the data list
1108	SFINS	–	✓	Inserting the data into the data list
1109	MBS	–	✓	Shifting matrix bits
1110	SFR	DSFR	✓	Shifting the values of the bits in registers by n bits to the right
1111	SFL	DSFL	✓	Shifting the values of the bits in registers by n bits to the left
1112	BSFR	–	✓	Shifting the states of n bit devices by one bit to the right
1113	BSFL	–	✓	Shifting the states of n bit devices by one bit to the left
1114	NSFR	–	✓	Shifting n registers to the right
1115	NSFL	–	✓	Shifting n registers to the left

6.12.2 Explanation of Shift Instructions

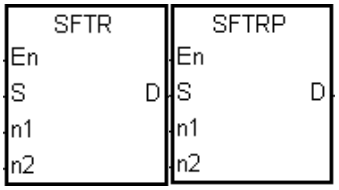
API	Instruction code			Operand	Function
1100		SFTR	P	**S, D, n_1, n_2**	Shifting the states of devices to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●				●								
D		●	●	●				●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D	●												
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device where the value is shifted from

D : First device where the value is shifted to

n_1 : Length of the data to be shifted

n_2 : Number of bits in a group

Explanation

1. This instruction divides the states of the **n_1** bit devices starting from **D** into groups (**n_2** bits in a group), and shifts these groups to the right. This instruction then shifts the states of the **n_2** bit devices starting from **S** to the devices starting from **D** to fill the vacancy.

2. In general, the SFTRP pulse instruction is used.

3. The operand **n_1** must be between 1–1024. The operand **n_2** must be between 1–**n_1**.

Example 1

1. When M500 switches from OFF to ON, the instruction divides the states of the sixteen bit devices starting from M0 to M15 into groups (four bits in a group), and shifts these groups to the right.

2. The shift of the states of the bit devices to the right during a scan is shown below.

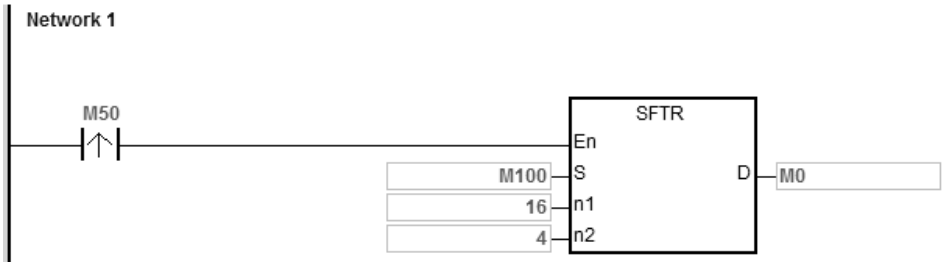
❶ M3-M0 → Being carried

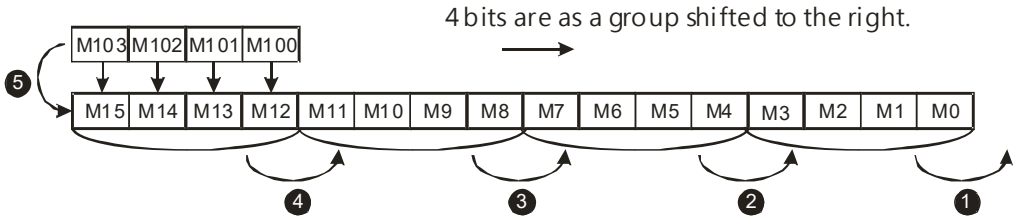
❷ M7-M4 → M3-M0

❸ M11-M8 → M7-M4

❹ M15-M12 → M11-M8

❺ M103-X100 → M15-M12





6

Example 2

1. When X50 switches from OFF to ON, the instruction divides the states of the sixteen bit devices starting from M0 to M15 into groups (five bits as a group), and shifts these groups to the right.

2. The shift of the states of the bit devices to the right during a scan is shown below.

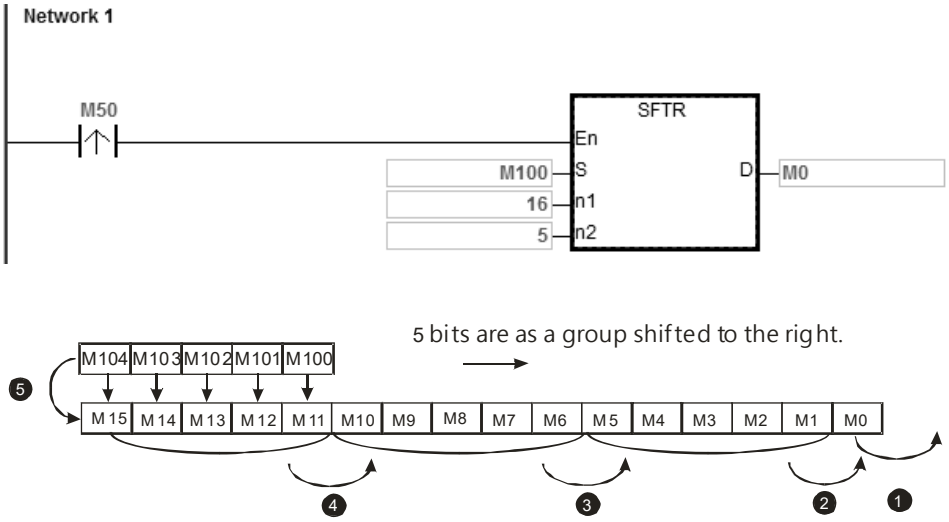
❶ M0 → Being carried

❷ M5 → M0

❸ M10-M6 → M5-M1

❹ M15-M11 → M10-M6

❺ M104-M100 → M15-M11



Additional remarks

1. If **S**+n_2-1 or **D**+n_1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If n_1 not between 1–1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If n_2 is not between 1–n_1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

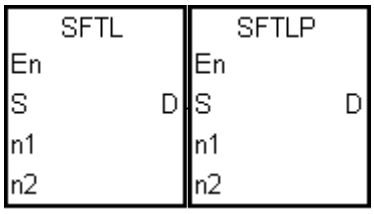
API	Instruction code			Operand	Function
1101		SFTL	P	**S, D, n_1, n_2**	Shifting the states of devices to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●				●								
D		●	●	●				●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D	●												
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device where the value is shifted from

D : First device where the value is shifted to

n_1 : Length of the data to be shifted

n_2 : Number of bits in a group

Explanation

1. This instruction divides the states of the **n_1** bit devices starting from **D** into groups (**n_2** bits in a group), and shifts these groups to the left. This instruction then shifts the states of the **n_2** bit devices starting from **S** to the devices starting from **D** to fill the vacancy.
2. In general, the SFTLP pulse instruction is used.
3. The operand **n_1** must be between 1–1024. The operand **n_2** must be between 1–**n_1**.

Example 1

1. When M50 switches from OFF to ON, the instruction divides the states of the sixteen bit devices starting from M0 to M15 into groups (four bits in a group), and shifts these groups to the left.
2. The shift of the states of the bit devices to the left during a scan is shown below.

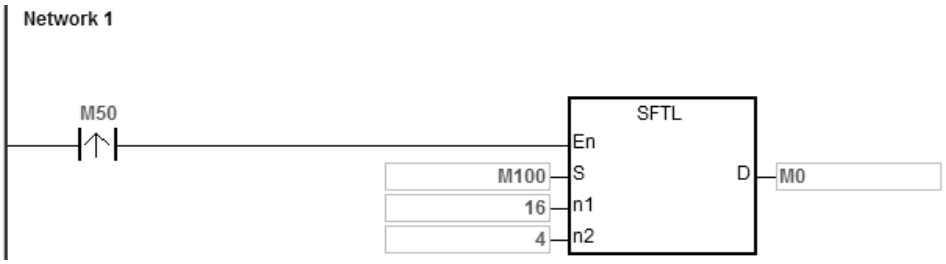
❶ M15-M12 → Being carried

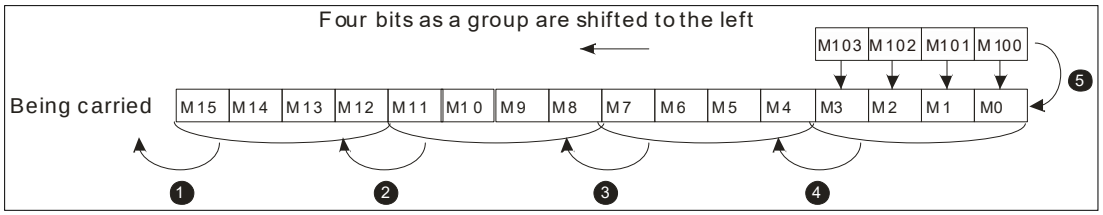
❷ M11-M8 → M15-M12

❸ M7-M4 → M11-M8

❹ M3-M0 → M7-M4

❺ M103-M100 → M3-M0





Example 2

1. When M50 switches from OFF to ON, the instruction divides the states of the sixteen bit devices starting from M0 to M15 into groups (five bits in a group), and shifts these groups to the left.

2. The shift of the states of the bit devices to the left during a scan is shown below.

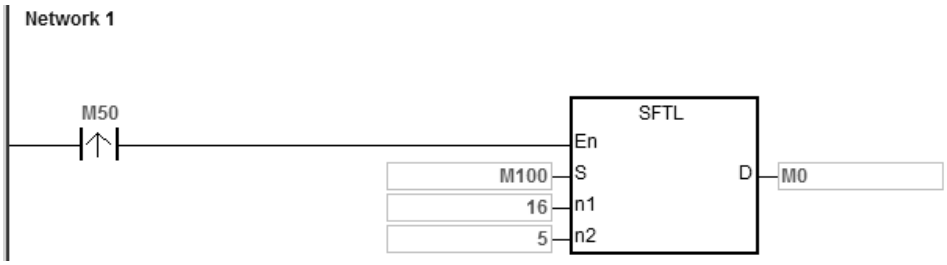
❶ M15 → Being carried

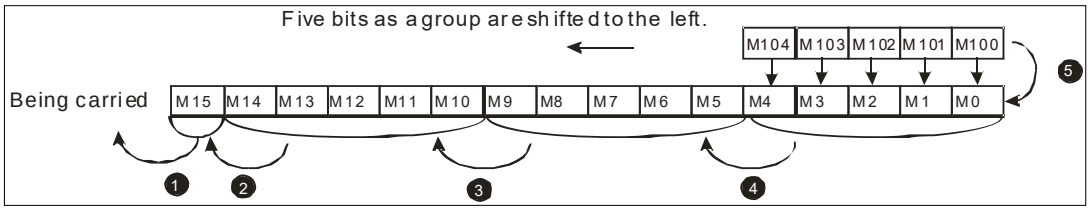
❷ M10 → M15

❸ M9-M5 → M14-M10

❹ M4-M0 → M9-M5

❺ M104-M100 → M4-M0





Additional remarks

1. If **S**+n_2-1 or **D**+n_1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If n_1 is not between 1–1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If n_2 is not between 1–n_1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

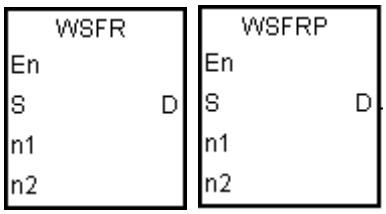
API	Instruction code			Operand	Function
1102		WSFR	P	**S, D, n_1, n_2**	Shifting the data in word devices to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●		●	●							
D		●			●	●		●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device where the value is shifted from

D : First device where the value is shifted to

n_1 : Length of the data to be shifted

n_2 : Number of bits in a group

Explanation

1. This instruction divides the data in the **n_1** word devices starting from **D** into groups (**n_2** words in a group), and shifts these groups to the right. This instruction then shifts the data in the **n_2** word devices starting from **S** to the devices starting from **D** to fill the vacancy.

2. In general, the WSFRP pulse instruction is used.

3. The operand **n_1** must be between 1–512. The operand **n_2** must be between 1–**n_1**.

Example 1

1. When M0 switches from OFF to ON, the instruction divides the data in the sixteen word devices starting from D20 to D35 into groups (four words in a group), and shifts these groups to the right.

2. The shift of the data in the word devices to the right during a scan is shown below.

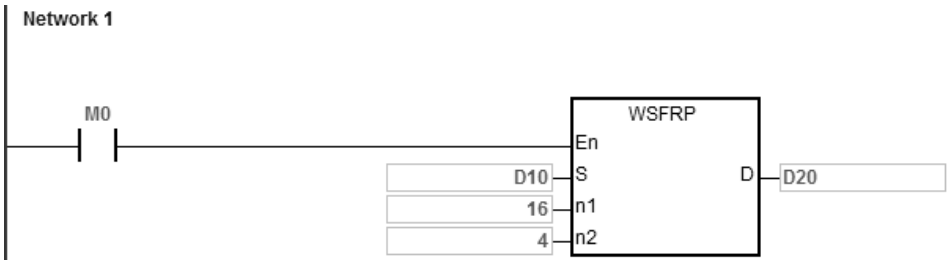
❶ D23-D20 → Being carried

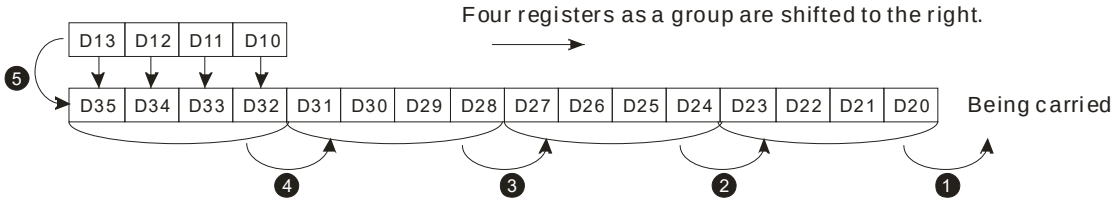
❷ D27-D24 → D23-D20

❸ D31-D28 → D27-D24

❹ D35-D32 → D31-D28

❺ D13-D10 → D35-D32





6

Example 2

1. When M0 switches from OFF to ON, the instruction divides the data in the sixteen word devices starting from D20 to D35 into groups (five words in a group), and shifts these groups to the right.

2. The shift of the data in the word devices to the right during a scan is shown below.

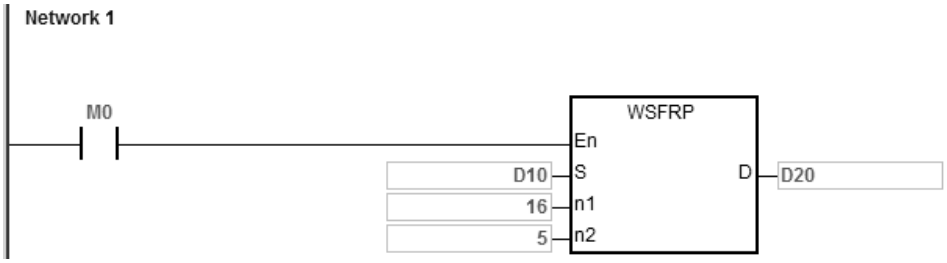
❶ D20 → Being carried

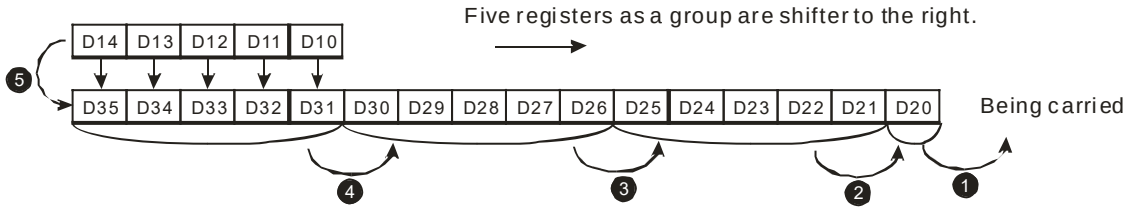
❷ D25 → D20

❸ D30-D26 → D25-D21

❹ D35-D31 → D30-D26

❺ D14-D10 → D35-D31





Additional remarks

1. If **S**+n_2-1 or **D**+n_1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If n_1 is not between1–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If n_2 is not between 1–n_1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1103		WSFL	P	**S, D, n_1, n_2**	Shifting the data in word devices to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
D					●	●		●								
n_1								●	●		○	○	○	○		
n_2								●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n_1		●			●	●							
n_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : First device in which the value is shifted

D : First device in which the value is shifted

n_1 : Length of the data to be shifted

n_2 : Number of bits in a group

Explanation

1. This instruction divides the data in the **n_1** word devices starting from **D** into groups (**n_2** words in a group), and shifts these groups to the left. This instruction then shifts the data in the **n_2** word devices starting from **S** to the devices starting from **D** to fill the vacancy.
2. In general, the WSFLP pulse instruction is used.
3. The operand **n_1** must be between 1–512. The operand **n_2** must be between 1–**n_1**.

Example 1

1. When M0 switches from OFF to ON, the instruction divides the data in the sixteen word devices starting from D20 to D35 into groups (four words in a group), and shifts these groups to the left.
2. The shift of the data in the word devices to the left during a scan is shown below.

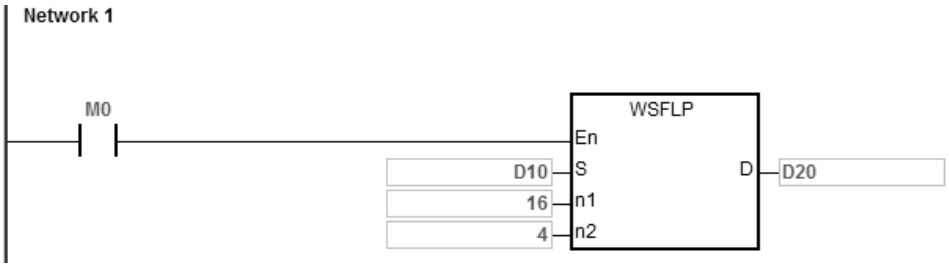
❶ D35-D32 → Being carried

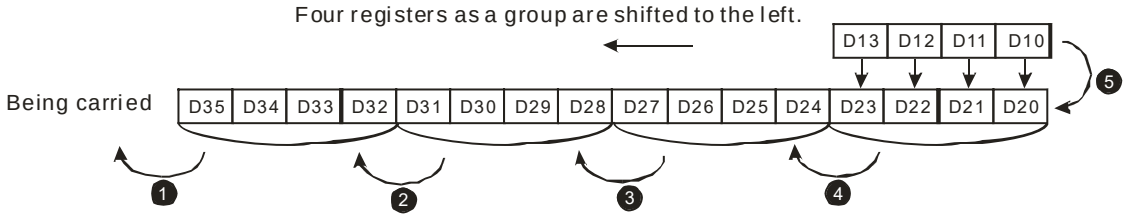
❷ D31-D28 → D35-D32

❸ D27-D24 → D31-D28

❹ D23-D20 → D27-D24

❺ D13-D10 → D23-D20





Example 2

1. When M0 switches from OFF to ON, the instruction divides the data in the sixteen word devices starting from D20 to D35 into groups (five words in a group), and shifts these groups to the left.

2. The shift of the data in the word devices to the left during a scan is shown below.

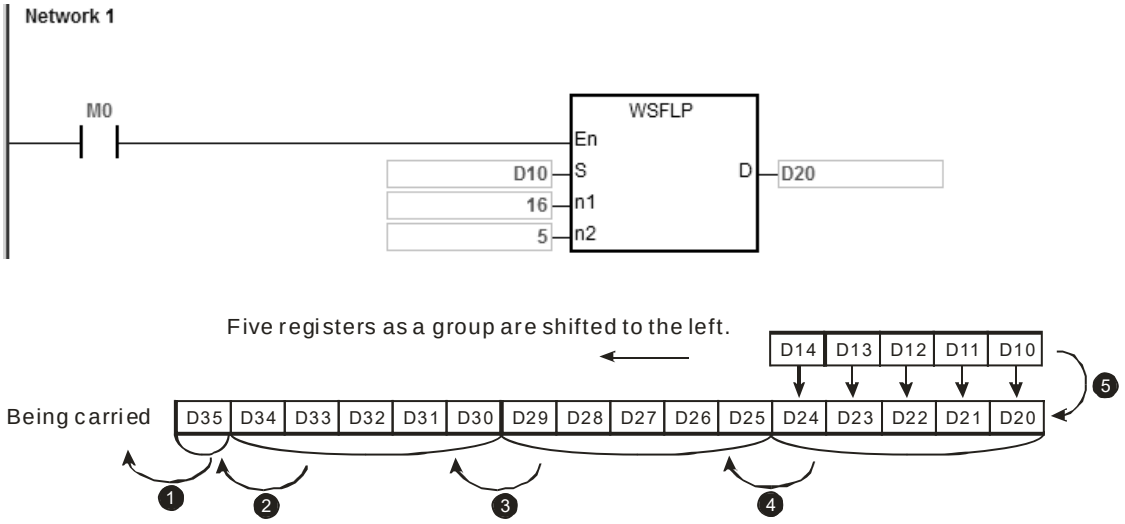
❶ D35 → Being carried

❷ D30 → D35

❸ D29-D25 → D34-D30

❹ D24-D20 → D29-D25

❺ D14-D10 → D24-D20



Additional remarks

1. If **S**+n_2-1 or **D**+n_1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If n_1 is not between 1–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If n_2 is not between 1–n_1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

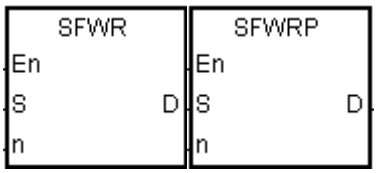
API	Instruction code			Operand	Function
1104		SFWR	P	**S, D, n**	Shifting the data and writing it into a word device

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		
D					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Device where the data is shifted

D : First device

n : Data length

Explanation

1. This instruction defines the data in the **n** word devices starting from the device specified by **D** as a first in-first out list type, and takes the device specified by **D** as a pointer. This instruction increments the value of the pointer by one, and writes the data in the device specified by **S** into the device specified by the pointer. When the value of the pointer is larger than or equal to **n**-1, the instruction stops writing data, and sets the carry flag SM602 is ON.

2. In general, the SFWRP pulse instruction is used.

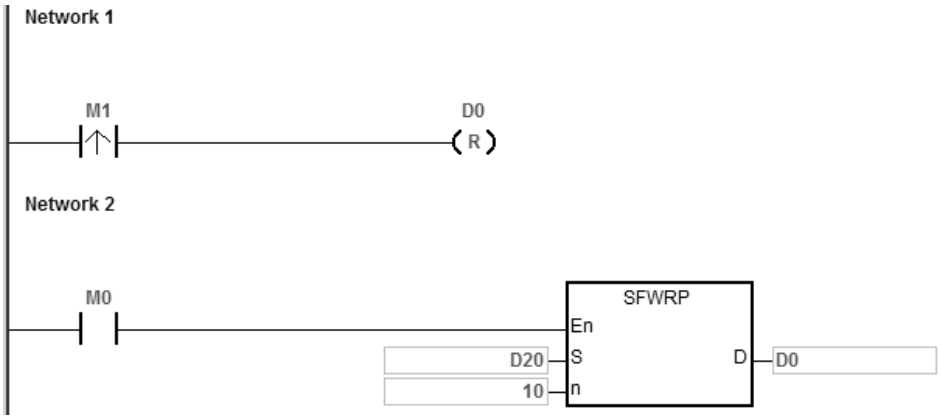
3. The operand **n** must be between 2–512.

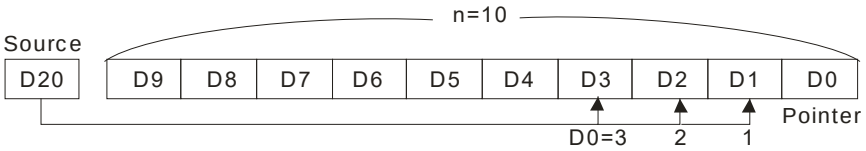
Example

1. The instruction clears the value of the pointer D0 to 0 first. When M0 switches from OFF to ON, the instruction writes the data in D20 into D1, and increments the value in D0 to 1. When M0 switches from OFF to ON again, the instruction writes the data in D20 to D2, and increments the value in D0 to 2.

2. The instruction shifts and writes the data in the word device as shown below.

   - The data in D20 is written into D1.
   - The value in D0 becomes 1.





Additional remarks

1. If the value in **D** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **D**+**n**-1 exceeds the device range, the instruction is not executed. SM0 is ON, and the error code in SR0 is 16#2003.
3. If **n** not between 2–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. You can use the SFWR instruction with the SFRD (API 1105) instruction to write and read the data.

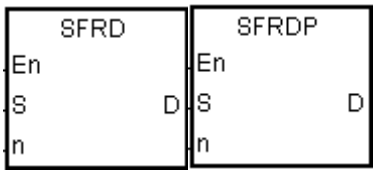
API	Instruction code			Operand	Function
1105		SFRD	P	**S, D, n**	Shifting the data and reading it from a word device

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●			○					
D					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device

D : Device where the data is shifted

n : Data length

Explanation

1. This instruction defines the data in the **n** word devices starting from the device specified by **S** as a first in-first out list type, and takes the device specified by **S** as a pointer. This instruction decrements the value in the device specified by **S** by one, writes the data in the device specified by **S**+**1** into the device specified by **D**, shifts the data in the devices specified by **S**+**n**-1–**S**+2 to the right, and leaves the data in the device specified by **S**+**n**-1 unchanged. When the value in the device specified by **S** is equal to 0, the instruction stops reading the data, and sets the zero flag SM600 is ON.

2. In general, the SFRDP pulse instruction is used.

3. The operand **n** must be between 2–512.

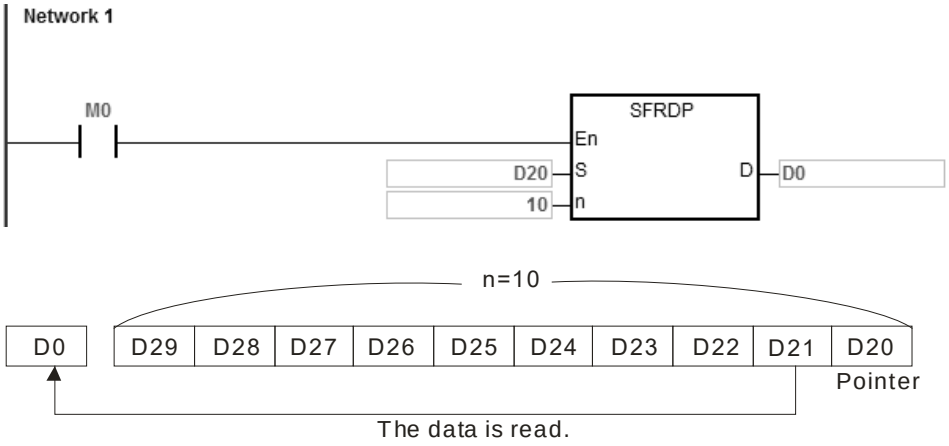
Example

1. When M0 switches from OFF to ON, the instruction writes the data in D21 into D0, shifts the data in D29–D22 to the right, leaves the data in D29 unchanged, and the decrements the value in D20 by one.

2. The data in the word device is shifted and read as shown below.

   - The data in D21 is read and shifted to D0.

- The data in D29-D22 is shifted to the right.
- The value in D20 decreases by one.



Additional remarks

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **n** not between 2–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. You can use the SFWR instruction with the SFRD instruction (API 1105) to write and read the data.

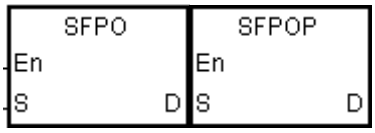
API	Instruction code			Operand	Function
1106		SFPO	P	**S, D**	Reading the latest data from a data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device

D : Device where the data is stored

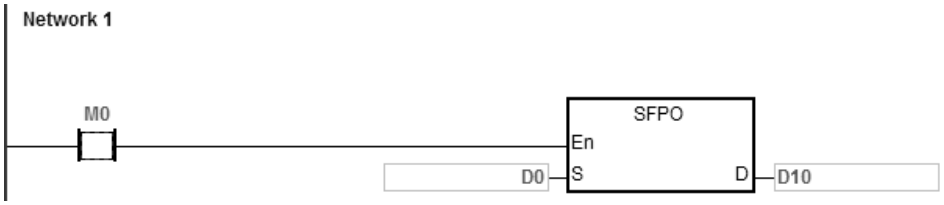
Explanation

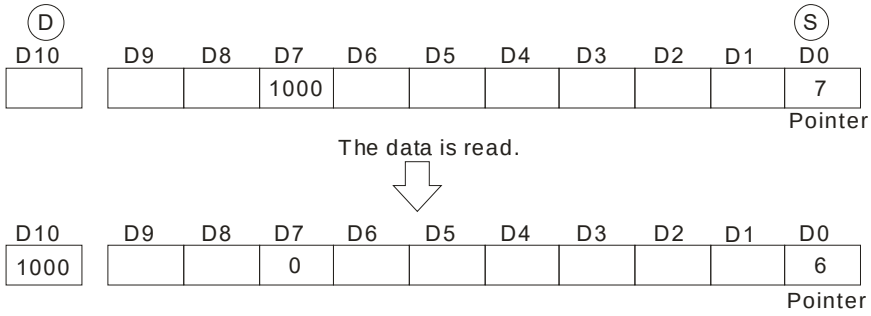
1. This instruction takes the device specified by **S** as a pointer. This instruction writes the data in the device specified by the value of the pointer into the device specified by **D** and clears it to 0, and decrements the value in the device specified by **S** by one. When the value in the device specified by **S** is equal to 0, the instruction stops reading the data, and sets the zero flag SM600 is ON.

2. In general, the SFPOP pulse instruction is used.

Example

When M0 is ON, the instruction writes the data in the device specified by D0 into D10. After the instruction shifts the data, the instruction clears the data in the device specified by D0 to 0, and increments the value in D0 by 1. If D0 is7 and M0 is ON. It moves the value (1000) in D7 to D10. D7 is clears to zero and D0 is 6.





Additional remarks

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **S**+(the value in **S**) exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

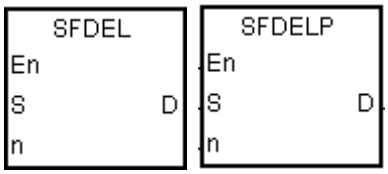
API	Instruction code			Operand	Function
1107		SFDEL	P	**S, D, n**	Deleting data from a data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D					●	●		●			○	○				
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device

D : Device where the data is stored

n : Device where the data is deleted

Explanation

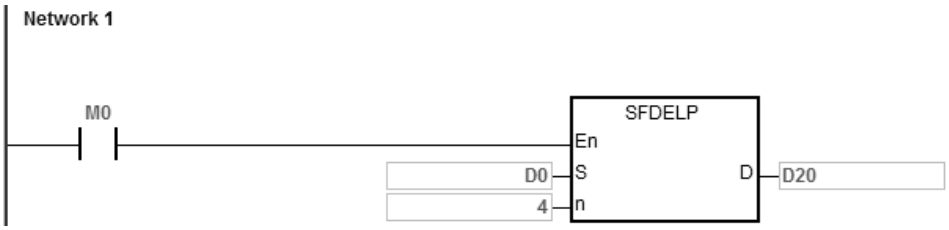
1. For this instruction, the length of the data is the value in the device specified by **S**, and the data itself is in the devices specified by **S**+1–**S**+(the value in **S**). This instruction stores the data in the device specified by **S**+**n** in **D** and deletes it, shifts the data in the devices specified by **S**+**n**+**1**–**S**+(The value in **S**) to the right, clears the data in the device specified by **S**+(the value in **S**) to 0, and decrements the value in the device specified by **S** by one. When the value in the device specified by **S** is equal to 0, the instruction stops deleting the data, and sets the zero flag SM600 is ON.

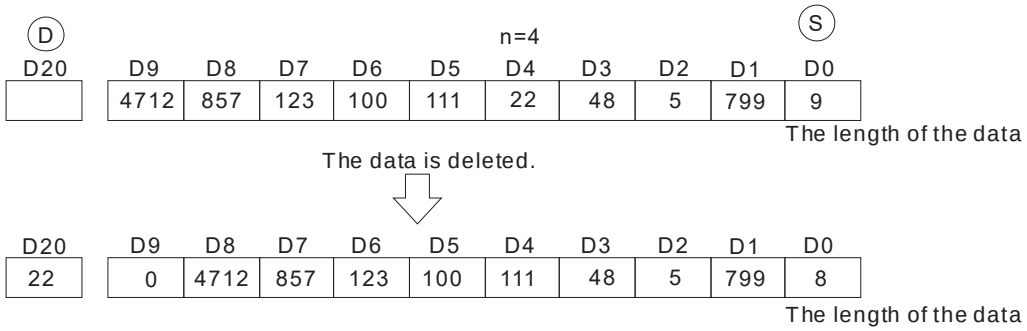
2. In general, the SFDELP pulse instruction is used.

3. The operand **n** must be between 1–32767.

Example

Suppose the value in D0 is 9, and **n** is 4. When M0 is ON, the instruction stores the data in D4 in D20. After the instruction deletes the data in D4, it shifts the data in D5–D9 to the right, and decrements the value in D0 by one.





Additional remarks

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **S**+**n** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **S**+(the value in **S**) exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If **n** is not between 0–**S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

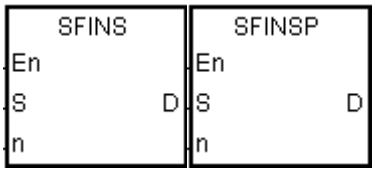
API	Instruction code			Operand	Function
1108		SFINS	P	**S, D, n**	Inserting data into a data list

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D					●	●		●	●		○	○	○	○		
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device

D : Data to be inserted

n : Device where the data is inserted

Explanation

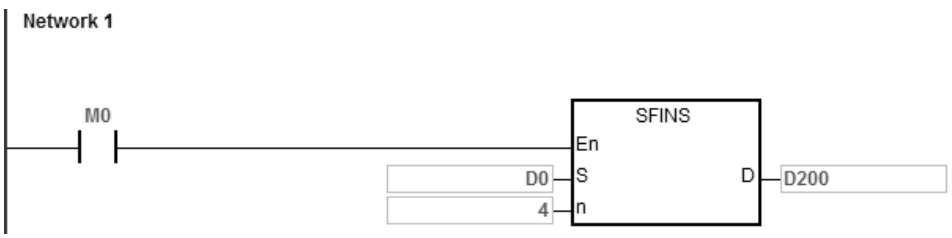
1. For this instruction, the length of the data is the value in the device specified by **S**, and the data itself is in the devices specified by **S**+1–**S**+(the value in **S**). This instruction inserts the data in **D** into **S**+**n**, shifts the original data in the devices specified by **S**+**n**–**S**+(the value in **S**) to the left, and increments the value in the device specified by **S** by one. When the value in the device specified by **S** is equal to 32767, the instruction stops writing the data, stops incrementing the value in the device specified by **S**, and sets the carry flag SM602 is ON.

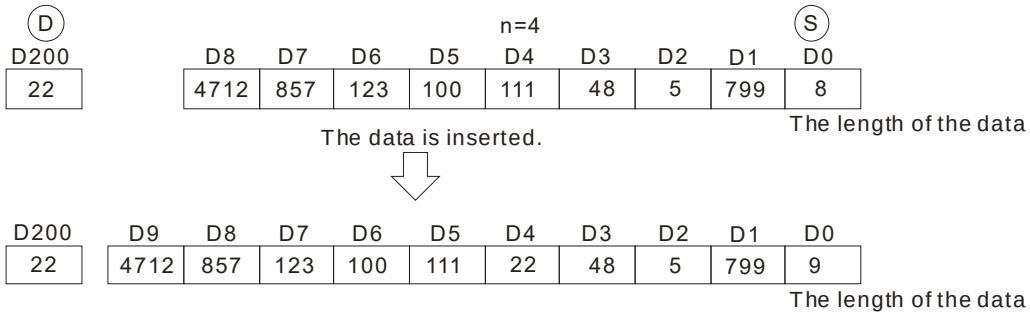
2. In general, the SFINSP pulse instruction is used.

3. The operand **n** must be between 1–32767.

Example

Suppose the value in D0 is 8, and **n** is 4. When M0 is ON, the instruction inserts the data in D200 into D4, shifts the original data in D4–D8 to D5–D9, and increments the value in D0 by one.





Additional remarks

1. If the value in **S** is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **S**+**n** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003
3. If **S**+(the value in **S**)+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If **n** is not between 0–**S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1109		MBS	P	**S, D, n**	Shifting matrix bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
D					●	●		●								
n					●	●		●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Matrix source

D : Operation result

n : Length of the array

Explanation

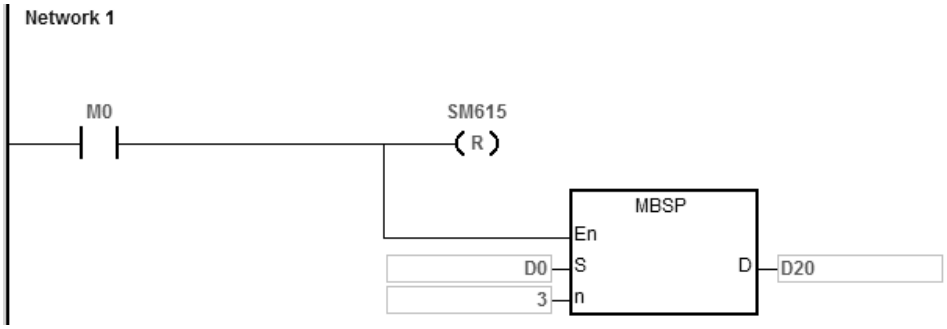
1. This instruction shifts the values of the **n** rows of bits in **S** to the right or to the left. When SM616 is OFF, the instruction shifts the values of the bits to the left. When SM616 is ON, the instruction shifts the values of the bits to the right. The instruction fills the vacancy (b0 when shifting to the left, and b16n-1 when shifting to the right) resulting from the shift with the state of the borrow flag SM615.The instruction transmits the value of the bit shifted last (from shifting to the left is b16n-1 and from shifting to the right is b0) to the carry flag SM614, and stores the operation result in **D**.

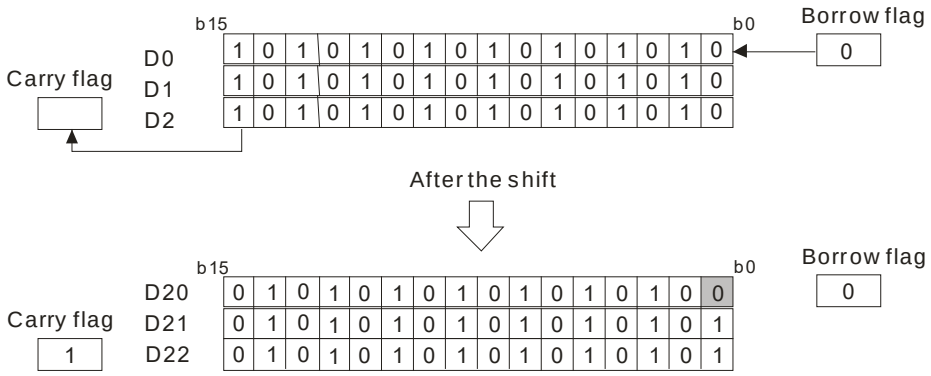
2. The operand **n** must be between 1–256.

3. In general, the MBSP pulse instruction is used.

Example 1

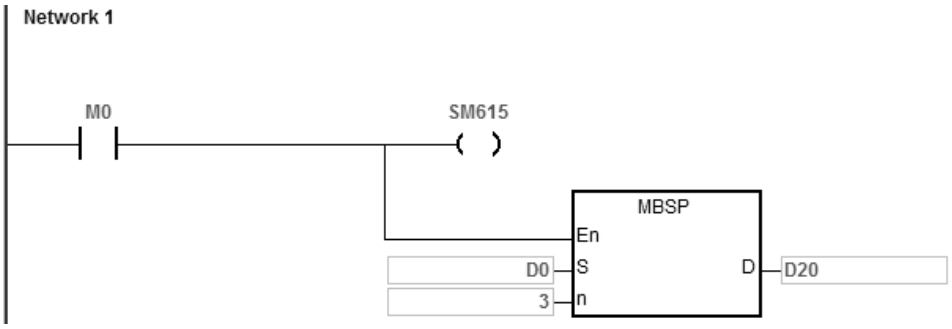
When M0 is ON, SM616 is OFF. The instructions shifts the values of the bits to the left. Suppose SM615 is OFF. After the instruction shifts the values of the bits in the 16-bit registers D0–D2 to the left, it stores the operation result in the 16-bit registers D20–D22, and SM614 is ON.

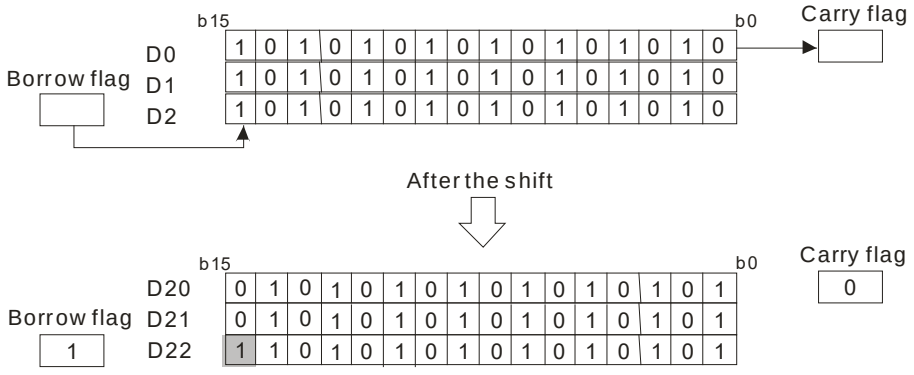




Example 2

When M0 is ON, SM616 is ON. The instruction shifts the values of the bits to the right. Suppose SM615 is ON. After the instruction shifts the values of the bits in the 16-bit registers D0–D2 to the right, it stores the operation result in the 16-bit registers D20–D22, and SM614 is OFF.





Additional remarks

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Instruction flags

SM614: Carry flag for the matrix rotation/shift/output.

SM615: Borrow flag for the matrix shift/output.

SM616: Direction flag for the matrix rotation/shift.

API	Instruction code			Operand	Function
1110	D	SFR	P	**D, n**	Shifting the values of the bits in registers by **n** bits to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

D : Device for the shift

n : Number of bits

Explanation

1. This instructions shift the values of the bits in **D** by **n** bits to the right. The 16-bit instruction fills the vacancies (b_{15}–b_{15-n+1}) resulting from the shift with 0 and the 32-bit instruction fills the vacancies (b_{31}–b_{31-n+1}) resulting from the shift with 0. The value of b_{n-1} is transmitted to SM602.

2. The operand **n** must be between 1–16 for 16-bit instructions and the operand **n** must be between 1–32 for 32-bit instructions.

3. In general, the SFRP/DSFRP pulse instruction is used more commonly.

4. Only the DSFR/DSFRP instruction (32-bit instructions) can use the 32-bit counter (high counter) but not the device E.

Example

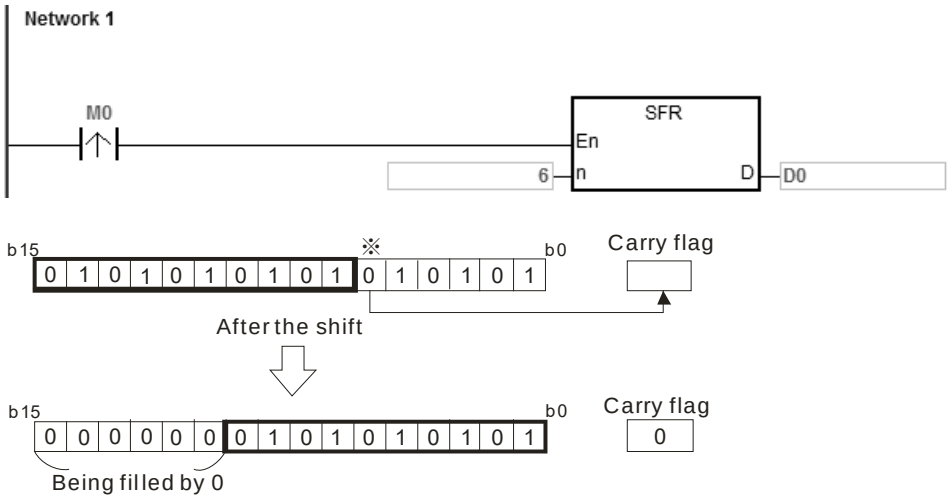
When M0 is ON, the instruction shifts the values of b0–b15 in D0 by 6 bits to the right, and transmits the value of b5 to SM602. The instruction clears the values of b10–b15 to zero after the shift.

The shift of the values of the bits to the right during a scan is shown below.

❶ b5-b0 → Being carried (The value of b5 is transmitted to SM602.)

❷ b15-b6 → b9-b0

❸ 0 → b15-b10



Additional remarks

If **n** is not between 0–16 (for 16-bit instructions) or between 0-32 (for 32-bit instructions), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1111	D	SFL	P	**D, n**	Shifting the values of the bits in registers by **n** bits to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

D : Device for the shift

n : Number of bits

Explanation

1. This instructions shift the values of the bits in **D** by **n** bits to the left. The instructions fill the vacancies (b_0–b_{n-1}) resulting from the shift with 0 and transmit the value of b_{16-n} (for 16-bit instruction) or b_{32-n} (for 32-bit instruction) to SM602.

2. The operand **n** must be between 1–16 for 16-bit instructions and the operand **n** must be between 1–32 for 32-bit instructions.

3. In general, the SFRP/DSFRP pulse instruction is used more commonly.

4. Only the DSFR/DSFRP instruction (32-bit instructions) can use the 32-bit counter (high counter) but not the device E.

Example

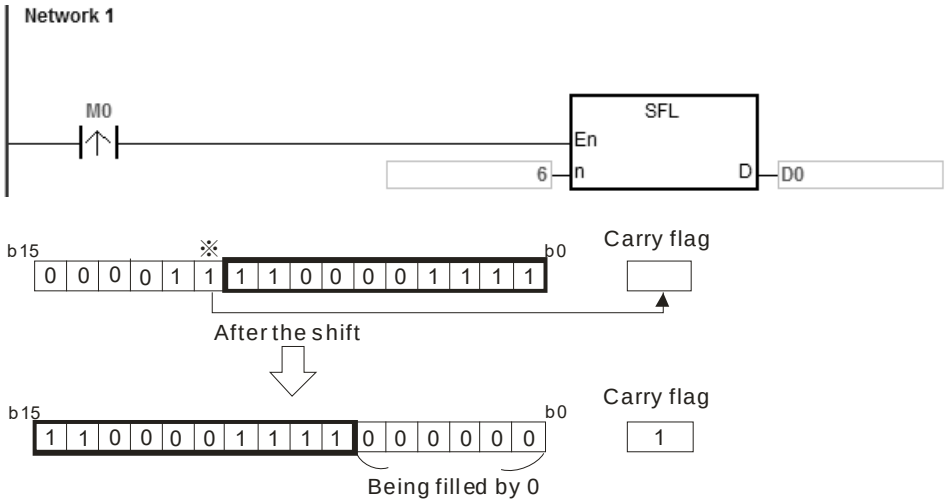
When M0 is ON, the instruction shifts the values of b0–b15 in D0 by 6 bits to the right, and transmits the value of b10 to SM602. The instruction fills the values of b0–b5 with zeros after the shift.

The shift of the values of the bits to the left during a scan shown below.

❶ b15-b10 → Being carried (The value of b10 is transmitted to SM602.

❷ b9-b0 → b15-b6

❸ 0 → b5-b0



Additional remarks

If **n** is not between 0–16 (for 16-bit instructions) or between 0-32 (for 32-bit instructions), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1112		BSFR	P	**D, n**	Shifting the states of **n** bit devices by one bit to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D			●	●				●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

D : First device for the shift

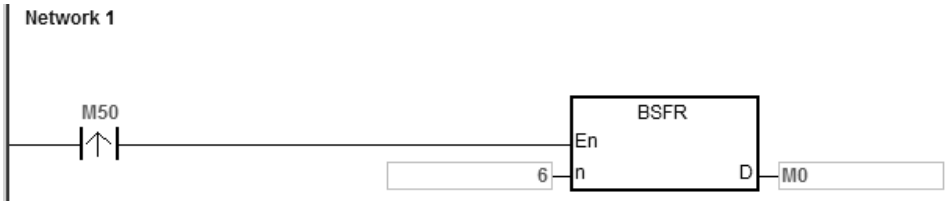
n : Data length

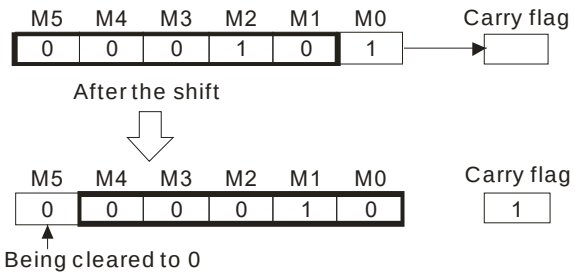
Explanation

1. This instruction shifts the states of the **n** bit devices starting from **D** by one bit to the right. The instruction clears state of **D**+**n**-1 to 0, and transmits the state of **D** to the carry flag SM602.
2. In general, the BSFRP pulse instruction is used.
3. The operand **n** must be between 1–1024.

Example

When M50 is ON, the instruction shifts the states of M0–M5 by one bit to the right, clears the state of M5 to zero, and transmits the state of M0 to the carry flag SM602.





Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** not between 1–1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

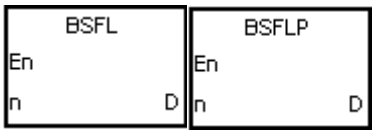
API	Instruction code			Operand	Function
1113		BSFL	P	**D, n**	Shifting the states of **n** bit devices by one bit to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D		●	●	●				●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



D : First device for the shift

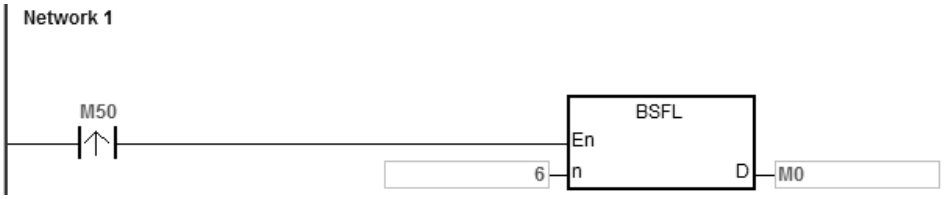
n : Data length

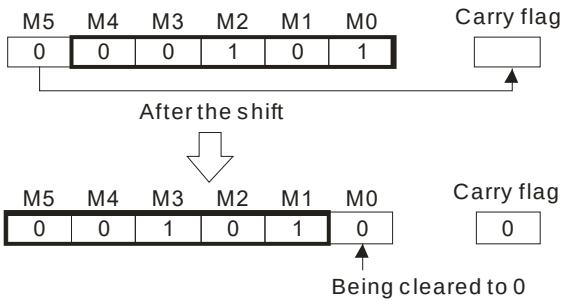
Explanation

1. This instruction shifts the states of the **n** bit devices starting from **D** by one bit to the left. The instruction clears the state of **D** to 0, and transmits the state of **D**+**n**-1 to the carry flag SM602.
2. In general, the BSFLP pulse instruction is used.
3. The operand **n** must be between 1–1024.

Example

When M50 is ON, the instruction shifts the states of M0–M5 by one bit to the left, clears the state of M0 to 0, and transmits the state of M5 to the carry flag SM602.





Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** not between 1–1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

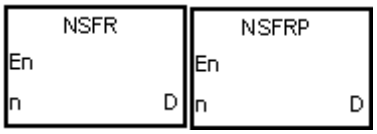
API	Instruction code			Operand	Function
1114		NSFR	P	**D, n**	Shifting **n** registers to the right

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



D : First device for the shift

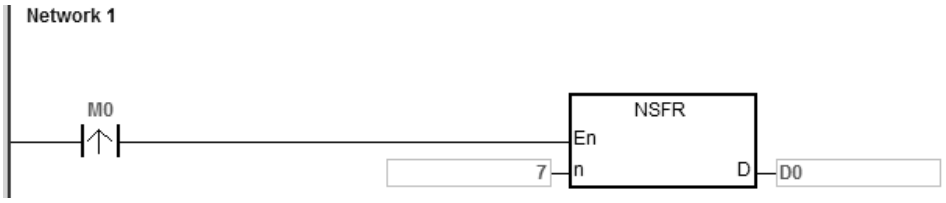
n : Data length

Explanation

1. This instruction shifts the data in the **n** registers starting from **D** to the right, and clears the data in **D**+**n**-1 to 0.
2. In general, the NSFRP pulse instruction is used.
3. The operand **n** must be between 1–512.

Example

When M0 is ON, the instruction shifts the data in D1–D6 to the right, and clears the data in D6 to 0.



D6	D5	D4	D3	D2	D1	D0
30	2235	9578	754	28	423	11

After the shift

D6	D5	D4	D3	D2	D1	D0
0	30	2235	9578	754	28	423

Being cleared to 0

Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1115		NSFL	P	**D, n**	Shifting **n** registers to the left

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

D : First device for the shift

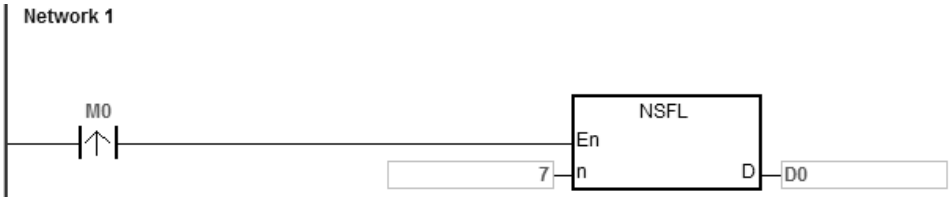
n : Data length

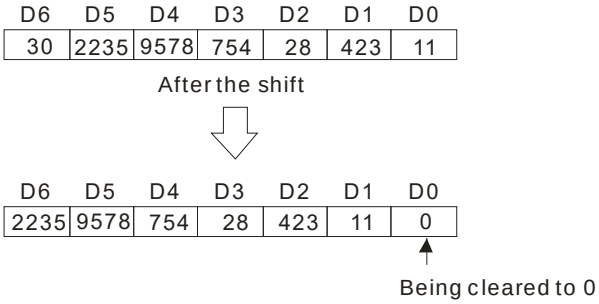
Explanation

1. This instruction shifts the data in the **n** registers starting from **D** to the left, and clears the data in **D** to 0.
2. In general, the NSFLP pulse instruction is used.
3. The operand **n** must be between 1–512.

Example

When M0 is ON, the instruction shifts the data in D0–D5 to the left, and clears the data in D0 to 0.





Additional remarks

1. If **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

6.13 Data Processing Instructions

6.13.1 List of Data Processing Instructions

The following table lists the Data Processing instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1200	SER	DSER	✓	Searching the data
1201	SUM	DSUM	✓	Finding the number of bits whose states are ON
1202	DECO	–	✓	Decoding bits
1203	ENCO	–	✓	Encoding bits
1204	SEGD	–	✓	Seven-segment decoding
1205	SORT	DSORT	✓	Sorting data
1206	ZRST	–	✓	Resetting a zone
1207	BON	DBON	✓	Checking the state of a bit
1208	MEAN	DMEAN	✓	Finding the mean
1209	CCD	–	✓	Finding the sum check
1210	ABS	DABS	✓	Finding the absolute value
1211	MINV	–	✓	Inverting matrix bits
1212	MBRD	–	✓	Reading a matrix bit
1213	MBWR	–	✓	Writing a matrix bit
1214	MBC	–	✓	Counting the bits with the value zero or one
1215	DIS	–	✓	Disuniting 16-bit data
1216	UNI	–	✓	Uniting 16-bit data
1217	WSUM	DWSUM	✓	Finding the sum
1221	LIMIT	DLIMIT	✓	Confining a value within bounds
1222	BAND	DBAND	✓	Deadband control
1223	ZONE	DZONE	✓	Controlling the zone
1224	–	FMEAN	✓	Finding the mean of floating point numbers
1225	–	FSUM	✓	Finding the sum of floating point numbers
1226	–	DTM	✓	Data conversion and move

6.13.2 Explanation of Data Processing Instructions

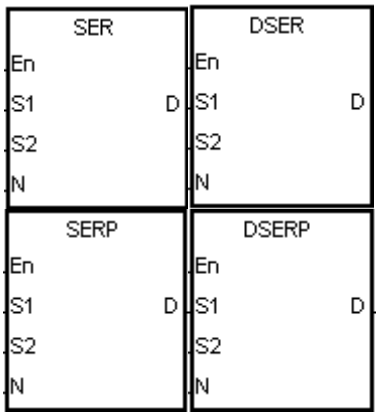
API	Instruction code			Operand	Function
1200	D	SER	P	**S_1, S_2, D, n**	Searching the data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							
S_2					●	●	●	●	●		○	○	○	○		
D					●	●	●	●								
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : First device for the comparison

S_2 : Compared data

D : First device where the comparison result is stored

n : Data length

Explanation

1. This instruction compares **n** signed decimal values in the registers starting from the register specified by $\mathbf{S_1}$ with the signed decimal value in the register specified by $\mathbf{S_2}$, and stores the comparison results in the registers **D**–**D**+4.

Device	Description
D	Number of equal values
D+1	Data number of the first equal value
D+2	Data number of the last equal value
D+3	Data number of the minimum value
D+4	Data number of the maximum value

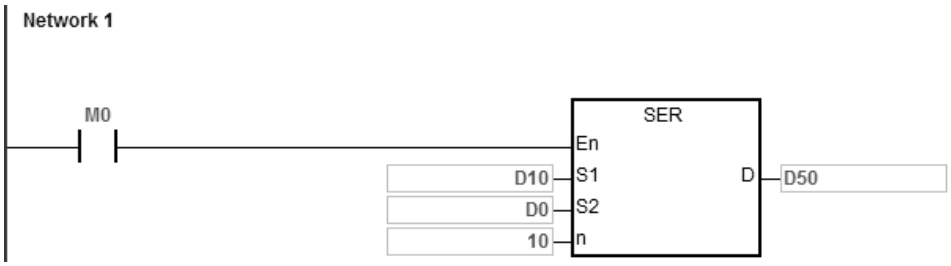
2. The operand **n** must be between 1–256.

3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

1. When M0 is ON, the instruction compares the values in D10–D19 with the value in D0, and stores the comparison results in D50–D54. When the equal value does not exist in D10–D19, the values in D50–D52 are 0.

2. The instruction stores the data number of the minimum value in D53, and stores the data number of the maximum value in D54. If there is more than one minimum value or maximum value, the instruction stores the data number that is bigger.



S_1	Value	Compared data	Data number	Result
D10	88	S_2 D0=100	0	
D11	100		1	Equal
D12	110		2	
D13	150		3	
D14	100		4	Equal
D15	300		5	
D16	100		6	Equal
D17	5		7	Minimum
D18	100		8	Equal
D19	500		9	Maximum

D	Value	Description
D50	4	Number of equal values
D51	1	Data number of the first equal value
D52	8	Data number of the last equal value
D53	7	Data number of the minimum value
D54	9	Data number of the maximum value

Additional remarks

1. If **S_1+n**-1 or **D**+4 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. For 16-bit instructions, if the value in **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. For 32-bit instructions, if the value in **n** is not between 1–128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. For 16-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [5] of WORD/INT.
5. For 32-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [5] of DWORD/DINT.

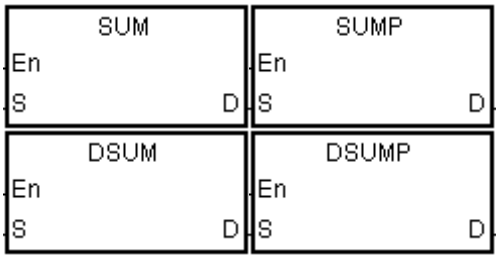
API	Instruction code			Operand	Function
1201	D	SUM	P	**S, D**	Finding the number of bits whose states are ON

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

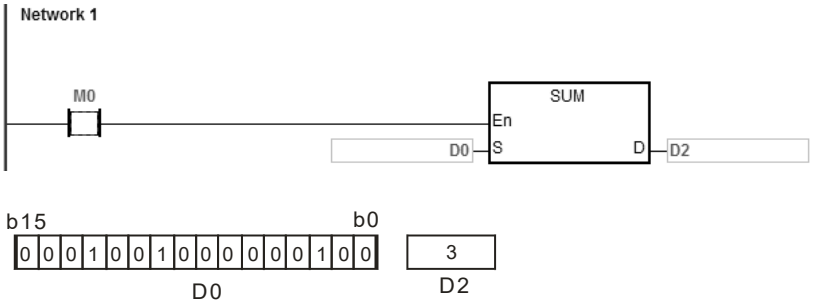
D : Destination device

Explanation

1. This instruction finds the number of bits in **S** whose values are ON and stores the number of ON bits in **D**.
2. When the values of all the bits in the source device specified by **S** are 0, the zero flag SM600 is ON.
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

When M0 is ON, the instruction stores the number of bits whose values are one in D0 in D2.



Additional remarks

If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

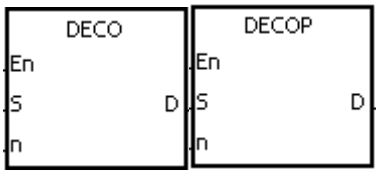
API	Instruction code			Operand	Function
1202		DECO	P	**S, D, n**	Decoding bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●		●	●	○	○	○				
D		●	●	●	●	●		●				○				
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●			●	●							
D	●	●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Source device

D : Device where the decoded values are stored

n : Number of bits whose values are decoded

Explanation

1. This instruction decodes the values of the lower **n** bits in the source device specified by **S** as the values of the lower 2^n bits in **D**.

2. The instruction decodes the values of the consecutive **n** bits in the source device specified by **S** as the values of the lower 2^n bits in **D**.

3. When the source device specified by **S** is a timer or counter, the instruction treats the device as a word device.

4. When **D** is a bit device, **n** between1–8. When **n** is 8, the instruction decodes the values of the eight bits as the values of the 256 bits. Please note that the devices in which the decoded values are stored cannot be used repeatedly.

5. When **D** is a word device, **n** between 1–4. When **n** is 4, the instruction decodes the values of the four bits as the values of the 16 bits.

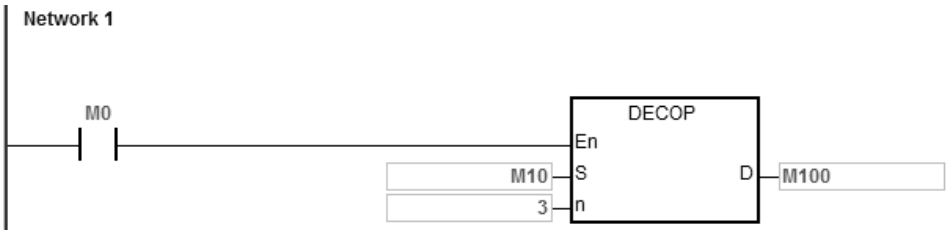
6. In general, the DECOP pulse instruction is used.

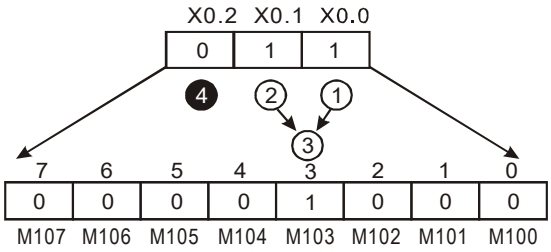
Example 1

1. When M0 switches from OFF to ON, the DECO instruction decodes the values of the 3 bits in M10–M12 as the values of the 8 bits in M100–M107.

2. The instruction adds the values of the 3 bits in M10–M12 to get the value 3. The instruction sets the third bit in M10–M1007, that is, the bit in M103 to 1.

3. After the DECO instruction is executed and M0 switches to OFF, the values of the eight bits in M100–M107 are unchanged.



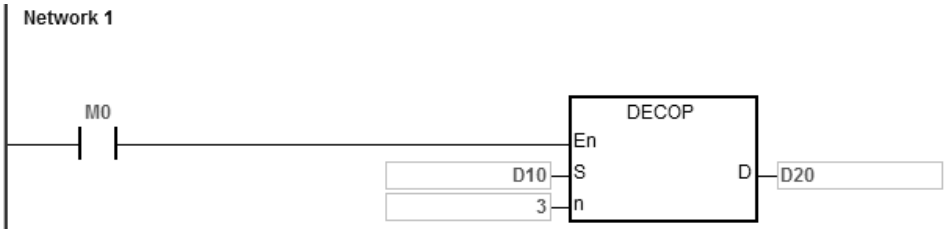


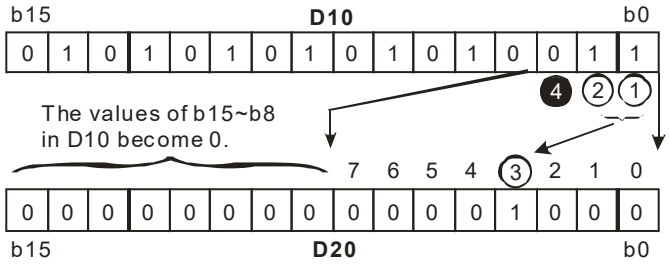
Example 2

1. When M0 switches from OFF to ON, the DECO instruction decodes the values of b2–b0 in D10 as the values of b7–b0 in D20, and sets the values of b15–b8 in D10 to 0.

2. The instruction decodes the values of the lower three bits in D10 as the values of the lower eight bits in D20. The instruction sets the values of the higher eight bits to 0.

3. After the DECO instruction is executed and M0 switches to OFF, the data in D20 is unchanged.





Additional remarks

1. If **D** is a bit device and if **n** not between 1–8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **D** is a word device and if **n** is not between 1–4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S** is a bit device and if **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If **D** is a bit device and if **D**+(2^**n**)-**1** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

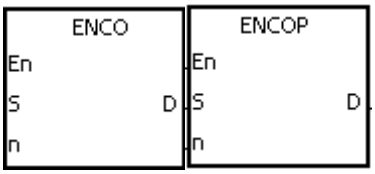
API	Instruction code			Operand	Function
1203		ENCO	P	**S, D, n**	Encoding bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●	●	●	●	●		●	●	○	○	○				
D					●	●		●			○	○				
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●	●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Source device

D : Device where the encoded values are stored

n : Number of bits whose values are encoded

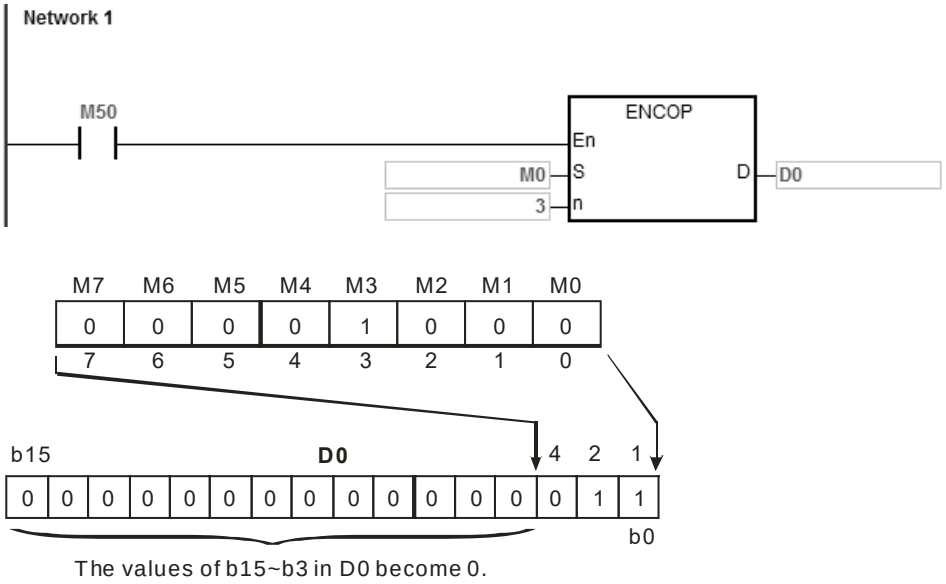
Explanation

1. When **S** is a word device, this instruction encodes the values of the lower 2^n bits in the source device specified by **S** as the values of the lower **n** bits in **D**.
2. When **S** is a bit device, the instruction processes the higher bit with the value **S**+(**n**-1) from the lower 2^n bits and stores the result in **D**.
3. When the source device specified by **S** is a timer or counter, the instruction treats the device as a word device.
4. When **S** is a bit device, **n** is between 1–8. When **n** is 8, the instruction encodes the values of the 256 bits as the values of the eight bits.
5. When **S** is a word device, **n** is between 1–4. When **n** is 4, the instruction encodes the values of the 16 bits as the values of the four bits.
6. In general, the ENCOP pulse instruction is used.

Example 1

1. When M50 switches from OFF to ON, the ENCO instruction encodes the values of the 8 bits in M0–M7 as the values of the lower 3 bits in D0, and sets the values of b15–b3 in D0 to 0.

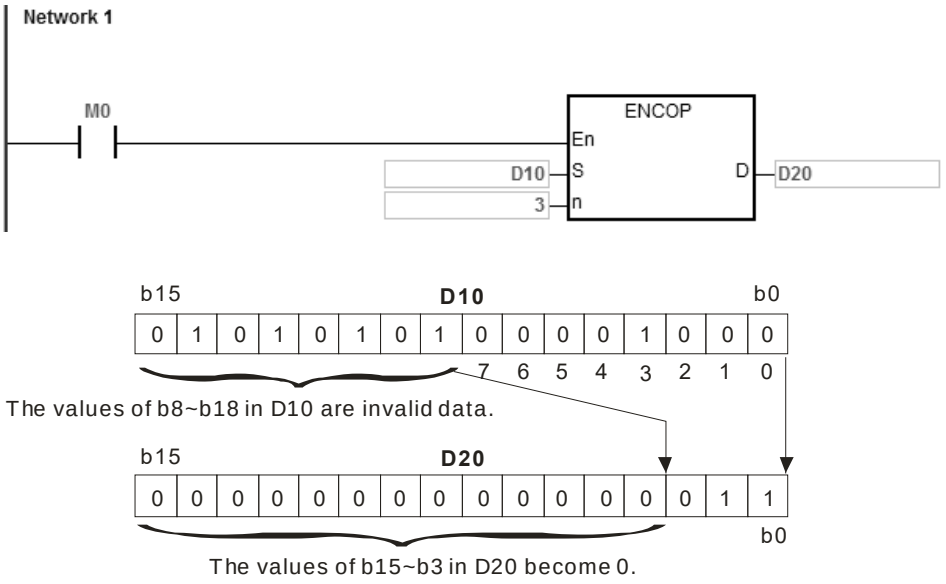
2. After the ENCO instruction is executed and M50 switches to OFF, the data in **D** is unchanged.



Example 2

1. When M0 switches from OFF to ON, the ENCO instruction encodes the values of b0–b7 in D10 as the values of b2–b0 in D20, and sets the values of b15–b3 in D20 to zero. The values of b8–b18 in D10 are invalid data.

2. After the ENCO instruction is executed and M0 switches OFF, the data in **D** is unchanged.



Additional remarks

1. If there is no bit whose value is one in the source device specified by **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S** is a bit device and if **n** is not between 1–8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S** is a word device and if **n** is not between 1–4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If **S** is a bit device and if **S**+(2^**n**)-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. If **D** is a bit device and if **D**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1204		SEGD	P	**S, D**	Seven-segment decoding

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Source device

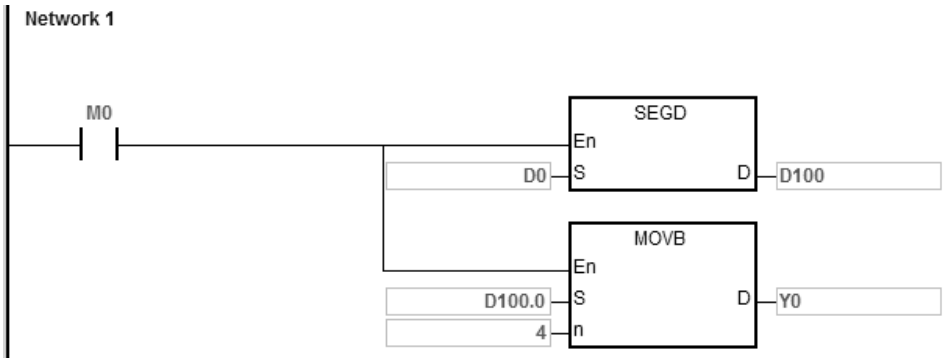
D : Device where the seven-segment data is stored

Explanation

The instruction decodes the values of the lower four bits (b0–b3) in the source device specified by **S** as the seven-segment data stored in **D**.

Example

When M0 is ON, the instruction decodes the values of b0–b3 in D0 as the seven-segment data and store it in D100. After that move the data to Y0–Y3. If the data in the source device exceeds four bits, the instruction decodes the values of the lower four bits.



The following table shows the relation between the seven-segment data and the bit pattern of source data.

Hex	Bit pattern	Assignment of segments	Segment state							Display
			B0(a)	B1(b)	B2(c)	B3(d)	B4(e)	B5(f)	B6(g)	
0	0000		ON	ON	ON	ON	ON	ON	OFF	
1	0001		OFF	ON	ON	OFF	OFF	OFF	OFF	
2	0010		ON	ON	OFF	ON	ON	OFF	ON	
3	0011		ON	ON	ON	ON	OFF	OFF	ON	
4	0100		OFF	ON	ON	OFF	OFF	ON	ON	
5	0101		ON	OFF	ON	ON	OFF	ON	ON	
6	0110		ON	OFF	ON	ON	ON	ON	ON	
7	0111		ON	ON	ON	OFF	OFF	ON	OFF	
8	1000		ON	ON	ON	ON	ON	ON	ON	
9	1001		ON	ON	ON	ON	OFF	ON	ON	
A	1010		ON	ON	ON	OFF	ON	ON	ON	
B	1011		OFF	OFF	ON	ON	ON	ON	ON	
C	1100		ON	OFF	OFF	ON	ON	ON	OFF	
D	1101		OFF	ON	ON	ON	ON	OFF	ON	
E	1110		ON	OFF	OFF	ON	ON	ON	ON	
F	1111		ON	OFF	OFF	OFF	ON	ON	ON	

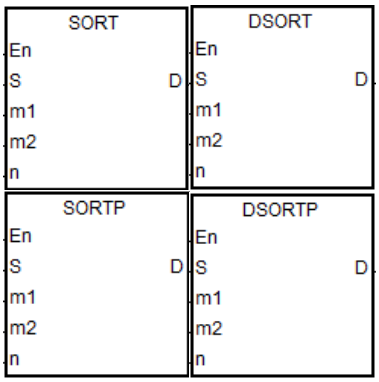
API	Instruction code			Operand	Function
1205	D	SORT	P	**S, m_1, m_2, D, n**	Sorting data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●								
m_1								●					○	○		
m_2								●					○	○		
D								●								
n								●					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : First device where the original data is stored

m_1 : Number of rows of data

m_2 : Number of columns of data

D : First device where the sorted data is stored

n : Reference value for sorting the data

Explanation

1. This instruction stores the data to be sorted in the **m1×m2** registers starting from the register specified by **D**. If **S** and **D** specify the same register, the sorted data is the same as the original data in the register specified by **S**.

2. The operand **m_1** must be between 1–32. The operand **m_2** must be between 1–6. The operand **n** must be between 1–**m_2**.

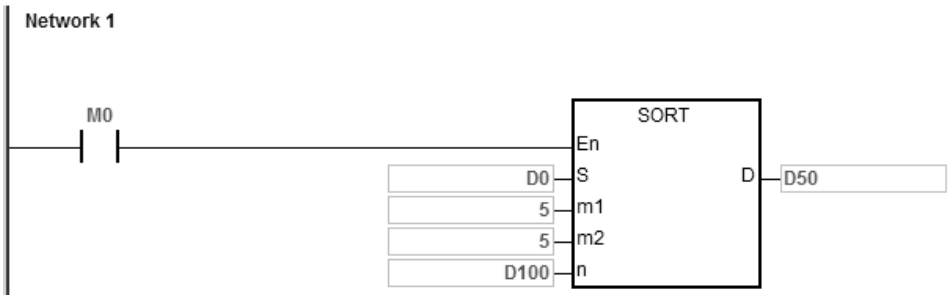
3. When SM604 is OFF, the instruction sorts the data in ascending order. When SM604 is ON, the instruction sorts the data in descending order.

4. It is suggested that you use the SORTP or DSORTP pulse type instruction instead of sorting repeatedly.

5. Only the 32-bit instruction can use the 32-bit counter, but not the device E.

Example

1. Suppose SM604 is OFF. When M0 switches from OFF to ON, the instruction sorts the data in ascending order.



2. The data which to be sorted is as in the following table.

m_2 columns of data

Column

Column / Row	1	2	3	4	5
	Student number	**Chinese**	**English**	**Math**	**Physics**
1	(D0) 1	(D5) 90	(D10) 75	(D15) 66	(D20) 79
2	(D1) 2	(D6) 55	(D11) 65	(D16) 54	(D21) 63
3	(D2) 3	(D7) 80	(D12) 98	(D17) 89	(D22) 90
4	(D3) 4	(D8) 70	(D13) 60	(D18) 99	(D23) 50
5	(D4) 5	(D9) 95	(D14) 79	(D19) 75	(D24) 69

m_1 rows of data

3. When the value in D100 is 3, the data is sorted as in the following table.

m_2 columns of data

Column

Column / Row	1	2	3	4	5
	Student number	Chinese	English	Math	Physics
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 1	(D57) 90	(D62) 75	(D67) 66	(D72) 79
4	(D53) 5	(D58) 95	(D63) 79	(D68) 75	(D73) 69
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

m_1 rows of data

4. When the value in D100 is 5, the data is as in the following table.

m_2 columns of data

Column

Column / Row	1	2	3	4	5
	Student number	Chinese	English	Math	Physics
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 5	(D57) 95	(D62) 79	(D67) 75	(D72) 69
4	(D53) 1	(D58) 90	(D63) 75	(D68) 66	(D73) 79
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

m_1 rows of data

Additional remarks

1. If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If $\mathbf{m_1}$, $\mathbf{m_2}$, or **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

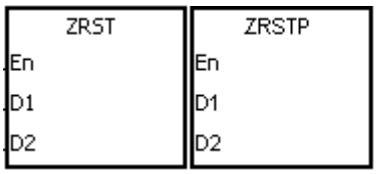
API	Instruction code			Operand	Function
1206		ZRST	P	D_1, D_2	Resetting a zone

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D_1		●	●	●	●	●	●	●		○	○	○				
D_2		●	●	●	●	●	●	●		○	○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D_1	●	●			●	●							
D_2	●	●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



D_1 : First device to be reset

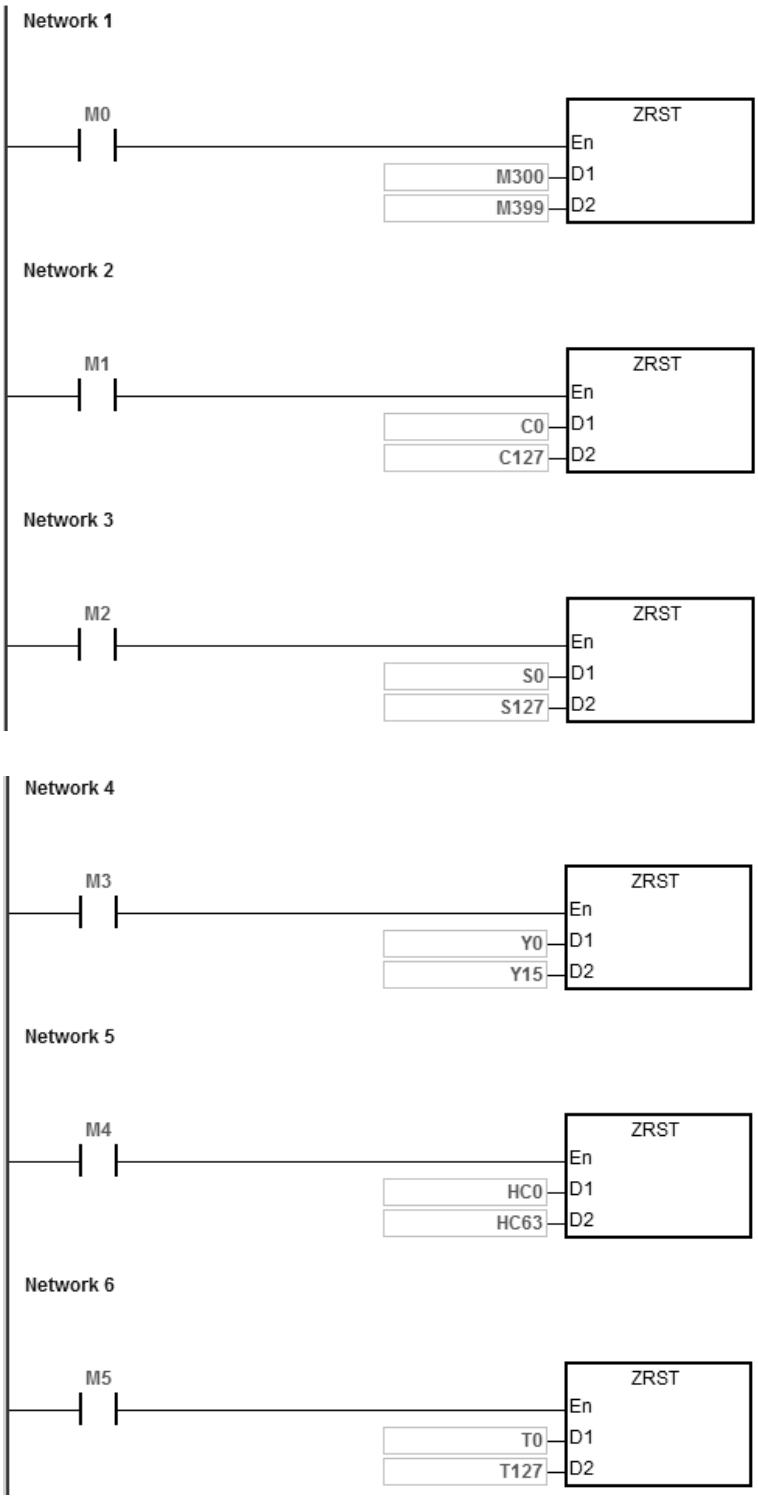
D_2 : Last device to be reset

Explanation

1. This instruction clears the values in **D_1**–**D_2**. The device type for **D_1**–**D_2** should be the same for this instruction.
2. When the device number of **D_1** is larger than the device number of **D_2**, the instruction resets only **D_2**.
3. The ZRST instruction can use the 32-bit counter.

Example

1. When M0 is ON, the instruction resets the auxiliary relays M300–M399 to OFF.
2. When M1 is ON, the instruction resets the 16-bit counters C0–C127. The values of C0–C127 are cleared to zero, and the contact and the coil are reset to OFF.
3. When M2 is ON, the instruction resets the stepping relays S0–S127 to OFF.
4. When M3 is ON, the instruction resets the output relays Y0–Y15 to OFF.
5. WhenM4 is ON, the instruction resets the 32-bit counters HC0–HC63. The values of HC0–HC63 are cleared to zero, and the contact and the coil are reset to OFF.
6. When M5 is ON, the instruction resets the timers T0–T127. The values of T0–T127 are cleared to 0. and the contact and the coil are reset to OFF.



Additional remarks

1. If D_1 and D_2 are different types of devices, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.

2. If D_1 and D_2 contain different data formats, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.

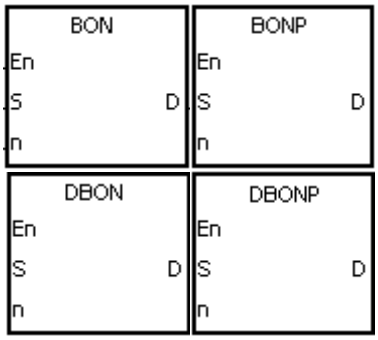
API	Instruction code			Operand	Function
1207	D	BON	P	**S, D, n**	Checking the state of a bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D		●	●	●				●		○						
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D	●												
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source device

D : Device where the check result is stored

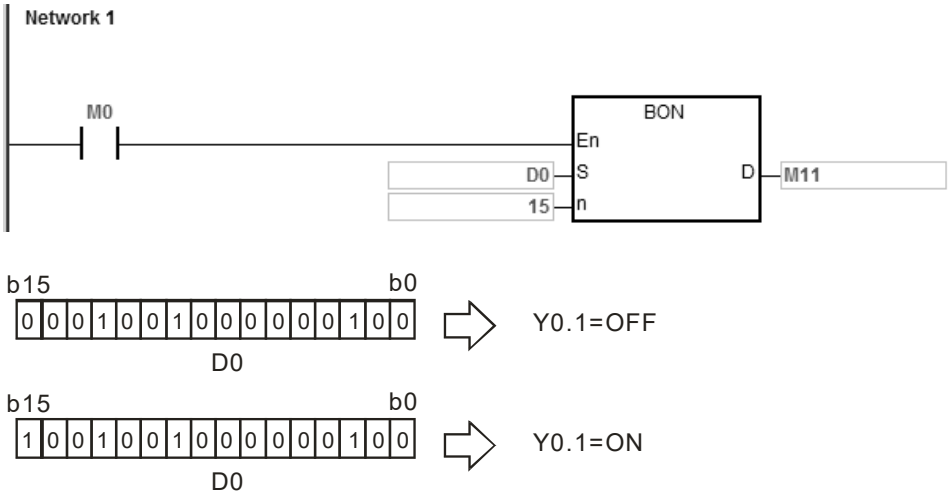
n : Bit whose state is checked

Explanation

1. This instruction checks the state of the **n**th bit in **S**, and stores the result in **D**.
2. The operand **n** used in the 16-bit instruction must be between 0–15. For 32-bit instructions, **n** must be between 0–31.
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

1. When M0 is ON, if the value of the 15th bit in D0 is one, M11 is ON. M11 is OFF if the value of the 15th bit in D0 is 0.
2. When M0 switches to OFF, the state of M1 remains the same as before M0 switches to OFF.



Additional remarks

If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

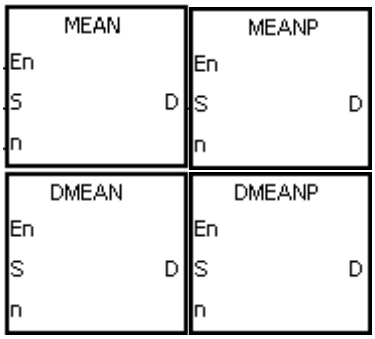
API	Instruction code			Operand	Function
1208	D	MEAN	P	**S, D, n**	Finding the mean

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●							
D					●	●	●	●			○	○				
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : First device

D : Device where the mean is stored

n : Number of devices

Explanation

1. This instruction adds up the values in the **n** devices starting from the device specified by **S**, and the stores the mean of the sum in **D**.

2. If a remainder appears in the calculation, the instruction discards it.

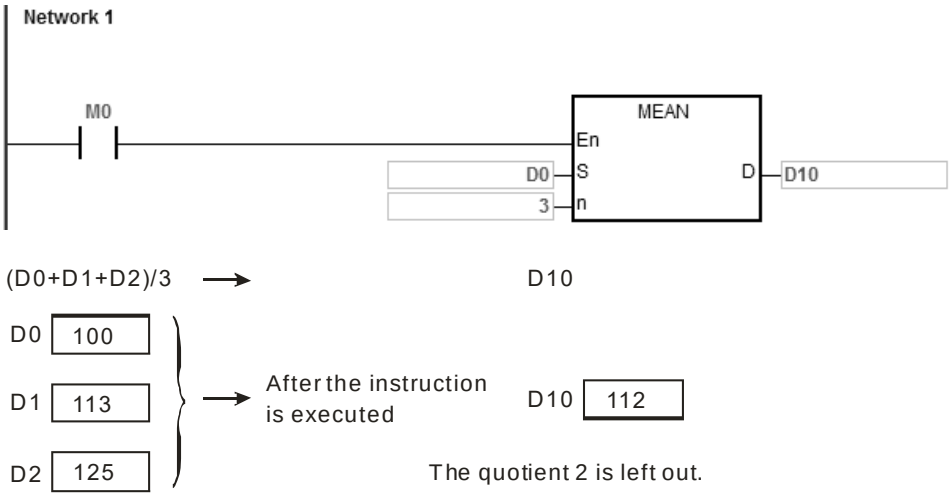
3. For 16-bit instructions, **n** must be between 1–256.

4. For 32-bit instructions, **n** must be between 1–128.

5. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

When M0 is ON, the instruction adds up the values in the three registers starting from D0. The instruction divides the sum by 3. The instruction stores the quotient in D10, and leaves out the remainder.



Additional remarks

1. For 16-bit instructions, if **n** not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. For 32-bit instruction, if **n** is not between1–128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

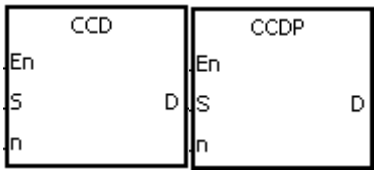
API	Instruction code			Operand	Function
1209		CCD	P	**S, D, n**	Finding the sum check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
D					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : First device

D : Device where the sum is stored

n : Number of pieces of data

Explanation

1. Communication protocols use the sum check function to compare checksums on the same data on different occasions or on different representations of the data to verify data integrity.

2. When SM606 is OFF, the instruction uses the 16-bit conversion mode. The instruction adds up **n** pieces of data in the registers starting from the register specified by **S** (eight bits as a group). The instruction stores the sum in the register specified by **D**, and stores the values of the parity bits in **D**+1.

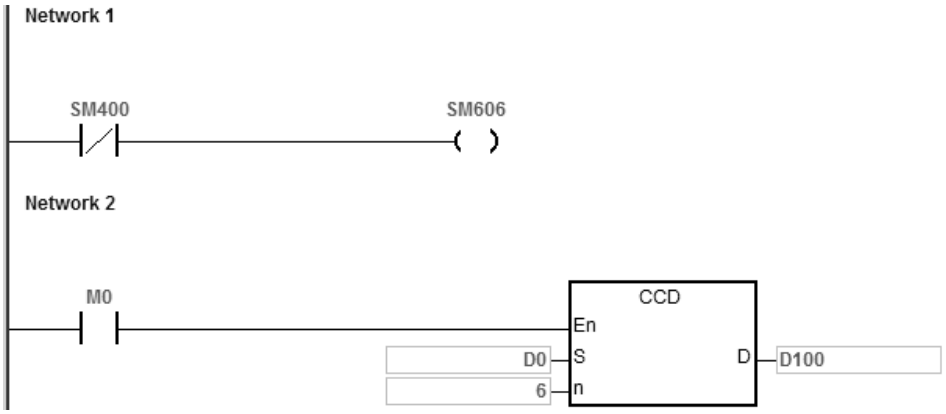
3. When SM606 is ON, the instruction uses the 8-bit conversion mode. The instruction adds up the **n** pieces of data in the registers starting from the register specified by **S** (eight bits in a group, and only low eight bits are valid). The instruction stores the sum in the register specified by **D**, and stores the values of the parity bits in **D**+1.

4. The operand **n** must be between 1–256.

Example 1

1. When SM606 is OFF, the instruction uses the 16-bit conversion mode.

2. When M0 is ON, the instruction adds up the six pieces of data in D0–D2 (eight bits in a group). The instruction stores the sum in D100, and stores the values of the parity bits in D101.



S	Data
D0 Low	100 = 0 1 1 0 0 1 0 0
D0 High	111 = 0 1 1 0 1 1 1 ①
D1 Low	120 = 0 1 1 1 1 0 0 0
D1 High	202 = 1 1 0 0 1 0 1 0
D2 Low	123 = 0 1 1 1 1 0 1 ①
D2 High	211 = 1 1 0 1 0 0 1 ①
D100	867
D101	0 0 0 1 0 0 0 ①

Sum

The parity bit is set to 1 if the number of ones is odd.
The parity bit is set to 0 if the number of ones is even.

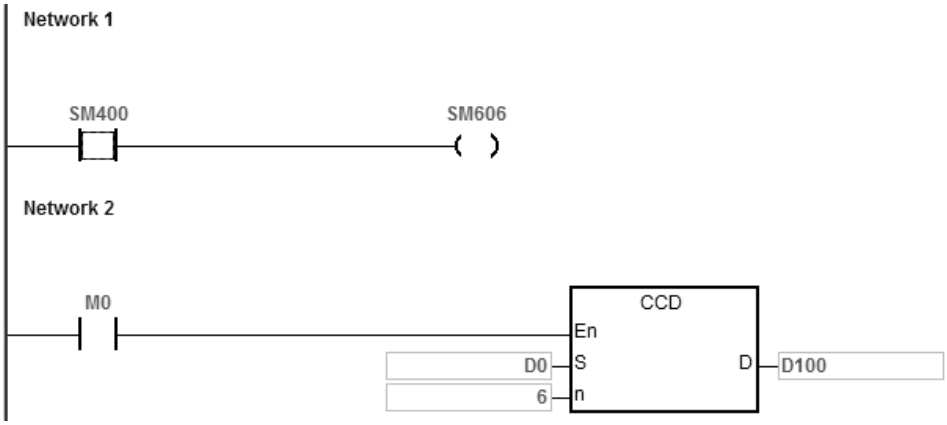
D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Parity bits

Example 2

1. When SM606 is ON, the instruction uses the 8-bit conversion mode.

2. When M0 is ON, the instruction adds up the six pieces of data in D0–D5 (eight bits in a group). The instruction stores the sum in D100, and stores the values of the parity bits in D101.



S	Data
D0 Low	100 = 0 1 1 0 0 1 0 0
D1 Low	111 = 0 1 1 0 1 1 1 ①
D2 Low	120 = 0 1 1 1 1 0 0 0
D3 Low	202 = 1 1 0 0 1 0 1 0
D4 Low	123 = 0 1 1 1 1 0 1 ①
D5 Low	211 = 1 1 0 1 0 0 1 ①
D100	867
D101	0 0 0 1 0 0 0 ①

Sum

The parity bit is set to "1" if the number of ones is odd.
The parity bit is set to "0" if the number of ones is even.

D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Parity bits

Additional remarks

1. Suppose SM606 is ON. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. Suppose SM606 is OFF. If **S**+**n**/2-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. If you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

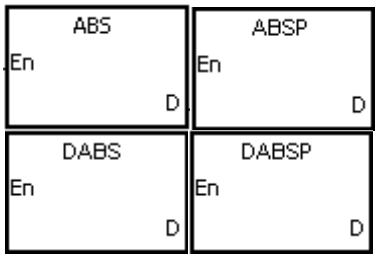
API	Instruction code			Operand	Function
1210	D	ABS	P	**D**	Finding the absolute value

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



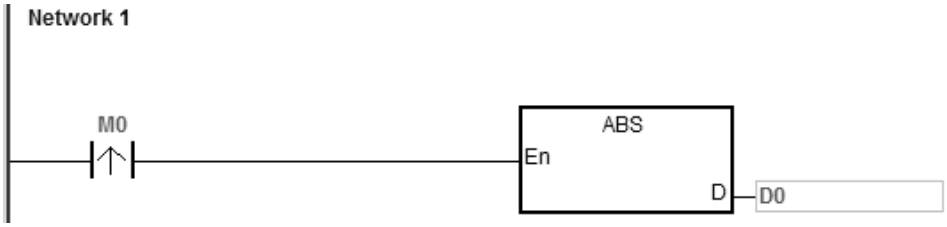
D : Device holding the original value

Explanation

1. This instruction finds the absolute value of the value in the device specified by **D**.
2. In general, the ABSP pulse instruction is used.
3. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

Suppose the value in D0 is originally -1234. When M0 switches from OFF to ON, the instruction finds the absolute value of -1234 in D0. That is, the value in D0 becomes 1234 after the instruction is executed.



API	Instruction code			Operand	Function
1211		MINV	P	**S, D, n**	Inverting matrix bits

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
D					●	●		●								
n					●	●		●	●		○		○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Matrix source

D : Operation result

n : Length of the array

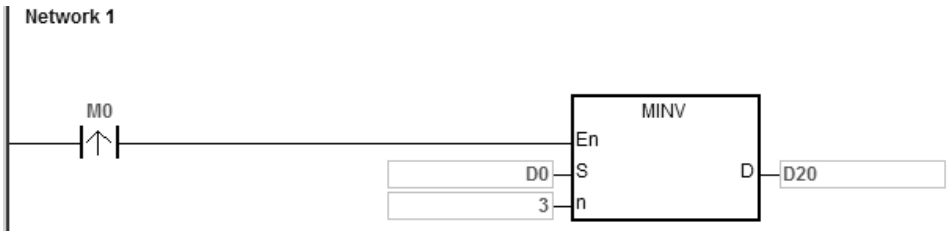
Explanation

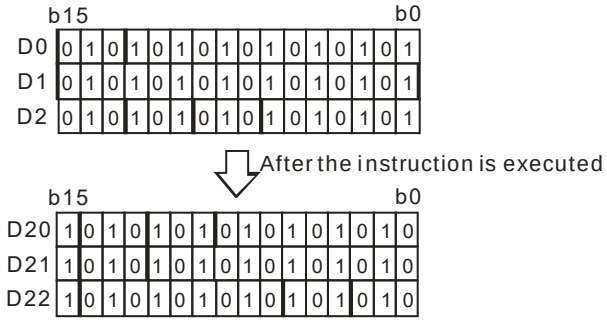
1. This instruction inverts the bits in the **n** devices starting from the device specified by **S**, and stores the inversion result in **D**.

2. The operand **n** must be between 1–256.

Example

When M0 is ON, the instruction inverts the bits in the three 16-bit registers D0–D2, and stores the inversion result in the 16-bit registers D20–D22.





Additional remarks

1. If **S**+**n**-1 or **D**+**n**-1 exceeds the device range, the instruction is not execute, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1212		MBRD	P	**S, n, D**	Reading a matrix bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
n					●	●		●	●		○		○	○		
D					●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Matrix source

n : Length of the array

D : Pointer

Explanation

1. This instruction checks the state of SM613. If SM613 is ON, the instruction clears the value of the pointer **D** to zero. The instruction reads the value of the bit specified by the value of the pointer **D** into SM614, and then checks the state of SM612. If SM612 is ON, the instruction increments the value of the pointer **D** by adding one.

2. When the instruction reads the value of the last bit, SM608 is ON, and the instruction stores the bit number in the pointer **D**.

3. The operand **n** must be between 1–256.

4. You specify the value of the pointer. The values are between 0–16**n**–1, and correspond to the range between b0–b16**n**–1. If the value of the pointer exceeds the range, SM611 is set to one, and the instruction is not executed.

Example

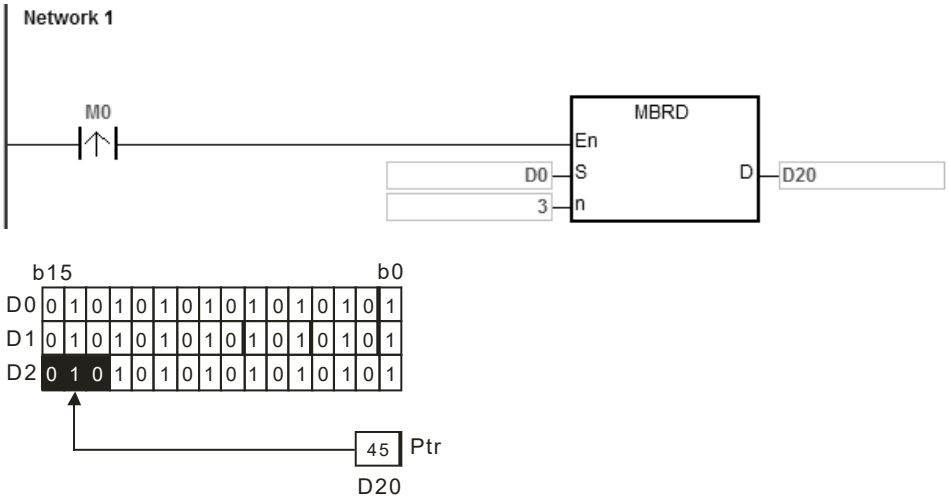
1. Suppose SM613 is OFF and SM612 is ON when M0 switches from OFF to ON.

2. Suppose the current value in D20 is 45. When M0 is switched from OFF to ON three times, the instruction gives the following execution results.

   ❶ The value in D20 is 46, SM614 is OFF, and SM608 is OFF.

   ❷ The value in D20 is 47, SM614 is ON, and SM608 is OFF.

   ❸ The value in D20 is 47, SM614 is OFF, and SM608 is ON.



Additional remarks

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Instruction flags:

   SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.

   SM611: Matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

   SM612: Matrix pointer increasing flag. The current value of the pointer increases by one.

   SM613: Matrix pointer clearing flag. The current value of the pointer is cleared to zero.

   SM614: Carry flag for the matrix rotation/shift/output.

API	Instruction code			Operand	Function
1213		MBWR	P	**S, n, D**	Writing a matrix bit

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
n					●	●		●	●		○		○	○		
D					●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Matrix source

n : Length of the array

D : Pointer

Explanation

1. This instruction checks the state of SM613. If SM613 is ON, the instruction clears the value of the pointer **D** to 0. The instruction writes the state of SM615 into the bit specified by the value of the pointer **D** and then checks the state of SM612. If SM612 is ON, the instruction increments the value in the pointer **D** by one.

2. When the instruction writes the state of SM615 into the last bit, sets SM608 is ON, and records the bit number in the pointer **D**. If value of the pointer **D** exceeds the range, SM611 is ON.

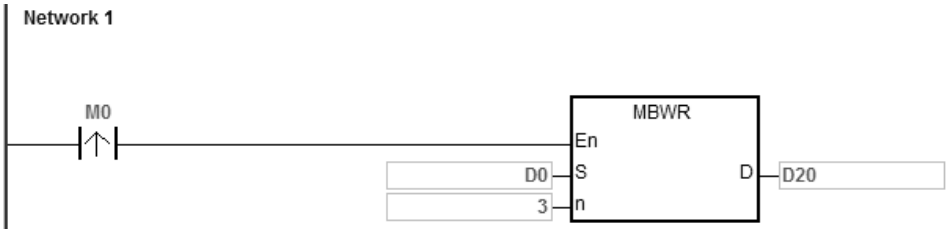
3. The operand **n** must be between 1–256.

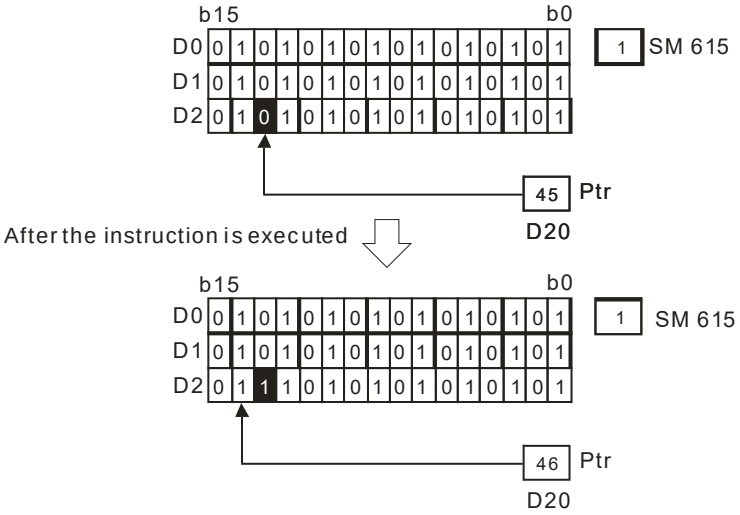
4. You specify the value of the pointer. The values are between 0–16**n**–1, and correspond to the range between b0–b16**n**–1. If the value of the pointer exceeds the range, SM611 is set to one, and the instruction is not executed.

Example

1. Suppose SM613 is OFF and SM612 is ON when M0 switches from OFF to ON.

2. Suppose the current value in D20 is 45. When M0 switches from OFF to ON one time, the instruction gives the execution result shown below. When the value in D20 is 45, SM615 is OFF, and SM608 is OFF.





Additional remarks

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Instruction flags:

   SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.

   SM611: Matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

   SM612: Matrix pointer increasing flag. The current value of the pointer increases by one.

   SM613: Matrix pointer clearing flag. The current value of the pointer is cleared to zero.

   SM615: Borrow flag for the matrix shift/output.

API	Instruction code			Operand	Function
1214		MBC	P	**S, n, D**	Counting the bits with the value zero or one

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
n					●	●		●	●		○		○	○		
D					●	●		●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Matrix source

n : Length of the array

D : Operation result

Explanation

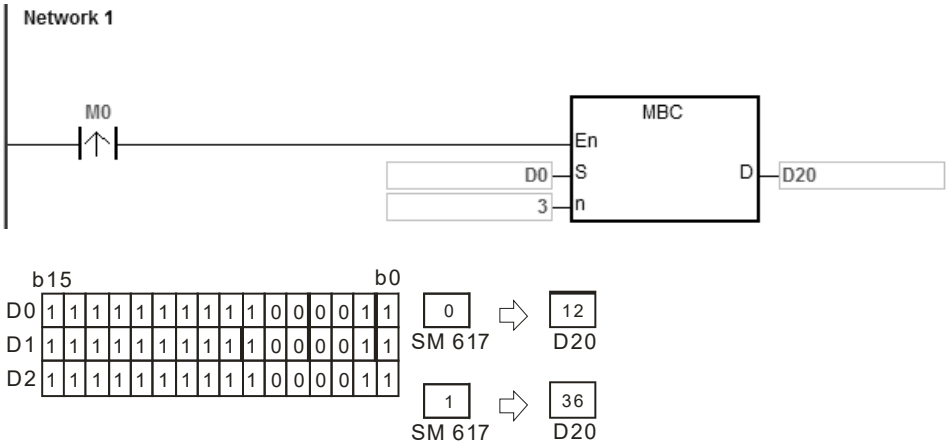
1. This instruction counts the bits with the value one or zero in the **n** devices starting from the device specified by **S**. The instruction stores the operation result in **D**.

2. When SM617 is ON, the instruction counts the bits with the value one. When SM617 is OFF, the instruction counts the bits with the value 0. When the operation result is 0, SM618 is ON.

3. The value in **n** must be between 1–256.

Example

Suppose SM617 is ON. When M0 is ON, the instruction counts the bits with the value one, and stores the operation result in D20. Suppose SM617 is OFF. When M0 is ON, the instruction counts the bits with the value zero, and stores the operation result in D20.



Additional remarks

1. If **S**+**n**-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. Instruction flags:

   SM617: The bits with the value zero or one are counted.

   SM618: ON when the matrix counting result is 0.

API	Instruction code			Operand	Function
1215		DIS	P	**S, n, D**	Disuniting 16-bit data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○				
n					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

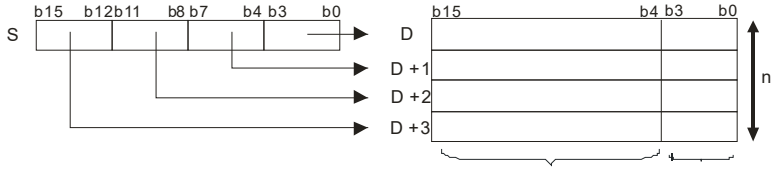
S : Data source

n : Number of devices

D : Operation result

Explanation

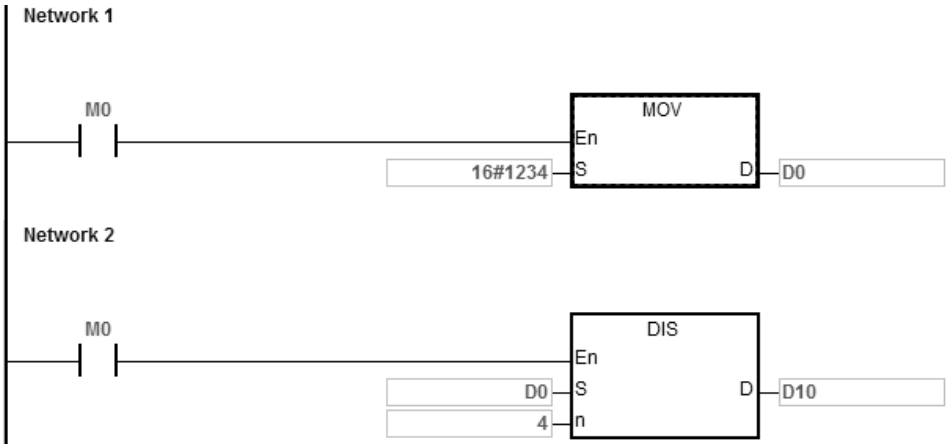
1. This instruction divides the 16-bit value in the register specified by **S** into four groups (four bits in a group), and stores these groups in the low four bits in every register (the registers range from **D** to **D**+(**n**-1)).

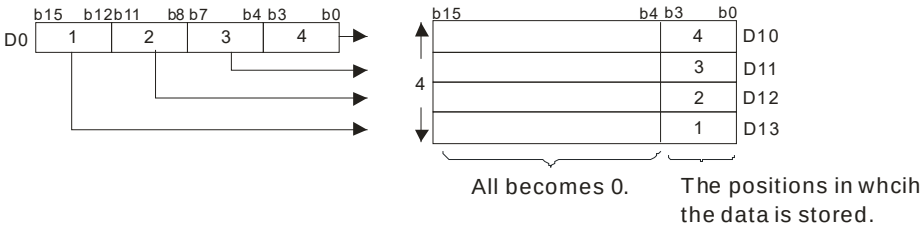


2. The value in **n** must be between 1–4.

Example

Suppose the value in D0 is 16#1234. When M0 is enabled, the instruction divides the value in D0 into four groups (four bits in a group), and stores these groups in the low four bits in every register (the registers range from D10 to D13.).





Additional remarks

1. If **D**–**D**+(**n**-1) exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** is not between 1–4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

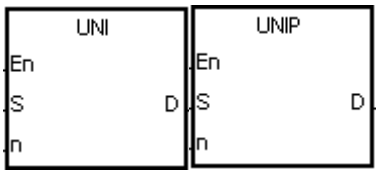
API	Instruction code			Operand	Function
1216		UNI	P	**S, n, D**	Uniting 16-bit data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							
n					●	●		●	●		○	○	○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



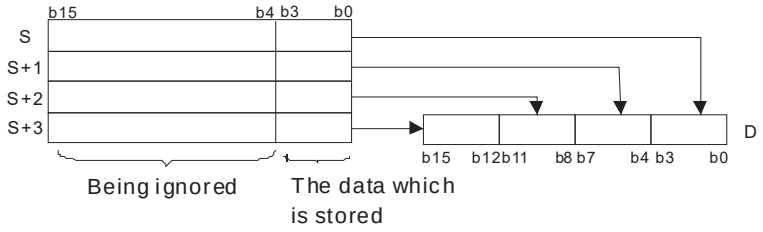
S : Data source

n : Data length

D : Operation result

Explanation

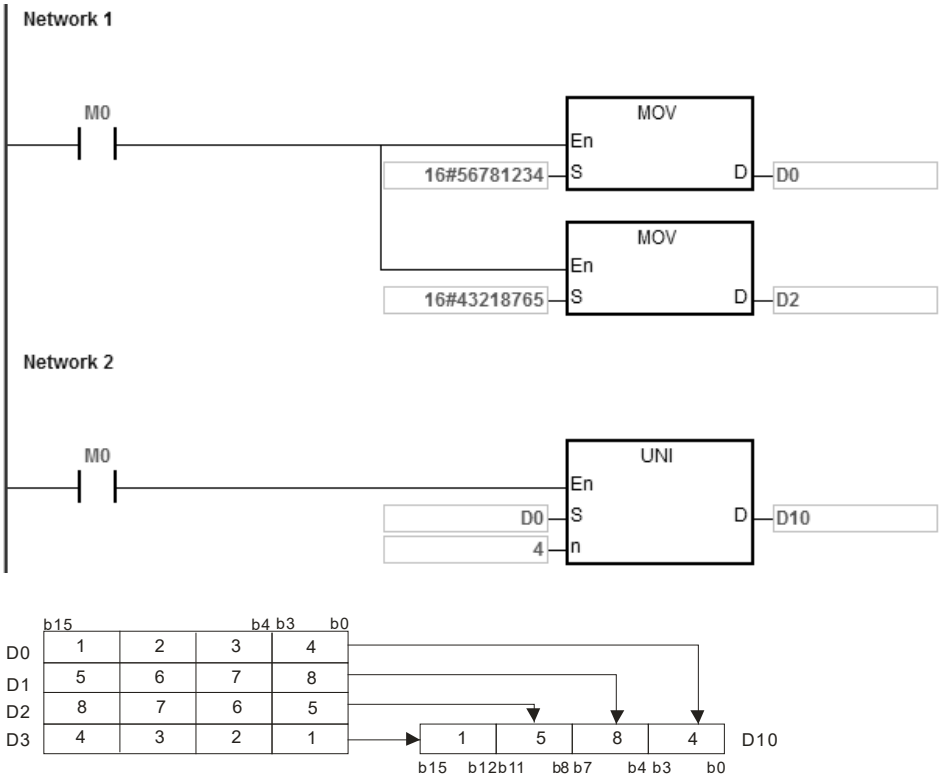
1. This instruction divides the 16-bit values in the registers specified by **S**–**S**+(**n**-1) into groups (four bits in a group), and stores every group that is in b0–b3 in the register specified by **D** (b0–b15).



2. The value in **n** must be between 1–4.

Example

Suppose the values in D0–D3 are 16#1234, 16#5678, 16#8765, and 16#4321 respectively. When M0 is enabled, the UNI instruction divides the values in D0–D3 into groups (four bits in a group), and stores every group in b0–b3 in D10(b0–b15).



Additional remarks

1. If **S** to **S**+(**n**-1) exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **n** not between 1–4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

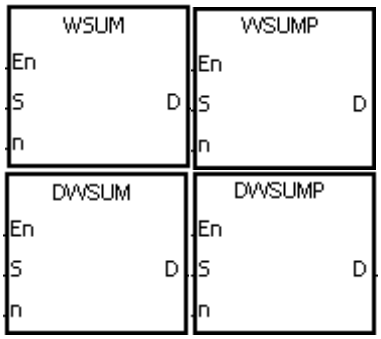
API	Instruction code			Operand	Function
1217	D	WSUM	P	**S, n, D**	Getting the sum

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●							
n					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
n		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



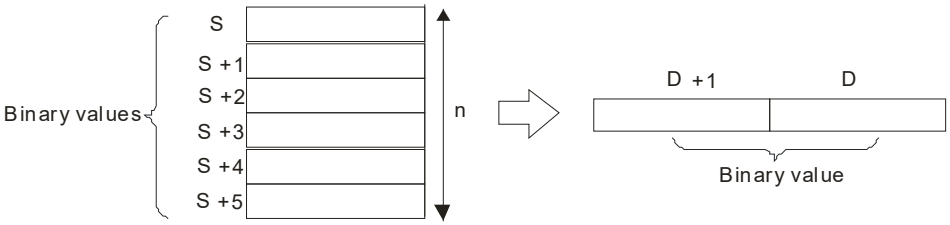
S : Data source

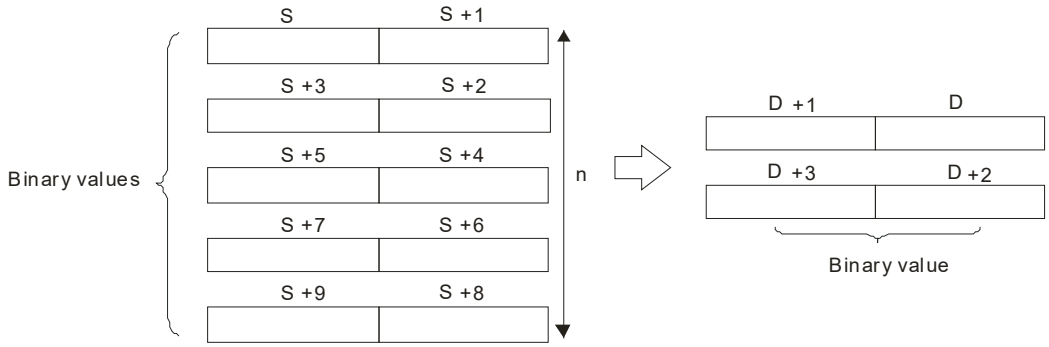
n : Data length

D : Operation result

Explanation

1. The instruction adds up the signed decimal values in **S** to **S**+**n**-1, and stores the sum in the register specified by **D**.





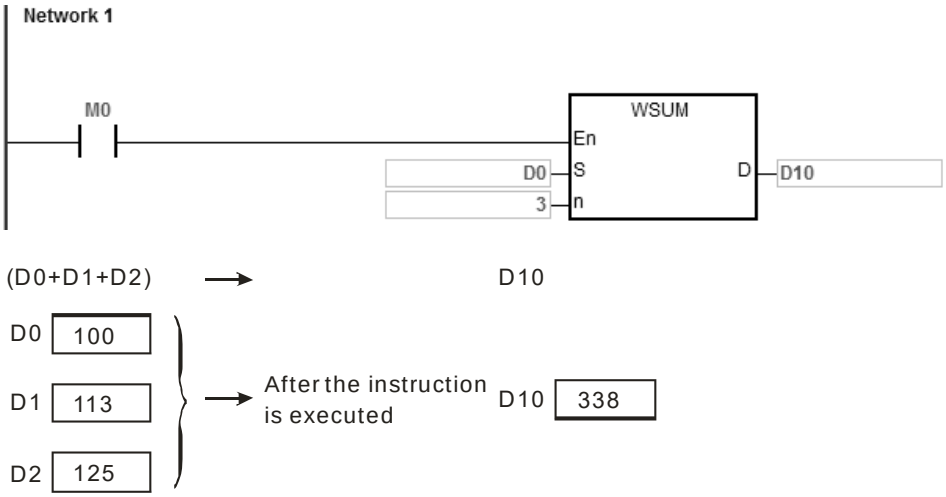
2. For 16-bit instructions, the value in **n** must be between 1–256.

3. For 32-bit instructions, the value in **n** must be between 1–128.

4. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

The WSUM instruction adds up the values in D0–D2, and stores the sum (32-bit) in D10.



Additional remarks

1. For 16-bit instructions, the value in **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. For 32-bit instructions, the value in **n** is not between 1–128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3. If **S**+**n**-1 or **D** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. For 16-bit instructions, ff you declare the operand **D** in ISPSoft, the data type is DWORD or ARRAY [2] of WORD.

5. For 32-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of DWORD.

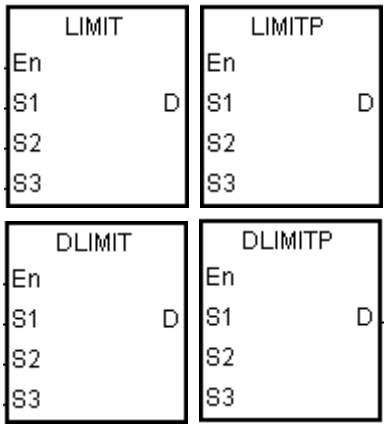
API	Instruction code			Operand	Function
1221	D	LIMIT	P	**S_1, S_2, S_3, D**	Confining a value within bounds

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Minimum output value

S_2 : Maximum output value

S_3 : Input value

D : Output value

Explanation

1. The instruction compares the input value in **S_3** with the minimum output value in **S_1** and the maximum output value in **S_2**, and stores the comparison result in **D**.

   If the input value in **S_3** is smaller than the minimum output value in **S_1**, the instructions stores minimum output value **S_1** in **D**.

   If the input value in **S_3** is larger than the maximum output value in **S_2**, the instruction stores the maximum output value **S_2** in **D**.

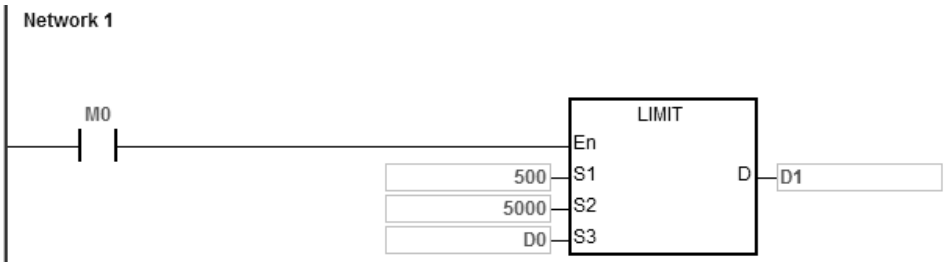
   If the input value in **S_3** is between the minimum output value **S_1** and the maximum output value **S_2**, the instruction stores the input value **S_3** in **D**.

If the minimum output value in **S_1** is larger than the maximum output value in **S_2**, the instruction is not executed.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

- When M0 is ON, the instruction compares the value in D0 with 500 and 5000, and stores the comparison result in D1.



Minimum output value	Maximum output value	Output value in D0	Function	Output value in D1
500	5000	499	D0<500	500
		5001	D0>5000	5000
		600	500≦D0≦5000	600

Additional remarks

If the minimum output value in **S_1** is larger than the maximum output value in **S_2**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

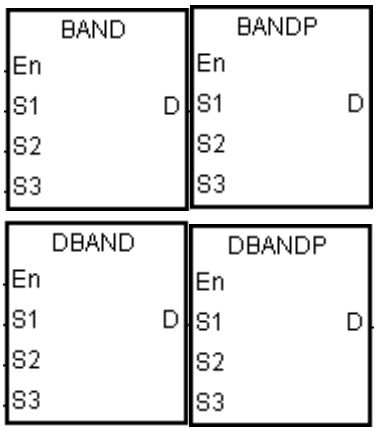
API	Instruction code			Operand	Function
1222	D	BAND	P	**S_1, S_2, S_3, D**	Deadband control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Minimum value of the deadband

S_2 : Maximum value of the deadband

S_3 : Input value

D : Output value

Explanation

1. This instruction subtracts the minimum value of the deadband in **S_1** or the maximum value of the deadband in **S_2** from the input value in **S_3**, and stores the difference in **D**.

   If the input value in **S_3** is smaller than the minimum value of the deadband in **S_1**, the instruction subtracts **S_1** from **S_3**, and stores the difference in **D**.

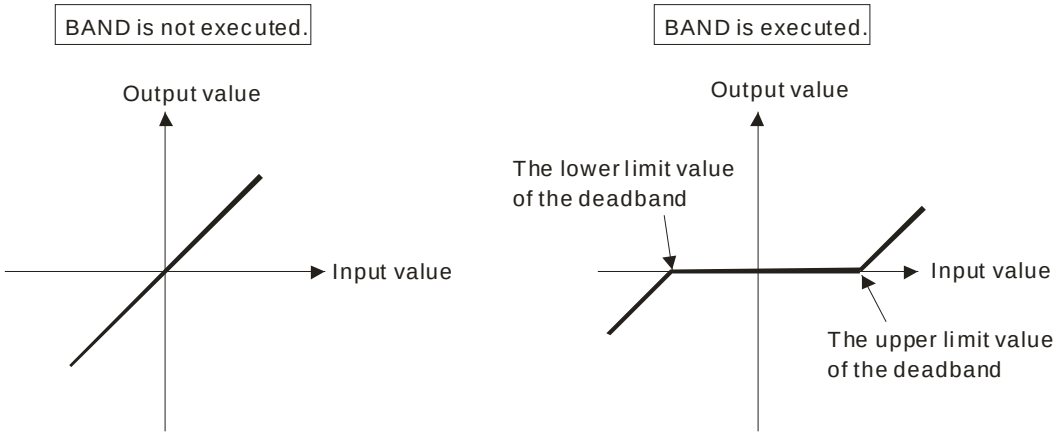
   If the input value in **S_3** is greater than the maximum value of the deadband in **S_2**, the instruction subtracts **S_2** from **S_3**, and stores the difference in **D**.

   If the input value in **S_3** is between the minimum of the deadband in **S_1** and the maximum value of the deadband in **S_2**, the instruction stores zero in **D**.

If the minimum value of the deadband in **S_1** is larger than the maximum value of the deadband in **S_2**, the instruction is not executed.

2. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

3. The following graphs show how this instruction uses the deadband.



4. The minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** must be within the range described below.

5. For the BAND instruction, the minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** must be between -32768 to 32767. Suppose the minimum value of the deadband in **S_1** is 10 and the maximum value of the deadband in **S_3** is -32768. The instruction calculates the output value in **D** as follows.

   Output value in **D** = -32768-10=16#8000-16#000A=16#7FF6=32758

6. For the DBAND instruction, the minimum value of the deadband in **S_1**, the maximum value of the deadband in **S_2**, the input value in **S_3**, and the output value in **D** must be between -2147483648 to 2147483647. Suppose the minimum value of the deadband in (**S_1**+1, **S_1**) is 1000 and the maximum value of the deadband in (**S_3**+1, **S_3**) is -2147483648. The instruction calculates the output value in (**D**+1, **D**) as follows.

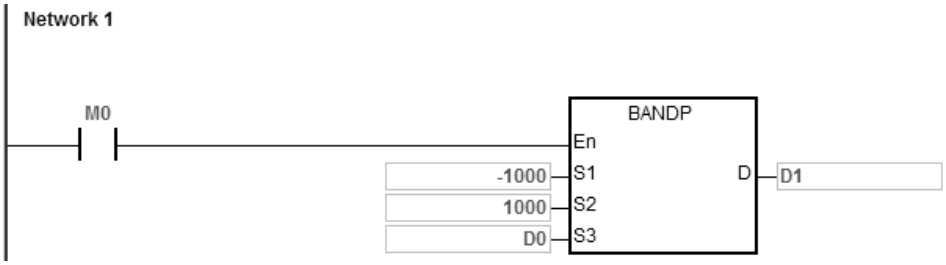
   Output value in (**D**+1, **D)**

   =-2147483648-1000=16#80000000-16#000003E8=16#7FFFFC18

   =2147482648

Example 1

When M0 is ON, the instruction subtracts -1000 or 1000 from the value in D0, and stores the difference in D1.

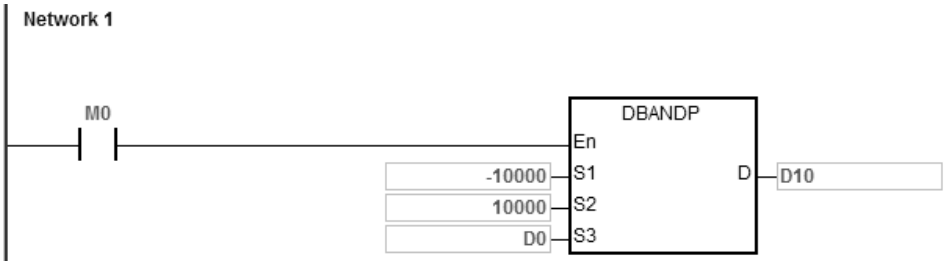


The following table shows the execution results.

Minimum value of the deadband	Maximum value of the deadband	Input value in D0	Function	Output value in D1
-1000	1000	-1200	D0<-1000=>D1=D0-（-1000）	-200
		1200	D0>1000=>D1=D0-1000	200
		500	-1000≦D0≦1000=>D0=0	0

Example 2

When M0 is ON, the instruction subtracts -10000 or 10000 from the value in (D0, D1), and stores the difference in (D11, D10).



The following table shows the execution results.

Minimum value of the deadband	Maximum value of the deadband	Input value in (D1, D0)	Function	Output value in (D11, D10)
-10000	10000	-12000	(D1 · D0) <-10000 => (D11 · D10) = (D1 · D0) - (-10000)	-2000
		12000	(D1 · D0) >10000 => (D11 · D10) = (D1 · D0) -10000	2000
		5000	-10000≦ (D1 · D0) ≦10000 => (D1 · D0) =0	0

Additional remarks

If the minimum value of the deadband in S_1 is larger than the maximum value of the deadband in S_2, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

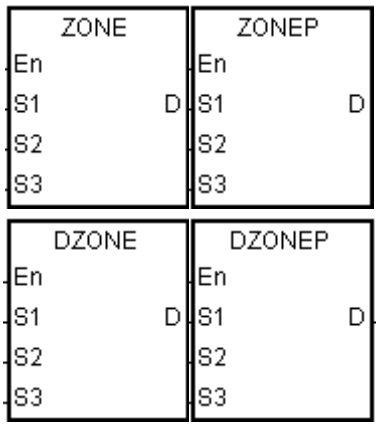
API	Instruction code			Operand	Function
1223	D	ZONE	P	**S_1, S_2, S_3, D**	Controlling the zone

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○	○	○	○		
S_2					●	●	●	●	●		○	○	○	○		
S_3					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S_1 : Negative deviation

S_2 : Positive deviation

S_3 : Input value

D : Output value

Explanation

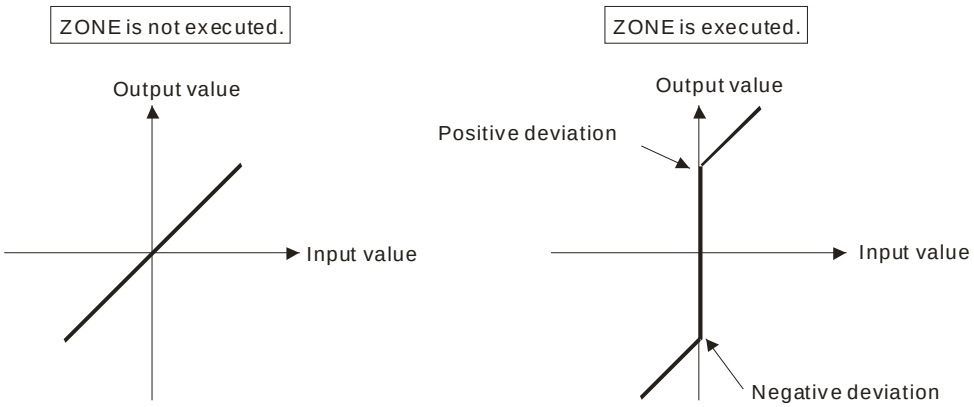
1. This instruction adds the negative deviation in **S_1** or the positive deviation in **S_2** to the input value in **S_3**, and stores the sum in **D**.

   If the input value in **S_3** is less than 0, the instruction adds the negative deviation in **S_1** to the input value in **S_3**, and stores the sum in **D**.

   If the input value in **S_3** is larger than 0, the instruction adds the positive deviation in **S_2** to the input value in **S_3**, and stores the sum in **D**.

   If the input value in **S_3** is equal to zero, the instruction stores zero in **D**.

2. The following graphs show how this instruction uses the zone:



3. Only the 32-bit instructions can use the 32-bit counter but not the device E.

4. The negative deviation in **S_1**, the positive deviation in **S_2**, the input value in **S_3**, and the output value in **D** must be within the range described below.

   - For the ZONE instruction, the negative deviation in **S_1**, the positive deviation in **S_2**, the input value in **S_3**, and the output value in **D** must be between -32768 to 32767. Suppose the negative deviation in **S_1** is -100 and the input value in **S_3** is -32768. The instruction calculates the output value in **D** as follows.

     Output value in **D**=(-32768)+(-100)=16#8000+16#FF9C=16#7F9C=32668

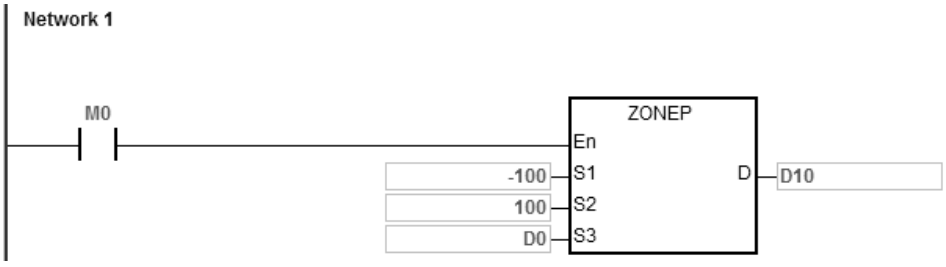
   - For the DZONE instruction, the negative deviation in **S_1**, the positive deviation in **S_2**, the input value in **S_3**, and the output value in **D** must be between -2147483648 to 2147483647. Suppose the negative deviation in (**S_1**+1, **S_1**) is -1000 and the input value in (**S_3**+1, **S_3**) is -2147483648. The instruction calculates the output value in (**D**+1, **D**) as follows.

     Output value in (**D**+1, **D)**

     =-2147483648+(-1000)=16#80000000+16#FFFFFC18=16#7FFFFC18=2147482648

Example 1

When M0 is ON, the instruction adds -100 or 100 to the value in D0, and stores the sum in D10.

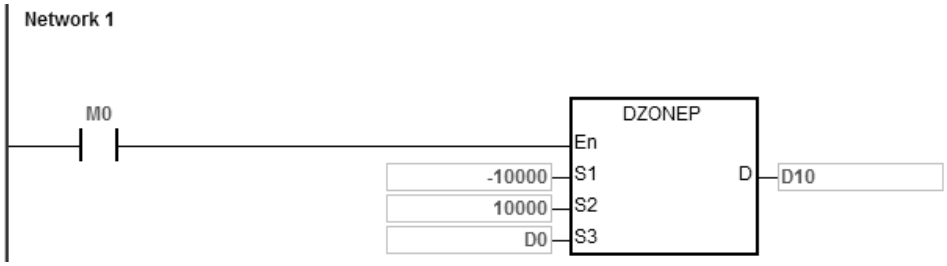


The following table shows the execution results.

Negative deviation	Positive deviation	Input value in D0	Function	Output value in D10
-100	100	-10	D0<0=>D10= (-10) + (-100)	-110
		0	D0=0=>D10=0	0
		50	D0>0=>D10=50+100	150

Example 2

When M0 is ON, the instruction adds -10000 or 10000 to the value in (D0, D1), and stores the sum in (D11, D10).



The following table shows the execution results.

Negative deviation	Positive deviation	Input value in (D1, D0)	Function	Output value in (D11, D10)
-10000	10000	-10	(D1 · D0) <0 => (D11 · D10) = (-10) + (-10000)	-10010
		0	(D1 · D0) =0 => (D11 · D10) =0	0
		50	(D1 · D0) >0 => (D11 · D10) =50+10000	10050

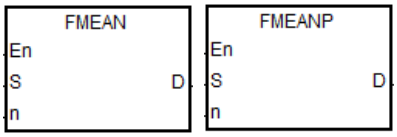
API	Instruction code			Operand	Function
1224		FMEAN	P	**S, D, n**	Finding the mean of floating point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							●	●								
n							●	●					○	○		
D							●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
n			●				●						
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S : First device

D : Device where the mean is stored

n : Number of devices

Explanation

1. This instruction adds up the single precision floating points in the **n** devices starting from the device specified by **S**, divides the sum by the value in **n**, then stores the mean of the sum in **D**.

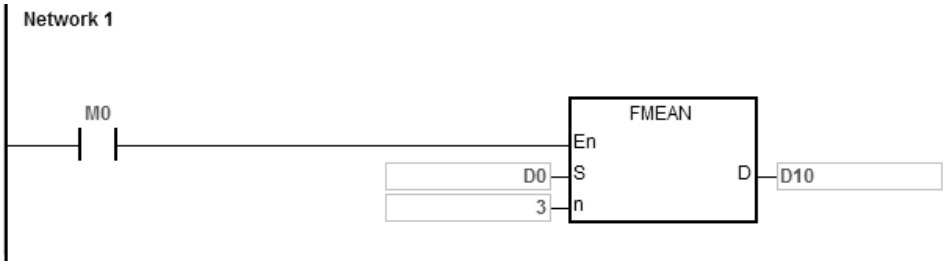
2. For 16-bit instructions, the value in **n** must be between 1–256,

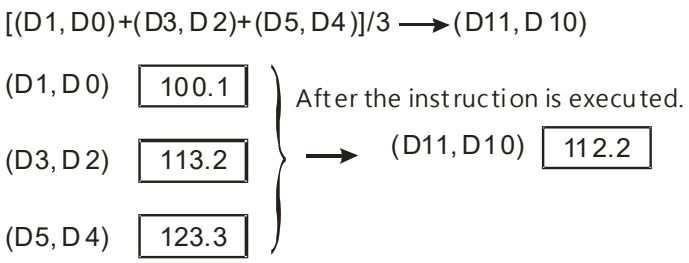
3. Instruction flags: SM600 (zero flag), SM601 (borrow flag), SM602 (carry flag):

   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
   - If the value while adding or the absolute result of the operation is less than the floating point number that can be shown, the D=16#FF800000 and the borrow flag SM601 is ON.
   - If the value while adding or the absolute result of the operation is larger than the floating point number that can be shown, the D=16#7F800000 and the carry flag SM602 is ON.

Example

When M0 is ON, the instruction adds the values of the 3 single precision floating points in (D1, D0), (D3, D2), (D5, D4) and then divides the addition result by 3, then stores the result in (D11, D10).





Additional remarks

1. If the value in **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **S**+2*__n__-1exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **S** exceeds the range of floating point numbers that can be shown, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

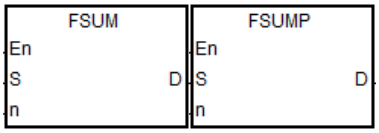
API	Instruction code			Operand	Function
1225		FSUM	P	**S, n, D**	Finding the sum of floating point numbers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							●	●								
n							●	●					○	○		
D							●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
n			●				●						
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S : Data source

n : Data length

D : Operation result

6

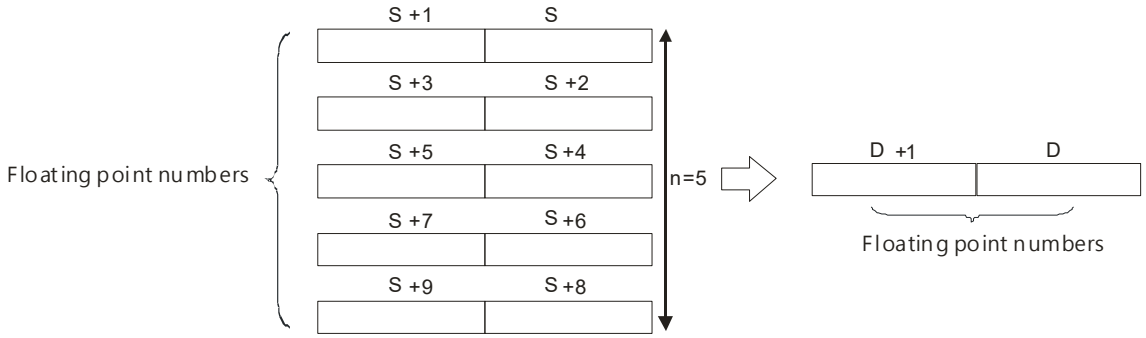
Explanation

1. This instruction adds up the single precision floating points in the **n** devices starting from the device specified by **S**, then stores the sum in **D**.

2. For 16-bit instructions, the value in **n** must be between 1–256,

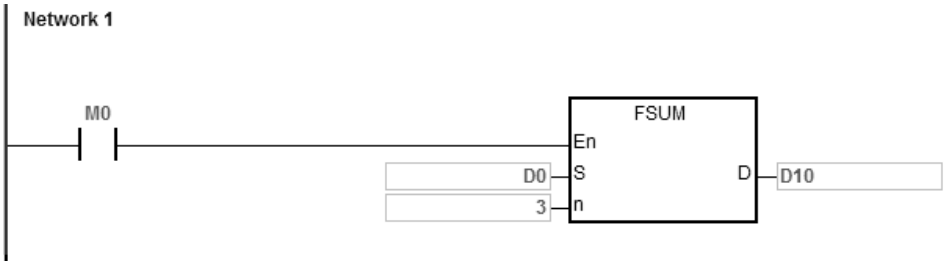
3. Instruction flags: SM600 (zero flag), SM601 (borrow flag), SM602 (carry flag):

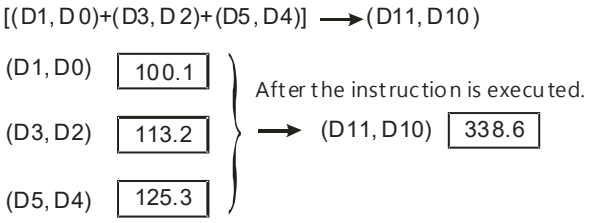
   - When the operation result is zero, SM600 is ON. Otherwise, it is OFF.
   - If the value while adding or the absolute result of the operation is less than the floating point number that can be shown, the D=16#FF800000 and the borrow flag SM601 is ON.
   - If the value while adding or the absolute result of the operation is larger than the floating point number that can be shown, the D=16#7F800000 and the carry flag SM602 is ON.



Example

The FSUM instruction adds up the values of the 3 single precision floating points in (D1, D0), (D3, D2), (D5, D4) and stores the result in (D11, D10).





Additional remarks

1. If the value in **n** is not between 1–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If **S**+2*__n__-1exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **S** exceeds the range of the floating point numbers that can be shown, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

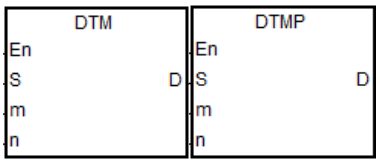
API	Instruction			Operand	Description
1226		DTM	P	**S，D，m，n**	Data conversion and move

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D					●	●		●								
m								●					○	○		
n								●					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							
m		●			●	●							
n		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Source data device

D : Device where the conversion result is stored

m : Conversion mode selection

n : The length of the data to be executed

Explanation

1. The parameter **m** is for you to select a conversion mode from the following table. See the details on modes in the following sections. If the parameter value is not one of the values in the following table, there will be no data conversion or move and no error message.

Parameter	Description
0	8-bit data converted into 16-bit data (high 8 bits, low 8 bits)
1	8-bit data converted into 16-bit data (low 8 bits, high 8 bits)
2	16-bit data (high 8 bits, low 8 bits) converted into 8-bit data
3	16-bit data (low 8 bits, high 8 bits) converted into 8-bit data
4	8-bit hex data (high 4 bits, low 4 bits) converted into ES3CII data
5	8-bit hex data (low 4 bits, high 4 bits) converted into ES3CII data.
6	8-bit ES3CII data converted into hex data (high 4 bits, low 4 bits).）
7	8-bit ES3CII data converted into hex data (low 4 bits, high 4 bits).
42	Caculate the duration of 2-Axis relative-position clockwise arc interpolation

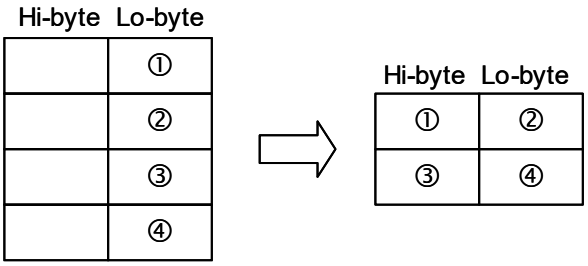
43	Caculate the duration of 2-Axis relative-position counterclockwise arc interpolation
44	Set up the built-in input points filtering time for the PLC

2. **n** is the setting value of data length. The range of the setting value is 1~256. If the input value exceeds the range, the PLC will execute the instruction at the minimum or maximum value.

3. The conversion modes and move modes are explained as below.

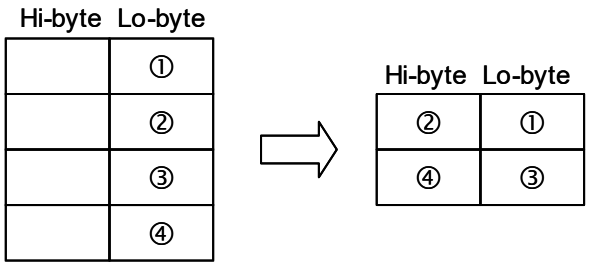
- **When m**=0:

  If **n**=4, the 8-bit data is converted into the 16-bit data (high 8-bits, low 8-bits), the conversion is as the following figure shows.



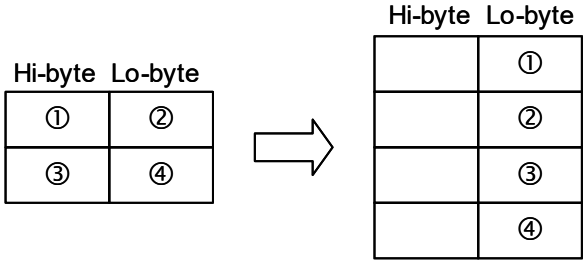
- **When m**=1:

  If **n**=4, the 8-bit data is converted into the 16-bit data (low 8-bits, high 8-bits), the conversion is as the following figure shows.

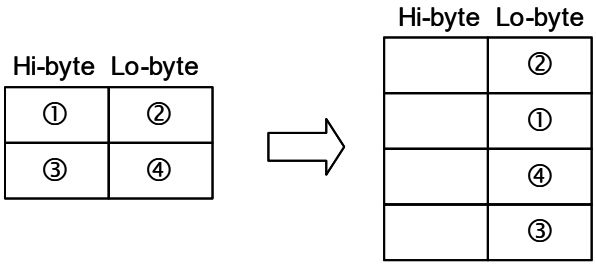


- **When m**=2:

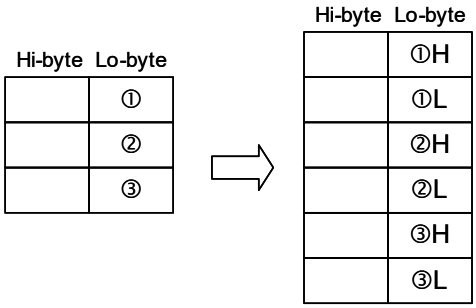
  If **n**=4, the 16-bit data (high 8-bits, low 8-bits) is converted into the 8-bit data, the conversion is as the following figure shows.



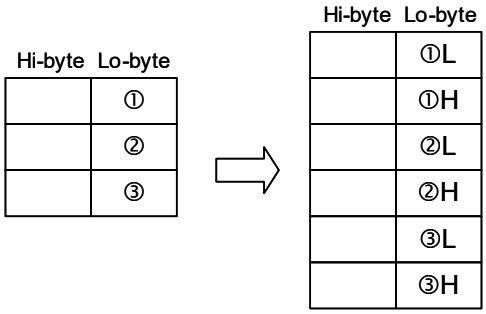
- **When m**=3:
  If **n**=2, the 16-bit data (low 8-bits, high 8-bits) is converted into the 8-bit data, the conversion is as the following figure shows.



- **When m**=4:
  If **n**=3, the 8-bit hex data (high 4-bits, low 4-bits) is converted into the ES3CII data and the conversion is as the following figure shows.

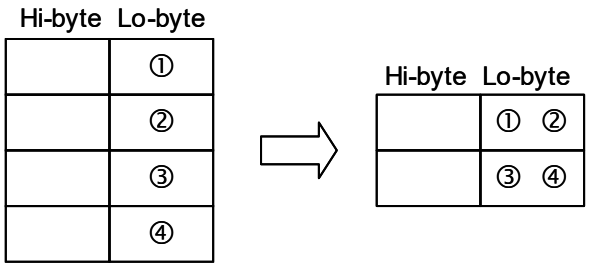


- **When m**=5:
  If **n**=3, the 8-bit hex data (low 4-bits, high 4-bits) is converted into the ES3CII data, the conversion is as the following figure shows.



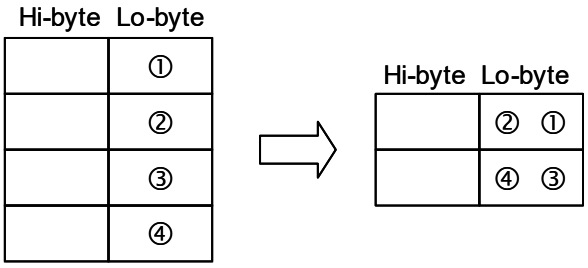
- **When m**=6:

  If **n**=4, the 8-bit ES3CII data is converted into the hex data (high 4-bits, low 4-bits), the conversion is as the following figure shows. ES3CII conversion values can be: 0 ~ 9 (0x30~0x39), A ~ F (0x41~0x46), a ~ f (0x61~0x66).



- **When m**=7:

  If **n**=4, the 8-bit ES3CII data is converted into the hex data (low 4-bits, high 4-bits), the conversion is as the following figure shows.



- **When m**=42:

  Operand S1:

  S1+0, S1+1: X axis target coordinate (relative positioning)

  S1+2, S1+3: Y axis target coordinate (relative positioning)

  S1+4, S1+5: Shift of the center (integer type) or central angle (floating point type)

  S1+6, S1+7: Target reference frequency (1~200000Hz; if exceeding the allowable range, the system uses the maximum or minimum value instead)

  S1+8: Function selections

  (K0, K1, K2: Shift of the center mode; shifts at 10 degree, 5 degree or 1 degree)

  (K10, K11, K12: Central angle; shifts at 10 degree, 5 degree or 1 degree)

  Operand D: D+0, D+1: Calculation of the duration of 2-Axis relative-position clockwise arc interpolation; unit:ms

Operand m: Function code K42

Operand n: Reserved

- **When m**=43：

Operand S1:

S1+0, S1+1: X axis target coordinate (relative positioning)

S1+2, S1+3: Y axis target coordinate (relative positioning)

S1+4, S1+5: Shift of the center (integer type) or central angle (floating point type)

S1+6, S1+7: Target reference frequency (1~200000Hz; if exceeding the allowable range, the system uses the maximum or minimum value instead)

S1+8: Function selections

(K0, K1, K2: Shift of the center mode; shifts at 10 degree, 5 degree or 1 degree)

(K10, K11, K12: Central angle; shifts at 10 degree, 5 degree or 1 degree)

Operand D: D+0, D+1: Calculation of the duration of 2-Axis relative-position anti-clockwise arc interpolation;

unit: ms

Operand m: Function code K43

Operand n: Reserved

- **When m**=44:

Operand S1: the starting point of PLC input points (K0~K15 ➔ X0~X17)

Operand D: filtering time; unit: μs (0~20000)

Operand m: function code K44

Operand n: total number of input points to be updated

Additional remarks

1. Using the matrix variables of Word data type for S and D is recommended.

6.14 Structure Creation Instructions

6.14.1 List of Structure Creation Instructions

The following table lists the Data Processing instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1300	FOR	–	–	Starting a nested loop
1301	NEXT	–	–	Ending a nested loop
1302	BREAK	–	–	Terminating a FOR-NEXT loop

6.14.2 Explanation of Structure Creation Instructions

API	Instruction code	Operand	Function
1300	FOR	**S**	Starting a nested loop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●			●	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

S : Number of times the loop is executed

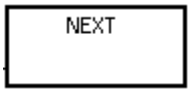
Explanation

Refer to the NEXT instruction (API 1301) for more details.

API	Instruction code			Operand	Function
1301		NEXT		-	Ending a nested loop

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Explanation

1. This instruction executes the program between the FOR and NEXT instructions N times, where N is the value in **S** specified for the FOR instruction (API 1300). After the program between the FOR and NEXT instructions is executed N times, the program following the NEXT instruction is executed. The instruction FOR specifies the number of times the program between the FOR and NEXT instructions is executed.

2. N must be between 1–32,767. If N is less than 1, the instruction processes it as 1.

3. If you do not want to execute the program between the FOR and NEXT instructions, you can skip it with the CJ instruction (API 0400).

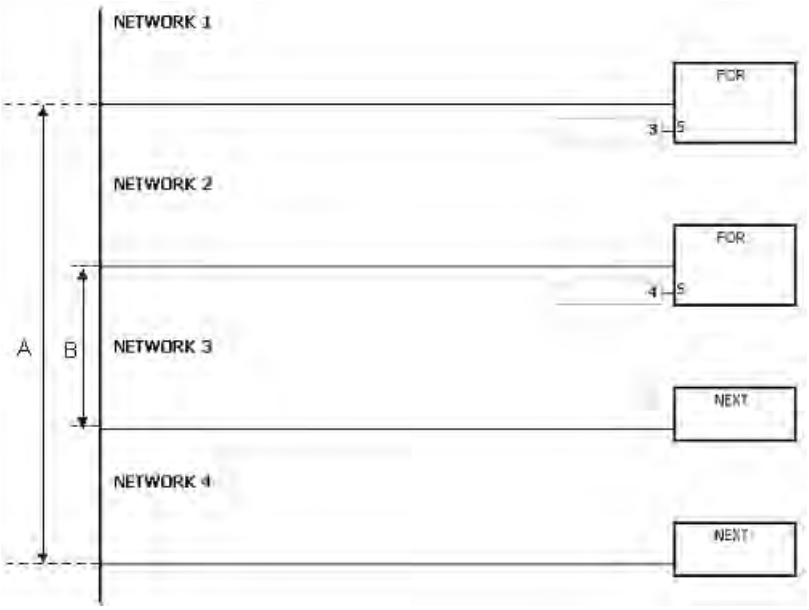
4. The following conditions result in errors.

   - The NEXT instruction is prior to the FOR instruction.
   - The FOR instruction exists, but the NEXT instruction does not exist.
   - The NEXT instruction follows the FEND or END instruction.
   - The number of times the FOR instruction is used in the program is different from the number of times the NEXT instruction is used in the program.

5. The FOR and NEXT instructions support the nested program structure. There can be at most 32 levels of nested program structures. If a loop is executed many times, it takes more time for the PLC to scan the program, and the watchdog timer error may occur. You can use the WDT instruction (API 1900) to resolve the problem.

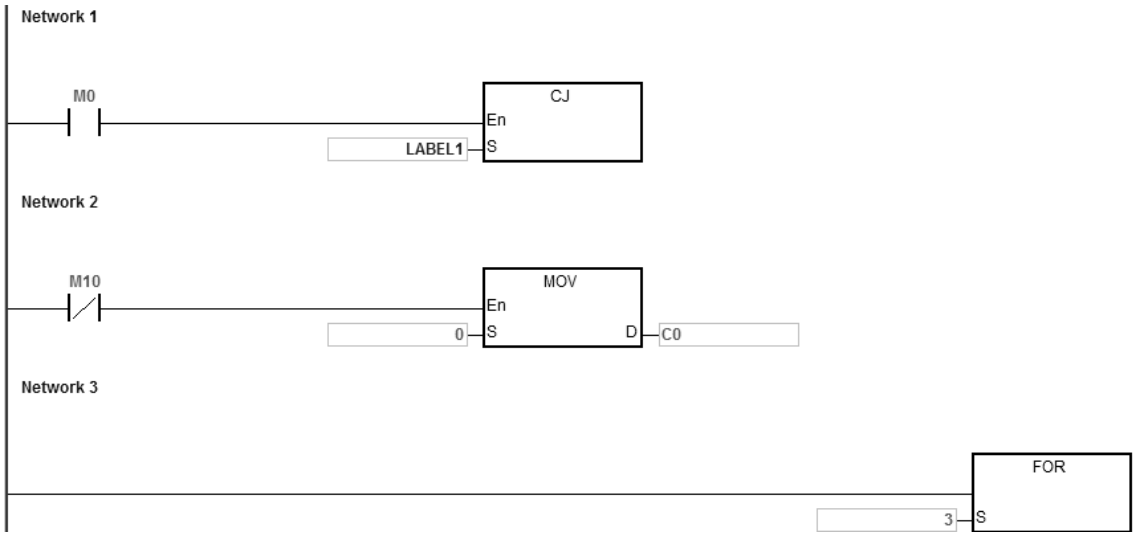
Example 1

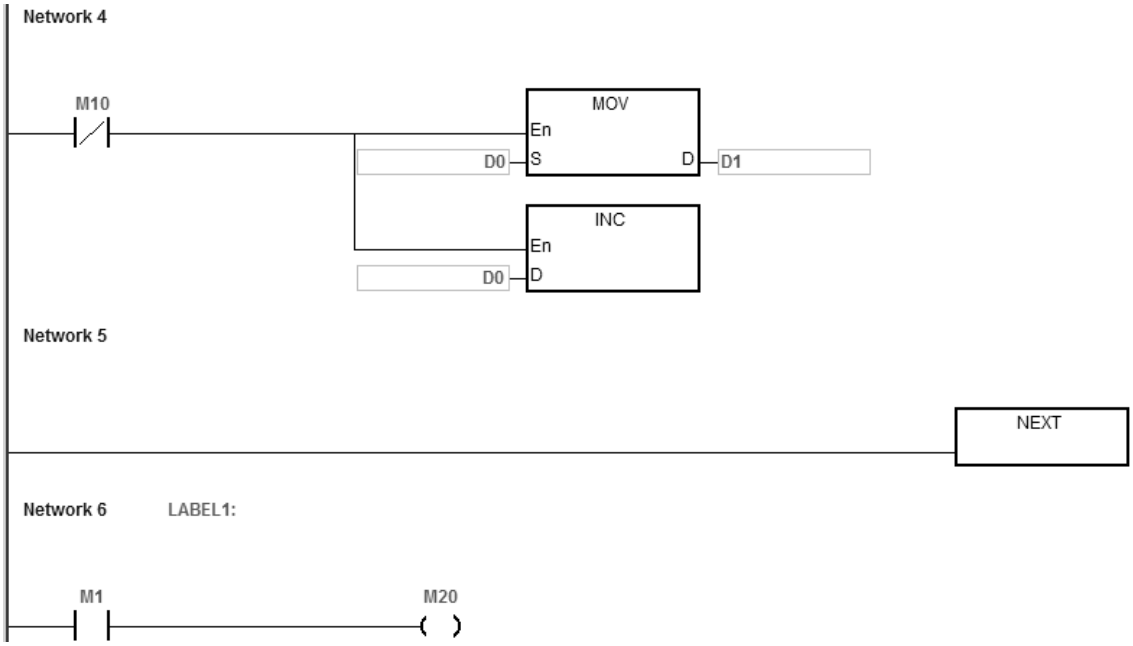
After program A is executed three times, the program following the instruction NEXT is executed. Program B is executed four times every time program A is executed. Therefore, program B is executed twelve times in total.



Example 2

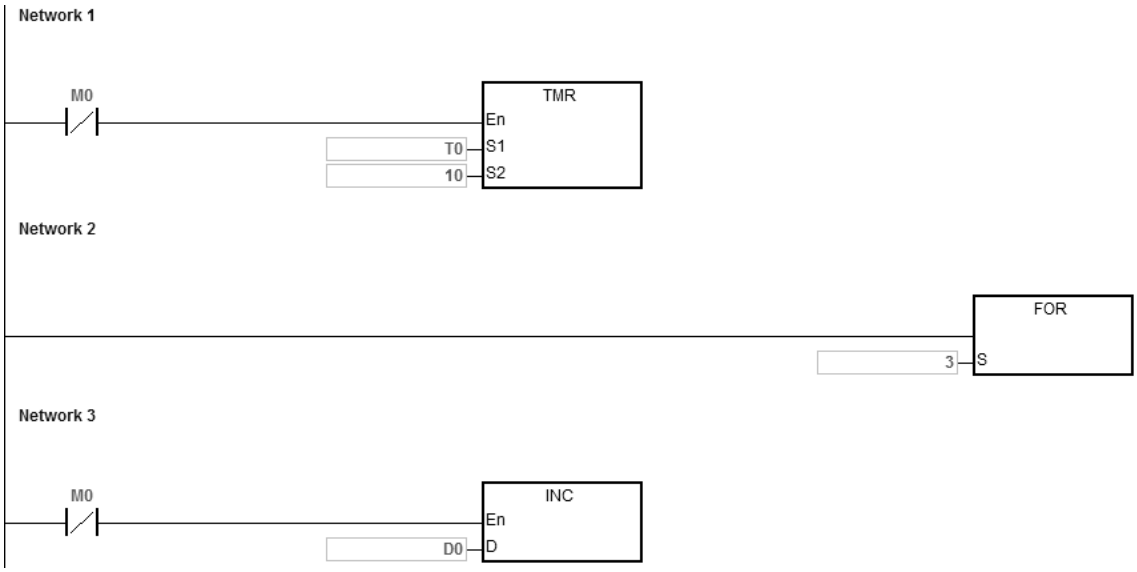
When M0 is OFF, the program between FOR and NEXT is executed. When M0 is ON, the CJ instruction is executed. The execution of the program jumps to LABEL 1:, i.e. network 6, and network 4–5 are not executed.

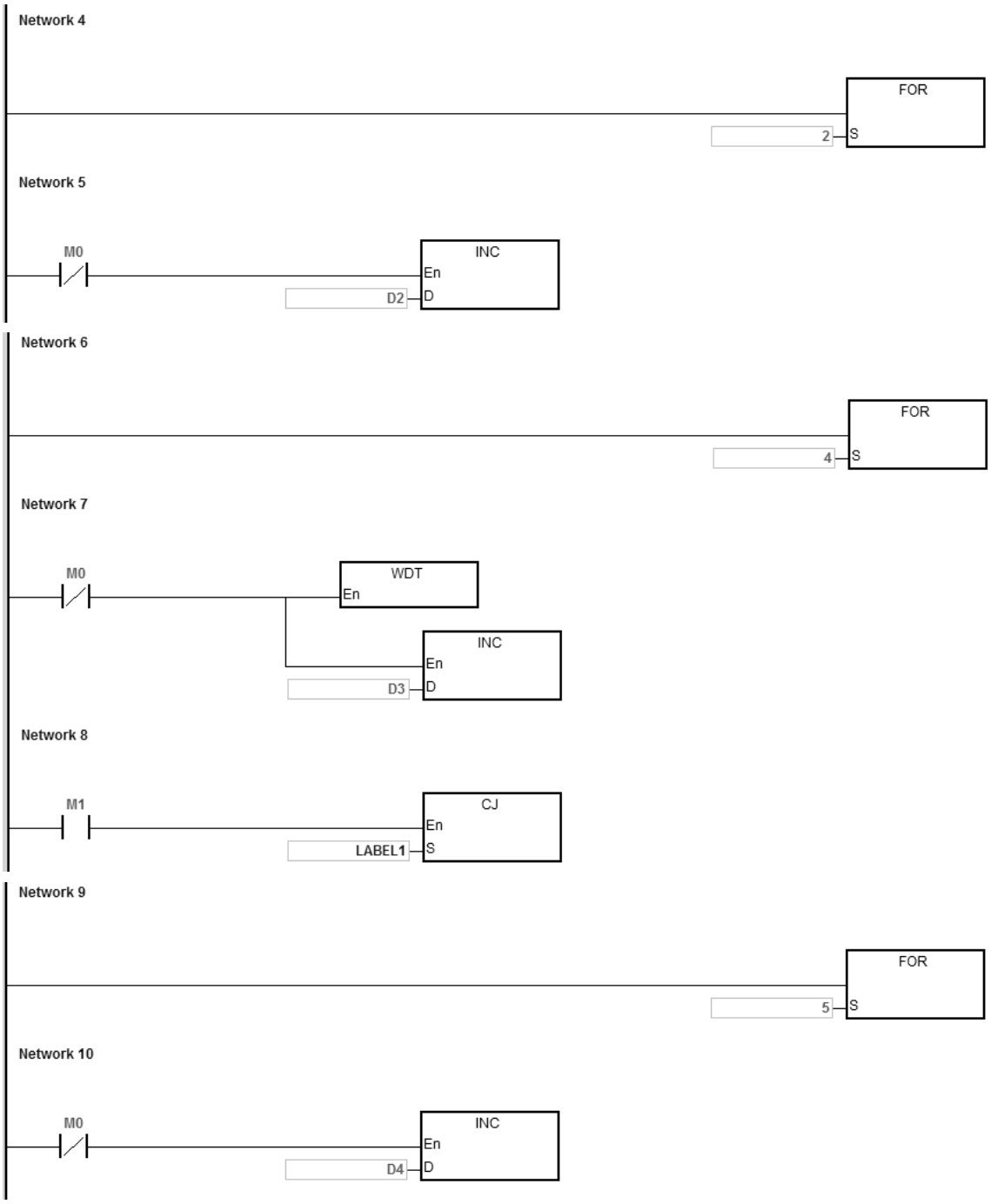




Example 3

If the program between FOR and NEXT is not to be executed, you can skip it with the CJ instruction. When M1 in network 8 is ON, the instruction CJ is executed. The execution of the program jumps to LABEL 1:, i.e. network 12, and network 9–11 are not executed.




Network 4
FOR
2
S
Network 5
M0
INC
En
D2
D
Network 6
FOR
4
S
Network 7
M0
WDT
En
INC
En
D3
D
Network 8
M1
CJ
En
LABEL1
S
Network 9
FOR
5
S
Network 10
M0
INC
En
D4
D

Additional remarks

Refer to the ISPSoft User Manual for more information on using labels.

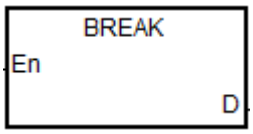
API	Instruction code	Operand	Function
1302	BREAK	**D**	Terminating the FOR-NEXT loop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●				●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



D : Device where the remaining number of times the loop can be executed is stored

Explanation

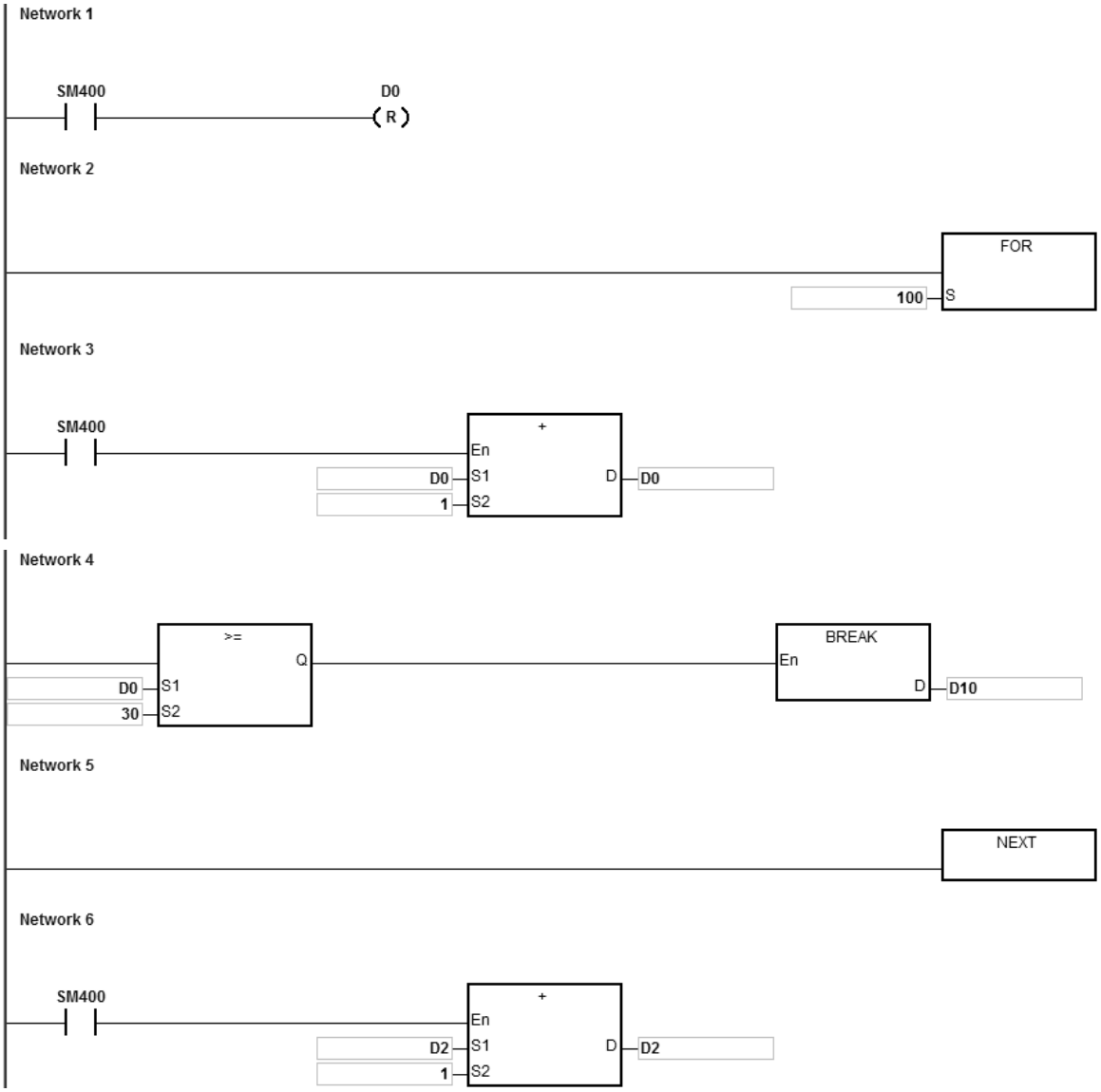
1. This instruction terminates the FOR/NEXT loop. The remaining number of times the FOR/NEXT loop can be repeated is stored in **D**. After the loop has executed **D** times, the program jumps to the NEXT instruction and executes the instruction after the NEXT instruction.

2. When the instruction is executed, the remaining number of times the FOR/NEXT loop can be repeated is stored in **D**, including this time the instruction BREAK is executed.

3. When the BREAK instruction is executed for the first time to terminate the FOR/NEXT loop, the program does not jump out of the FOR/NEXT loop to execute the next instruction. If the BREAK instruction is executed more than one time to terminate the FOR/NEXT loop, the program jumps to the NEXT instruction and executes the instruction following the NEXT instruction.

Example

When the FOR/NEXT loop is executed, 1 is added to the value in D0. When the value in D0 is equal to 30, the FOR/NEXT loop is terminated, and the remaining number of times the FOR/NEXT loop can be repeated, i.e. 71, is stored in D10. The execution of the program jumps to LABEL 1:, i.e. network 6, and 1 is added to the value in D2.



Additional remarks

1. If the instruction BREAK is outside the FOR/NEXT loop, it causes an operation error, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2017.
2. Refer to the ISPSoft User Manual for more information on using labels.

6.15 Module Instructions

6.15.1 List of Module Instructions

The following table lists the Module instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1400	FROM	DFROM	✓	Reading data from the control register in an extension module
1401	TO	DTO	✓	Writing data into the control register in an extension module
1402	PUCONF	DPUCONF	✓	Setting output control parameters of PU module
1403	PUSTAT	–	–	Reading PU module output state
1404	–	DPUPLS	–	PU module pulse output (no acceleration)
1405	–	DPUDRI	–	Relative position output of PU module (with acceleration and deceleration)
1406	–	DPUDRA	–	Absolute addressing output of PU module (with acceleration and deceleration)
1407	–	DPUZRN	–	PU module homing
1408	–	DPUJOG	–	PU module jog output
1409	–	DPUMPG	–	PU module MPG output
1410	–	DPUCNT	–	High-speed counter function of PU module

6.15.2 Explanation of Module Instructions

API	Instruction code			Operand	Function
1400	D	FROM	P	m_1, m_2, m_3, D_1, D_2, n	Reading data from the control register in an extension module

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
m_1					●	●	●	●	●		○	○	○	○		
m_2					●	●	●	●	●		○	○	○	○		
m_3					●	●	●	●	●		○	○	○	○		
D_1					●	●	●	●								
D_2					●	●	●	●								
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
m_3		●	●		●	●	●						
D_1		●	●		●	●	●						
D_2		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

m_1 : CPU module number or the remote extension module number

m_2 : Order numbers of the extension number

m_3 : Control register number

D_1 : Device where the data is stored

D_2 : Device where the error code is stored

n : Data length

Explanation

1. This instruction reads data from the control register in an extension module.
2. The value in m_1 must be 0. If the value in m_1 is NOT 0, it will be seen as 0.
3. The operand m_2 represents the number of the right-side special extension modules (digital I/O module excluded)

that are connected to the CPU module. The first device is number 1, the second device is number 2 and so on. Connected modules are counted and up to 8 devices can be connected.

4. The operand **m_3** specifies the control register number.

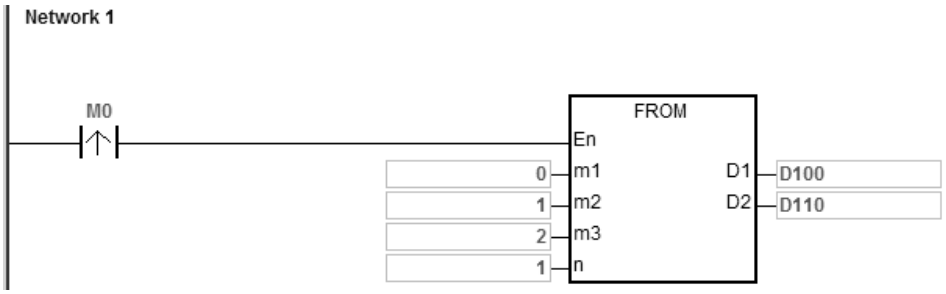
5. The FROM instruction sets **D_2** to 0. When an error occurs, the instruction does not set **D_2** to 0. Please refer to the Additional remarks below for more information about the error codes. When the instruction is not executed, **D_2** does not contain an error code.

6. The operand **n** must be between 1–8.

7. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

When M0 is switched from OFF to ON, the instruction reads the data stored in CR#2 from the right side of the first module and stores the data in D100. If no error occurs, the code in D110 is 16#0000.



Additional remarks

1. If the values in **m_1** and **m_2** exceed their range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D_1** to **D_1+n**-1 exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **n** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. Due to the fact that the FROM instruction decreases the execution efficiency of both the CPU module and the I/O module, it is suggested that you use the pulse type instruction to perform a single trigger as in the example shown above.

5. If there is any error response from the modules, the instruction stores the error code in **D_2**. The error code descriptions shown in the following table.

Error code	Description
16#1400	Attempted to read the data from the control register (CR) in the module but no such CR number exists.
16#1401	The value is not valid for the module.
16#1402	The module is not responding, communication timeout.

API	Instruction code			Operand	Function
1401	D	TO	P	**m_1**, **m_2**, **m_3**, **S**, **D**, **n**	Writing data into the control register in an extension module

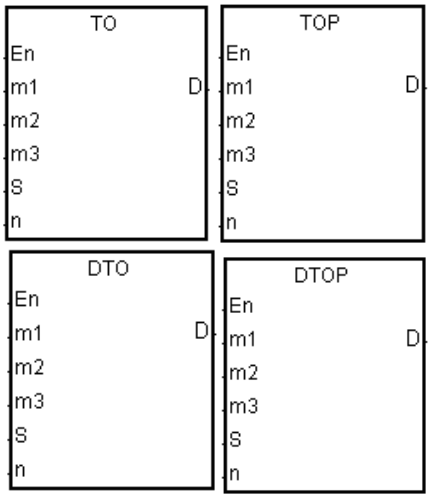
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
m_1					●	●	●	●	●		○	○	○	○		
m_2					●	●	●	●	●		○	○	○	○		
m_3					●	●	●	●	●		○	○	○	○		
S					●	●	●	●	●				○	○		
D					●	●	●	●								
n					●	●	●	●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m_1		●	●		●	●	●						
m_2		●	●		●	●	●						
m_3		●	●		●	●	●						
S		●	●		●	●	●						
D		●	●		●	●	●						
n		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

6

Symbol



m_1 : CPU module number or the remote extension module number

m_2 : Order numbers of the extension number

m_3 : Control register number

S : Device where the data is stored

D : Device where the error code is stored

n : Data length

Explanation

1. This instruction writes data to the control register in an extension module.

2. The value in **m_1** must be 0. If the value in **m_1** is NOT 0, it will be seen as 0.

3. The operand **m_2** represents the number of the right-side special extension modules (digital I/O module excluded)

that are connected to the CPU module. The first device is number 1, the second device is number 2 and so on. Connected modules are counted and up to 8 devices can be connected.

4. The operand **m_3** specifies the control register number.

5. The TO instruction sets **D** to 0. When an error occurs, the instruction does not set **D_2** to 0. Please refer to the Additional remarks below for more information about the error codes. When the instruction is not executed, **D** does not contain an error code.

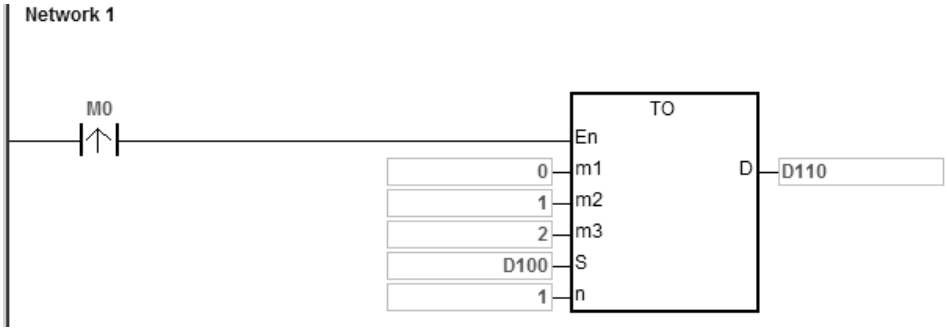
6. The operand **m_1** must be between 1–8.

7. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

8. When **S** is a hexadecimal value, the instruction transmits **n** hexadecimal values to the I/O module. Suppose **S** is 16#0001 and **n** is 3. The instruction transmits three 16#0001s to the I/O module.

Example

When M0 switches from OFF to ON, the TO instruction writes the data stored in D100 to CR#2 in the right side of the first module. If no error occurs, the code in D110 is 16#0000.



Additional remarks

1. If the values in **m_1** and **m_2** exceed their range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D_1**-**D_1**+**n**-1 exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **n** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

4. Due to the fact that the TO instruction decreases the execution efficiency of both the CPU module and the I/O module, it is suggested that you use the pulse type instruction to perform a single trigger as in the example shown above.

5. If there is any error response from the modules, the instruction stores the error code in **D**$_2$. The error code descriptions are shown in the following table.

Error code	Description
16#1400	Attempted to read the data from the control register (CR) in the module but no such CR number exists.
16#1401	The value is not valid for the module.
16#1402	The module is not responding, communication timeout.

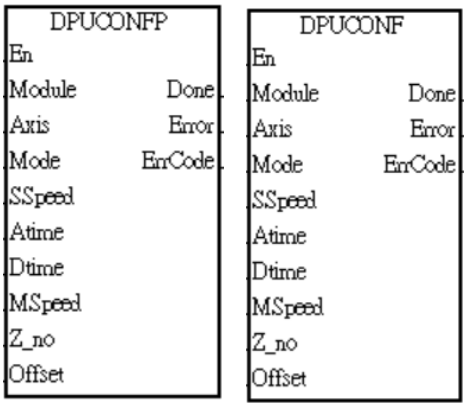
API	Instruction			Operand	Description
1402	D	PUCONF	P	**Module ~ Error, ErrCode**	Setting output control parameters of PU module

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
Mode								●					○	○		
SSpeed								●					○	○		
Atime								●					○	○		
Dtime								●					○	○		
MSpeed								●					○	○		
Z_no								●					○	○		
Offset								●					○	○		
Done		●	●	●				●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
Mode		●			●	●							
SSpeed		●			●	●							
Atime		●			●	●							
Dtime		●			●	●							
Mspeed			●				●						
Z_no		●			●	●							
Offset		●			●	●							
Done	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	—	ES3

Symbol



Module : Module number

Axis : Output axis number

Mode : Output mode

SSpeed : Speed for starting/ ending frequency

Atime : Acceleration time

Dtime : Deceleration time

MSpeed : Maximum output frequency

Z_no : Number of Z phases to look for after returing to the original point

Offset : Specify the number of outputs after returning to the original point

Done : Completion flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

   See the following combination of axis numbers and corresponding output points of PU modules.

PU module name	Axis 1	Axis 2
DVP02PU-E2	Y0 / Y1	Y2 / Y3

4. **Mode** sets the output mode of an output axis and the setting values are explained in the following table.

Output mode value	Description	Remark
0	Single-point pulse output (An even-number point for output only)	E.g. Y0 or Y2 for output
1	Pulse (An even-number point) + direction (An odd-number point)	E.g. Y0 is for the pulse and Y1 is for the direction. Y1: ON, negative direction; Y1: OFF, positive direction
2	CW (An even-number point) + CCW (An	E.g. Y0 is for CW (positive direction) and Y1

	odd-number point)	is for CCW (negative direction)
3	Phase A (An even-number point) + Phase B (An odd-number point)	E.g. Y0 is for phase A and Y1 is for phase B. When phase A is leading phase B: positive direction; when phase B is leading phase A: negative direction
Other value	Automatically switch to mode 1 (default value)	

5. **SSpeed~ Offset**

See the explanation of the following non-latched parameters and setting values. If the setting values exceed the range, the instruction will automatically be executed at the minimum or maximum value.

Parameter	Function	Range	Default	Remark
SSpeed	Starting/ending frequency	0 ~ 10,000 (Unit: Hz)	100	
Atime	Acceleration time	0 ~ 10,000 (Unit: ms)	100	
Dtime	Deceleration time	0 ~ 10,000 (Unit: ms)	100	
MSpeed	Maximum output frequency	100 ~ 200,000 (Unit: Hz)	100K	A 32-bit value
Z_no	Number of Z phase signals to seek after returing to the origin.	-100 ~ 100 (Unit: times)	0	0: disabled
Offset	Outputs the offset position after the homing is finished and Z phase seeking is done.	-10,000 ~ 10,000 (Unit: pulses)	0	0: disabled

6. **Done**, an output of the specified PU module has been set as the completion flag. When **Done** is On, it indicates that the parameter setting is successful. You can continue to perform positioning output based on the On state of the completion flag. The clearing of the **Done** flag need be conducted by manual. The **Done** flag changes to ON only when the setting is completed.

7. **Error**, an output of the specified PU module is a parameter error flag. Most parameter ranges are filtered automatically by the PLC. Thus if the error flag is ON, it means that there is no specified PU module or the PU module number is wrong or the output axis number is incorrect.

8. The instruction is a pulse instruction. Even if the A contact is adopted as the condition contact, PU module parameters are also set only when the instruction is started. Therefore, if a parameter value is to be updated, restart the instruction to make the parameter set again.

9. Since the set parameters are delivered through the module communication command, confirm the state of the output **Done** or **Error** before a parameter value is modified and then proceed with relevant operations.

10. **ErrCode** shows error codes. See the description as follows.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.

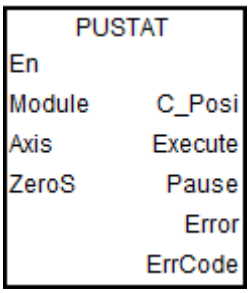
API	Instruction	Operand	Description
1403	PUSTAT	**Module ~ ErrCode**	Reading PU module output state

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
ZeroS	●	●	●	●				●								
C_Posi								●								
Execute		●	●	●				●								
Pause		●	●	●				●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
ZeroS	●												
C_Posi			●				●						
Execute	●												
Pause	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



Module : Module number

Axis : Output axis number

ZeroS : Clear present output position to 0

C_Posi : Current output position

Execute : Execution flag

Pause : Pause flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **C_Posi** sets the present position of the output axis for the specified PU module. The parameter value is a latched value and stored in the PU module. If the value is to be cleared to 0, set **ZeroS** from Off to ON when the instruction is started.

5. **Execute** is an only-read flag which means the output axis of the specified PU module is outputting or not. When **Execute** is On, it means the output is being conducted. When **Execute** is Off, it means the output axis is unused and can accept the next output command.

6. **Pause** is an only-read flag to control the output axis of the specified PU module to pause its output. When **Pause** is On, it means the output is paused, the present velocity is 0 and the present output has not reached the specified target output position. If you restore the output, the flag will be cleared automatically.

   Note: While **Pause** is On, **Execute** is constantly On as well.

7. **Error** is an only-read error flag which means an error occurs during the reading of the specified PU module. Refer to the explanation of error codes in **ErrCode**.

8. After the PUSTAT instruction gives the pause command, the flags **Execute, Pause and Error** become read-only flags and at the moment, their states cannot be modified. The **Execute, Pause and Error** flags can be set or cleared only when the PUSTAT instruction is turned off.

9. **ErrCode** shows error codes and the explanations are seen in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

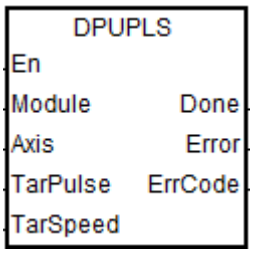
API	Instruction			Operand	Description
1404	D	PUPLS		**Module ~ ErrCode**	PU module pulse output (no acceleration)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
TarPulse								●					○	○		
TarSpeed								●					○	○		
Done		●	●	●				●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
TarPulse			●				●						
TarSpeed			●				●						
Done	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



Module : Module number

Axis : Output axis number

TarPulse : Target number of output pulses

TarSpeed : Target output frequency

Done : Completion flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the

error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **TarPulse** sets the number of output pulses. The pulse number is a positive signed 32-bit value. When the value is 0, it means the output is always being performed, the number of output pulses is not limited and the output is not stopped until the instruction is disabled. When the value is less than 0, the PLC automatically uses 2's complement to transform the value into a positive integer as the number of output pulses.

5. **TarSpeed** sets the target output speed (Unit: Hz). The input value is a signed 32-bit value within the range of -100,000 (-100K) ~ 100,000 (100K). You can modify the target frequency any time after the instruction is enabled and the PU module will automatically switch to the newly set target frequency after outputting a full pulse.

   Note: Before the target frequency is changed, please take into consideration whether the modified speed and PLC scan time match or not.

6. When **TarSpeed** is a positive number (>0), it means that the “positive direction” output point is Off. When **TarSpeed** is a negative number (<0), it means that the “negative direction” output point is On. When **TarSpeed** is 0, it means that the output will be paused after the being executed pulse is output fully.

7. The instruction does not support the function of acceleration and deceleration. Use the DPUDRI instruction instead if you need the function of acceleration and deceleration.

8. The instruction can be used for the speed change. While the instruction is being executed, you can change the value of **TarSpeed** so as to change the output speed.

9. When the outputs have reached the pulse number specified by **TarPulse**, the **Done** flag changes to ON. The **Done** flag need be cleared by manual. The instruction sets the completion flag to ON only when the output is completed.

10. If any error occurs as the instruction is in process of the output, the **Error** flag changes to ON. Refer to the error codes **ErrCode** shows for the trouble shooting.

    The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The value stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

API	Instruction			Operand	Description
1405	D	PUDRI		**Module ~ ErrCode**	Relative position output of PU module (with acceleration and deceleration)

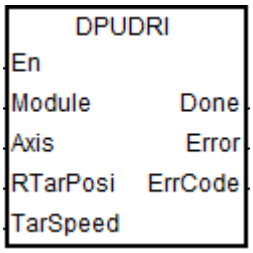
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
RTarPosi								●					○	○		
TarSpeed								●					○	○		
Done		●	●	●				●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
RTarPosi			●				●						
TarSpeed			●				●						
Done	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



Module : Module number

Axis : Output axis number

RTarPosi : Number of output pulses for relative positioning

TarSpeed : Target output frequency

Done : Completion flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **RTarPosi** sets the position for relative positioning. The pulse number is a signed 32-bit value. When the value is greater than 0, the output will go in the positive direction (and the direction output point is off). When the value is less than 0, the output will go in the negative direction (and the direction output point is on). When the value is 0, the output completion flag **Done** changes to ON.

5. **TarSpeed** sets the target output frequency (Unit: Hz). The frequency value is a positive signed 32-bit integer. When the value is less than 0, the instruction will automatically use 2's complement to transform the value into a positive integer. When the value is 0, the instruction will notify the module to enter the pause mode. The actual output is decelerated at the deceleration rate till the output speed is equal to 0 and the pause flag changes to ON. Refer to PUSTAT instruction for more details.

6. After the output is started, the target frequency is allowed to change any time. In the actual frequency change, the PLC will automatically change the frequency based on the set acceleration and deceleration rate in the DPUCONF instruction.

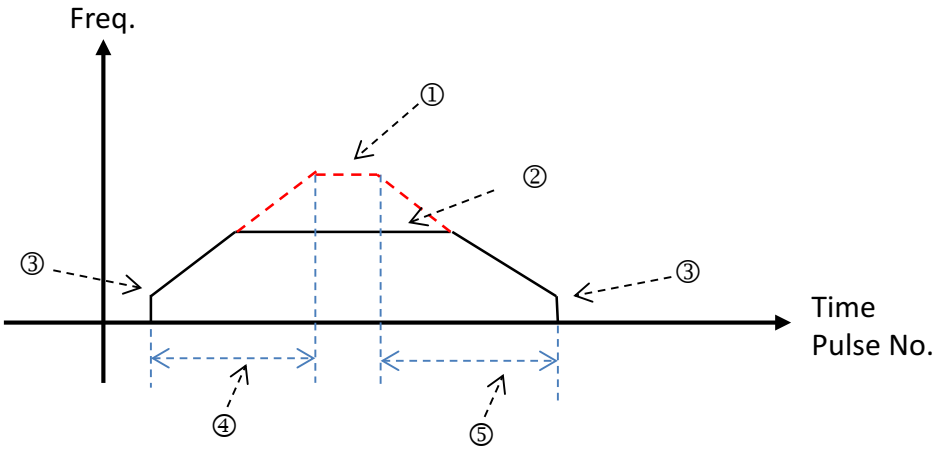
7. When the outputs have reached the pulse number for relative positioning specified by **RTarPosi**, the **Done** flag changes to ON. The **Done** flag need be cleared by manual. The instruction sets the completion flag to ON only when the output is completed.

8. If any error occurs as the instruction is in process of the output, the **Error** flag changes to ON. Refer to the error codes that **ErrCode** shows for the trouble shooting.

9. The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

10. Illustration of the acceleration and deceleration curve of the DPUDRI instruction



① : Maximum output frequency value. Refer to the setting in the DPUCONF instruction for the parameter setting. Alternatively, set the parameter value through HWCONFIG.

②: The target frequency specified by the PU module output instruction. The target frequency output must not exceed the maximum output frequency. If the maximum output frequency is exceeded, the maximum output frequency is regarded as the output frequency.

③: Starting/ending output frequency value. Refer to the setting in the DPUCONF instruction for the parameter setting. Alternatively, set the parameter value through HWCONFIG.

④: The acceleration time value. Refer to the setting in the DPUCONF instruction for the parameter setting. Alternatively, set the parameter value through HWCONFIG.

⑤: The deceleration time value. Refer to the setting in the DPUCONF instruction for the parameter setting. Alternatively, set the parameter value through HWCONFIG.

The acceleration and deceleration that the PU module controls is performed according to the fixed slope. So the actual acceleration time and deceleration time change based on the output target frequency. The formula for calculation of acceleration rate and deceleration rate are respectively shown as follows.

(Max. output frequency - starting frequency)/acceleration time;

(Max. output frequency - ending frequency)/deceleration time.

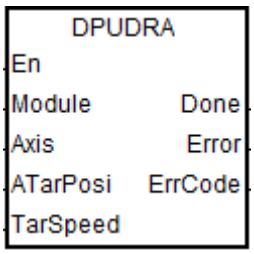
API	Instruction			Operand	Description
1406	D	PUDRA		**Module ~ ErrCode**	Absolute addressing output of PU module (with acceleration and deceleration)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
ATarPosi								●					○	○		
TarSpeed								●					○	○		
Done		●	●	●												
Error		●	●	●												
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
ATarPosi			●				●						
TarSpeed			●				●						
Done	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



Module : Module number

Axis : Output axis number

ATarPosi : Number of output pulses for absolute addressing

TarSpeed : Target output frequency

Done : Completion/pause flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **ATarPosi** is the position for absolute addressing. The input pulse number is a signed 32 bit value. The PU module will automatically compare it with the present position. If the comparison result is greater than 0, the output will be conducted in the positive direction (and the direction output point is off). If the comparison result is less than 0, the output will be conducted in the negative direction and the direction output point is on). When the value is 0, the instruction sets the **Done** flag to ON.
5. Refer to the DPUDRI instruction for the explanation of other parameters.

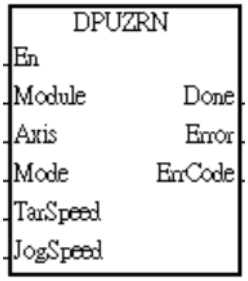
API	Instruction			Operand	Description
1407	D	PUZRN		**Module ~ ErrCode**	PU module homing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
Mode								●					○	○		
TarSpeed								●					○	○		
JogSpeed								●					○	○		
Done		●	●	●												
Error		●	●	●												
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
Mode		●			●	●							
TarSpeed			●				●						
JogSpeed		●			●	●							
Done	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	ES3

Symbol



Module : Module number

Axis : Output axis number

Mode : Homing mode selection

TarSpeed : Maximum output frequency for the homing

JogSpeed : The jog frequency for the homing

Done : Completion flag

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **Mode** sets a homing mode. The explanation of modes is shown in the following table.

Mode value	Function description	Input point used with other setting together	Remark
0	Directly clear current position to 0.	None	
1	Specify the point where DOG point is touched as the original point; the axis starts to go toward the negative direction and then stops after leaving the DOG point position.	DOG	Use the setting in HWCONFIG
2	Specify the point where DOG point is touched as the home point; the axis starts to go toward the negative direction and then stops after leaving the DOG point position.	DOG	Use the setting in HWCONFIG
3	After the behavior of when Mode=1 is finished, seek the set number of Z phases.	DOG and Z phase input	Use the setting in the DPUCONF instruction
4	After the behavior of when Mode=2 is finished, seek the set number of Z phases.	DOG and Z phase input	Use the setting in the DPUCONF instruction
5	After the behavior of when Mode=1 is finished, output the specified number of output pulses.	DOG	Use the setting in the DPUCONF instruction
6	After the behavior of when Mode=2 is finished, the PLC outputs the specified number of output pulses.	DOG	Use the setting in the DPUCONF instruction
7	The behavior of when Mode=1 + positive and negative limits	DOG, negative limit, positive limit	Use the setting in HWCONFIG

8	The behavior of when Mode=2 + positive and negative limits	DOG, negative limit, positive limit	Use the setting in HWCONFIG
9	After the behavior of when Mode=1, positive and negative limits are completed, seek Z phase based on the set Z phase number.	DOG, negative limit, positive limit and Z phase input	Use the setting in HWCONFIG or DPUCONF
10	After the behavior of when Mode=2 and positive and negative limits are completed, seek Z phase based on the set Z phase number.	DOG , negative limit, positive limit and Z phase input	Use the setting in HWCONFIG or DPUCONF
11	After the behavior of when Mode=1 and positive and negative limits are completed, output the specified number of output pulses.	DOG, negative limit, positive limit	Use the setting in HWCONFIG
12	After the behavior of when Mode=2, and positive and negative limits are completed, output the specified number of output pulses.	DOG, negative limit, positive limit	Use the setting in HWCONFIG
255	Modify the current output position for the axis.	None	Use the setting value of **TarSpeed**
Other	Reserved		

Note: The specified homing behavior may not be realized if the input points for the selected mode are not used with the setting in HWCONFIG together.

5. **TarSpeed** sets the maximum output frequency for the homing. The setting value is a signed 32-bit value. When **Mode** value is between 0~10, the range of the setting value is 100~100,000 (Hz). If **Mode** value is 255, **TarSpeed** value will become the present output position value of the PU module.

6. **JogSpeed** is the jog frequency for reaching the home position. The setting value is a signed 16 bit value within the range of 1~10,000 (Hz).

7. When the specified home position is reached during the instruction is executed, the **Done** flag changes to ON. The **Done** flag need be cleared by manual. The instruction sets the completion flag to ON only when the output is completed.

8. If any error occurs as the instruction is in process of the output, the **Error** flag changes to ON. Refer to the error codes that **ErrCode** shows for the trouble shooting.

9. The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

API	Instruction			Operand	Description
1408	D	PUJOG		**Module - ErrCode**	PU module jog output

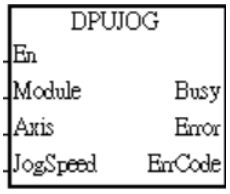
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
JogSpeed								●					○	○		
Busy		●	●	●				●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
JogSpeed			●				●						
Busy	●												
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



Module : Module number

Axis : Output axis number

JogSpeed : Jog output frequency

Busy : Output in execution

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

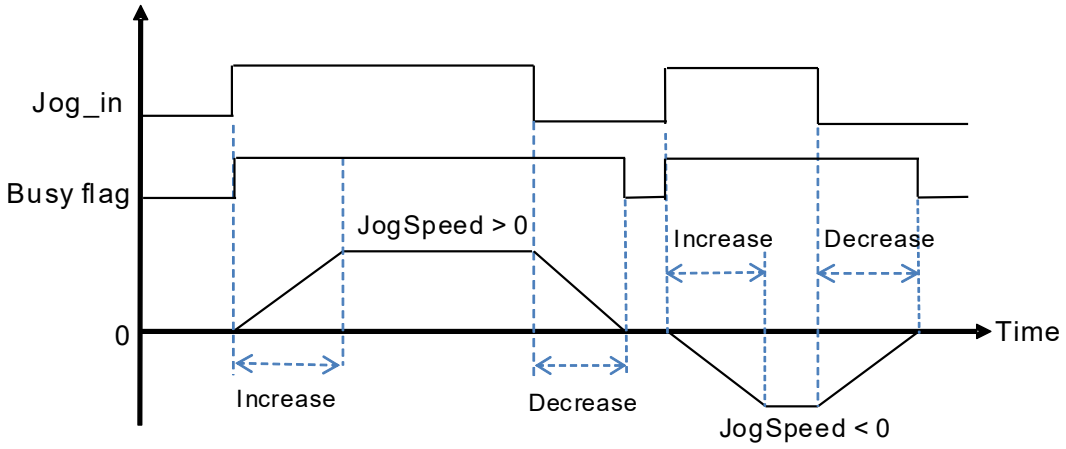
2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **JogSpeed** sets the jog output frequency. The setting value is a signed 32 bit value within the range of -100,000 ~ 100,000 (Hz). When the value is greater than 0, the output will go in the positive direction (and the direction output point is off). When the value is less than 0, the output will go in the negative direction (and the direction output point is on). When the value is 0, the output will stop.
5. If any error occurs as the instruction is in process of the output, the **Error** flag changes to ON. Refer to the error codes that **ErrCode** shows for the trouble shooting.
6. The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

7. See the output timing diagram as below. (Jog_in is the switch to start the instruction and the Busy flag is the **Busy** flag.)



8. After the PUJOG instruction is disabled and the **Busy** flag is off, other output control can be carried out.

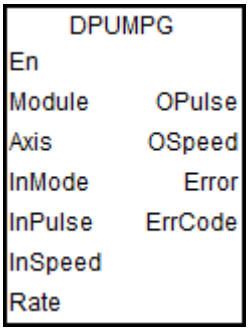
API	Instruction			Operand	Description
1409	D	PUMPG		**Module ~ ErrCode**	PU module MPG output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
Axis								●					○	○		
InMode								●					○	○		
InPulse								●								
InSpeed								●								
Rate								●								○
OPulse								●								
OSpeed								●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
Axis		●			●	●							
InMode		●			●	●							
InPulse			●				●						
InSpeed			●				●						
Rate									●				
OPulse			●				●						
OSpeed			●				●						
Error	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



Module : Module number

Axis : Output axis number

InMode : Encoder input mode and frequency multiplication for counting

InPulse : Number of pulses which have been input

InSpeed : Detected input frequency

Rate : Input/output rate (floating point number)

OPulse : Number of pulses which have been output

OSpeed : Frequency at which pulses are being output

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **Axis** sets the output axis number for the specified PU module. The setting values 1~2 represent the axis1~axis2 output of the specified PU module respectively. If the PU module has no corresponding axis number for output, the error flag **Error** will change to ON.

4. **InMode** sets the input mode of the encoder source and the frequency multiplication for counting.

   See the explanation of **InMode** value in the following table.

High 8-bit value		Low 8-bit value	
Value	Function description	Value	Function description
16#00	A/B phase input	16#00	Fourfold frequency
16#01	A-phase input	16#01	Original frequency
16#02	CW (A) /CCW (B) input	16#02	Doubled frequency

   For example: If the A/B phase input and doubled frequency is used, input the value 16#0002.

   Do not use the values which have not been listed in the above table since they represent the reserved functions.

   For the counting method of A/B phase and CW/CCW, refer to the explanation of HC (the high-speed counter of the PLC). If you choose the single phase input or CW/CCW input, only the original frequency or doubled frequency can be selected. If you enter an incorrect value, the instruction will use the default original frequency.

5. **InPulse** displays the number of already input pulses, which is a signed 32-bit value. Every time the instruction is started, the PU module will automatically clear the value to 0 and then starts counting.

6. **InSpeed** displays the already detected input frequency which is a 32-bit value. The basic time for the frequency detection is 20ms. Therefore, the detected input frequency is 0 if there is no counting value within 20ms. If there is a counting value within 20ms, the output starts at the minimum frequency of 50Hz. Even if **OSpeed** value is lower than 50Hz through the **Rate-**value-based conversion, the output is still conducted at 50Hz.

7. **Rate** is the input / output rate and the value is a floating point number. The number of actual output pulses and frequency are respectively equal to the input pulse number and frequency multiplied by the rate value.

For example: The input frequency is 100Hz and rate is 0.5. So the output frequency is 100x0.5=50Hz. If the maximum output frequency after conversion exceeds 100KHz, the output frequency is limited to 100KHz.

Note: The long-time maximum frequency output may lead to the fact that as the MPG has stopped running, the number of outputs is still increased and the output need keep going until it is complete.

8. **OPulse** shows the number of pulses which have been output. **OSpeed** displays the frequency at which the output is being conducted. They are signed 32-bit values.

9. When the **DPUMPG** instruction is disabled, check the frequency at which the output is being conducted and see if it has reached 0. If the instruction is disabled before the frequency reaches 0, the PU module will stop the output immediately and the output of the pulses which are counted based on the conversion rate will not continue any more.

10. The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1403	There is no such output axis number in the PU module.
16#1404	The output frequency of the PU module is illegal.
16#1405	The output axis specified by the PU module is outputting data. It is not allowed to specify the output repeatedly.

11. When the DPUMPG instruction is enabled or disabled, the PLC will have to notify the module to enable or disable the high-speed counter function. Thus the instruction can not be used with API1410 DPUCNT together. Otherwise it may occur that the two instructions enable or disable the counting of the module with each other.

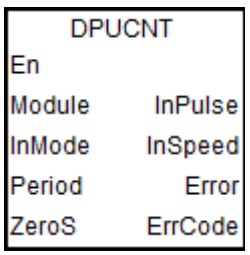
API	Instruction			Operand	Description
1410	D	PUCNT		**Module ~ ErrCode**	High-speed counter function of PU module

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Module								●					○	○		
InMode								●					○	○		
Period								●					○	○		
ZeroS	●	●	●	●				●								
InPulse								●								
InSpeed								●								
Error		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Module		●			●	●							
InMode		●			●	●							
Period		●			●	●							
ZeroS	●												
InPulse			●				●						
InSpeed			●				●						
Error	●												
ErrCode			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



Module : Module number

InMode : Encoder input mode and frequency multiplication for counting

Period : Period time for capturing the frequency

ZeroS : Clear the counter to 0

InPulse : Number of pulses which have been input

InSpeed : Number of pulses per cycle

Error : Error flag

ErrCode : Error code

Explanation

1. This instruction is available for PLC with FW V1.02.00 or later.

2. **Module** sets the serial number of modules at the right of the PLC. The first one is number 1, the second one is

number 2 and so on. Whatever modules at the right of the PLC must be numbered. The maximum number is 8. The instruction is exclusive to the PU modules at the right of the PLC. If the specified module is not a PU module, the error flag **Error** will change to ON.

3. **InMode** sets the input mode of the encoder source and the frequency multiplication for counting.

   See the explanation of **InMode** value in the following table.

High 8-bit value		Low 8-bit value	
Value	Function description	Value	Function description
16#00	A/B phase input	16#00	Fourfold frequency
16#01	A-phase input	16#01	Original frequency
16#02	CW (A) /CCW (B) input	16#02	Doubled frequency

   For example: If the A/B phase input and doubled frequency is used, input the value 16#0002.
   Do not use the values which have not been listed in the above table since they represent the reserved functions.
   For the counting method of A/B phase and CW/CCW, refer to the explanation of HC (the high-speed counter of the PLC). If you choose the single input or CW/CCW input, only the original frequency or doubled frequency can be selected. If you enter an incorrect value, the instruction will use the default original frequency.

4. **Period** is the setting value of a cycle time for capturing the frequency within the range of 10ms ~ 1000ms. If the setting value exceeds the range, the maximum value or minimum value will be automatically taken as the setting value by the PLC.

5. **InPulse** is the number of already input pulses, which is a signed 32-bit value. The counting value is a latched value. If the value need be cleared to 0, just set **ZeroS** from Off to ON while the instruction is running.

6. **InSpeed** displays the counting value for every **Period** time, which is a signed 32-bit value. If you need convert it into the value with the unit of Hz, use the calculation formula for conversion by yourself.

7. The error codes that **ErrCode** shows are listed in the following table.

Error code	Description
16#1400	The module does not support the function.
16#1401	The data stored in the module is illegal or exceeds the allowed range.
16#1402	There is no response from the module; communication timeout occurs.
16#1406	The PU module does not support the counting function.

8. When the DPUCNT instruction is enabled or disabled, the PLC will have to notify the module to enable or disable the high-speed counter function. Thus the instruction can not be used with API1409 DPUMPG together. Otherwise it may occur that the two instructions enable or disable the counting of the module with each other.

6.16 Floating-point Number Instructions

6.16.1 List of Floating-point Number Instructions

The following table lists the Module instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1500	–	FSIN	✓	Sine of a floating-point number
1501	–	FCOS	✓	Cosine of a floating-point number
1502	–	FTAN	✓	Tangent of a floating-point number
1503	–	FES3IN	✓	Arcsine of a floating-point number
1504	–	FACOS	✓	Arccosine of a floating-point number
1505	–	FATAN	✓	Arctangent of a floating-point number
1506	–	FSINH	✓	Hyperbolic sine of a floating-point number
1507	–	FCOSH	✓	Hyperbolic cosine of a floating-point number
1508	–	FTANH	✓	Hyperbolic tangent of a floating-point number
1509	–	FRAD	✓	Converting degrees to radians
1510	–	FDEG	✓	Converting radians to the degrees
1511	SQR	DSQR	✓	Square root of a binary number
1512	–	FSQR	✓	Square root of a floating-point number
1513	–	FEXP	✓	Exponentiation of a floating-point number
1514	–	FLOG	✓	Logarithm of a floating-point number
1515	–	FLN	✓	Natural logarithm of a binary floating-point number
1516	–	FPOW	✓	Power of a floating-point number
1517	RAND	–	✓	Generating a random number

6.16.2 Explanation of Floating-point Number Instructions

API	Instruction code		Operand	Function
1500	FSIN	P	**S, D**	Sine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value

D : Sine value

Explanation

1. This instruction finds the sine of the value in **S** and stores it in **D**. The state of SM695 determines whether the source value in **S** is in radians or degrees.

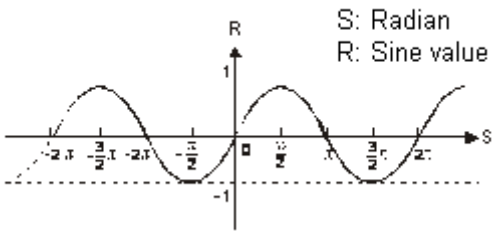
2. If SM695 is OFF, the source value is **S** in radians
   Radian=Degree×π/180.

3. If SM695 is ON, the source value in **S** is in degrees.

   Degree=Radian×180/π. ($0^o \leqq$ Degree $\leqq 360^0$)

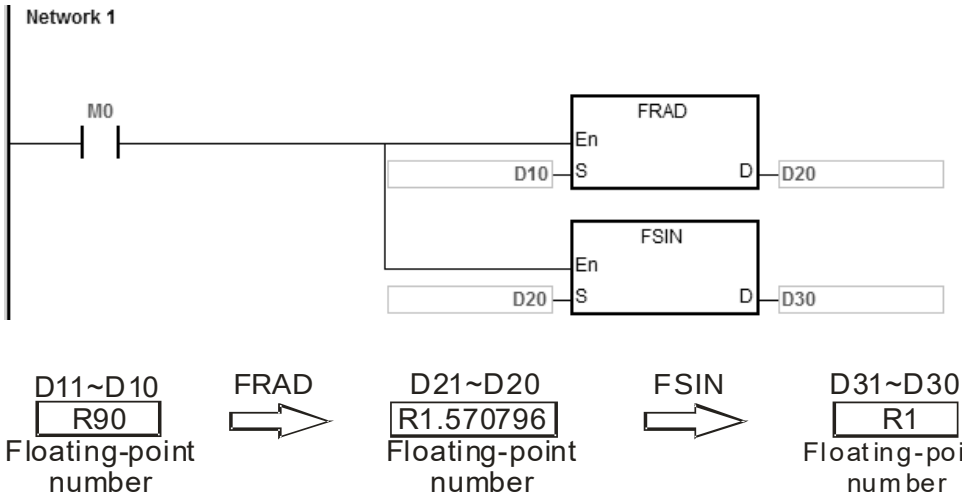
4. If the conversion result is zero, SM600 is ON.

5. The following graph shows the relation between radian and sine values.



Example

When M0 is ON, the FRAD instruction converts a floating-point number in (D11, D10) into radians, and stores the conversion result in (D21, D20). The FSIN instruction finds the sine of the radian value in (D21, D20), and stores it in (D31, D30). The sine value is a floating-point number.



Note: "R" indicates REAL type data.

Additional remarks

1. If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

2. If SM695 is ON, and the value in S is not between 0–360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1501		FCOS	P	**S, D**	Cosine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value

D : Cosine value

Explanation

1. This instruction finds the cosine of the value in **S** and stores it in **D**. The state of SM695 determines whether the source value in **S** is in radians or degrees.

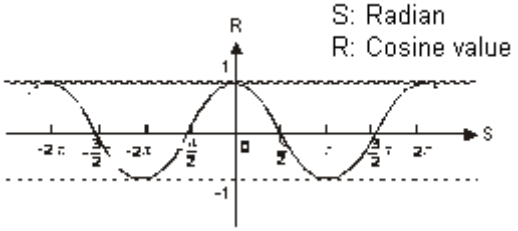
2. If SM695 is OFF, the source value in **S** is in radians.
   Radian=Degree×π/180.

3. If SM695 is ON, the source value in **S** is in degrees.

   Degree=Radian×180/π. ($0^{o} \leqq$ Degree $\leqq 360^{0}$)

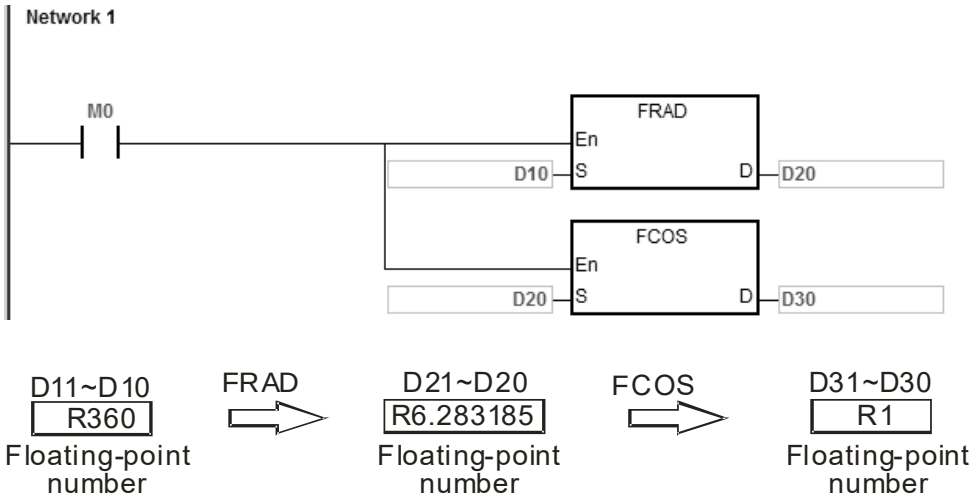
4. If the conversion result is zero, SM600 is ON.

5. The following graph shows the relation between radians and cosine values.



Example

When M0 is ON, the FRAD instruction converts a floating-point number in (D11, D10) into radians, and stores the conversion result in (D21, D20). The FCOS instruction finds the cosine of the radian value in (D21, D20) and stores it in (D31, D30). The cosine value is a floating-point number.



Note: "R" indicates REAL type data.

Additional remarks

1. If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.
2. If SM695 is ON, and the value in S is not between 1–360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1502		FTAN	P	**S, D**	Tangent of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value

D : Tangent value

Explanation

1. This instruction finds the tangent of the value in **S** and stores it in **D**. The state of SM695 determines whether the source value in **S** is in radians or in degrees.

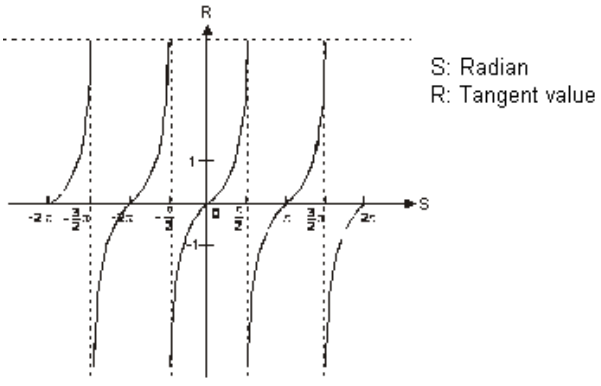
2. If SM695 is OFF, the source value in **S** is in radians.
   Radian=Degree×π/180.

3. If SM695 is ON, the source value in **S** is in degrees.

   Degree=Radian×180/π. ($0^o \leqq$ Degree $\leqq 360^0$)

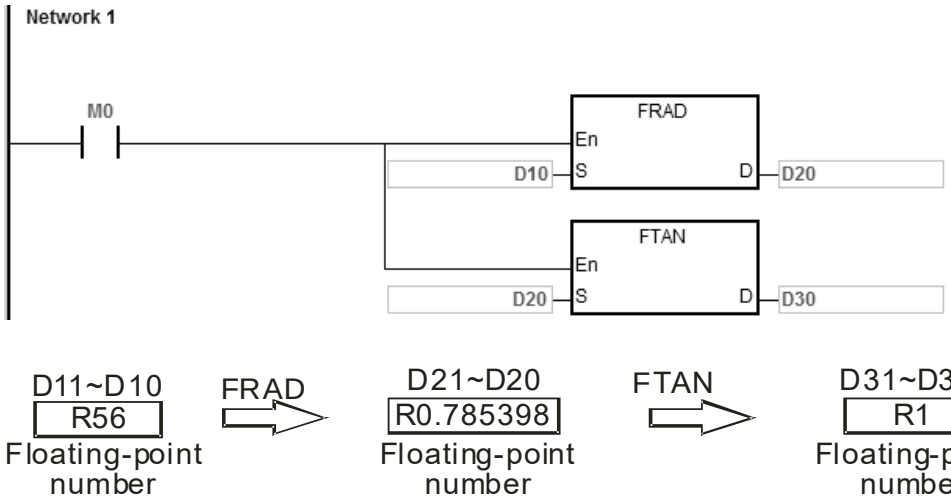
4. If the conversion result is zero, SM600 is ON.

5. The following graph shows the relation between radians and tangent values.



Example

When M0 is ON, the FRAD instruction converts the floating-point number in (D11, D10) into radians, and stores the conversion result in (D21, D20). The FTAN instruction finds the tangent of the radian value in (D21, D20) and stores it in (D31, D30). The tangent value is a floating-point number.



Note: "R" indicates REAL type data.

Additional remarks

1. If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.
2. If SM695 is ON, and the value in S is not between 0–360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

API	Instruction code			Operand	Function
1503		FES3IN	P	**S, D**	Arcsine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

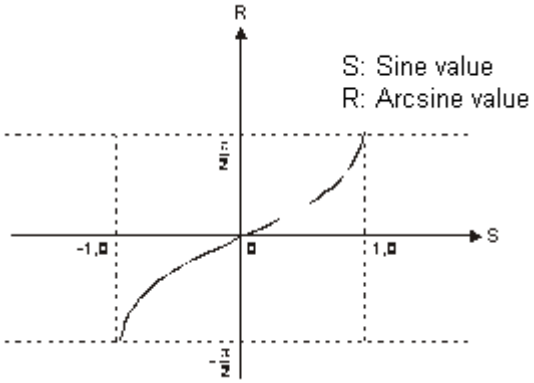
S : Source value

D : Arcsine value

Explanation

1. This instruction finds the arcsine of the value in **S** and stores it in **D**. Arcsine value=$\sin^{-1}$

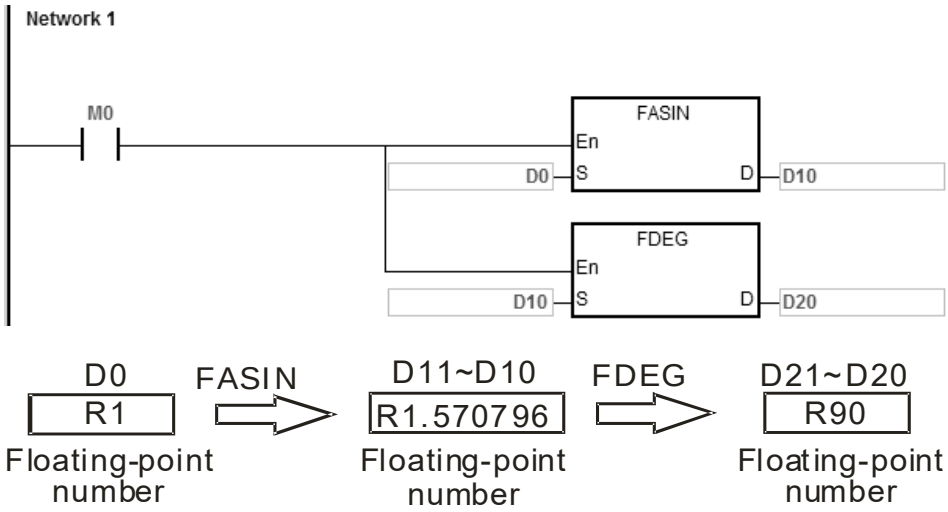
   The following graph shows the relation between sine and arcsine values.



2. If the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the instruction finds the arcsine of the floating-point number in (D1, D0) and stores it in (D11, D10). The FDEG instruction converts the arcsine value in (D11, D10) into degrees, and stores the conversion result in (D21, D20).



Note: "R" indicates REAL type data.

Additional remarks

1. The floating-point number specified by the operand **S** must be between –1.0 to 1.0. If the floating-point number is not in that range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

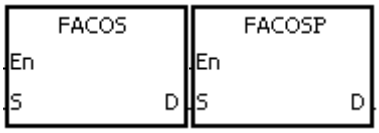
API	Instruction code			Operand	Function
1504		FACOS	P	**S, D**	Arccosine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



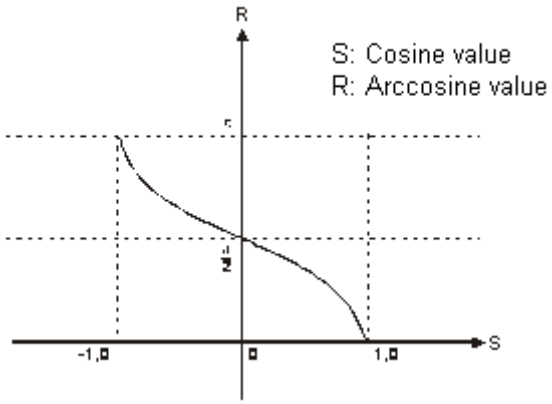
S : Source value

D : Arccosine value

Explanation

1. This instruction finds the arccosine of the value in **S** and stores it in **D**. Arccosine value=$\cos^{-1}$

   The following graph shows the relation between cosine and arccosine values.



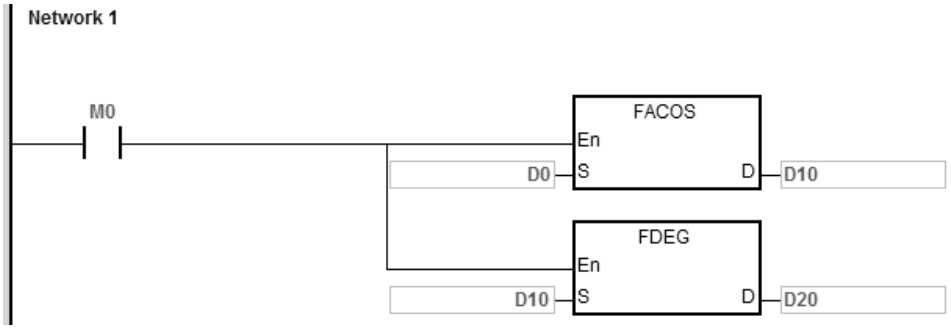
2. If the absolute value of the conversion result is larger than the value that can be represented by the maximum floating-point number, SM602 is ON.

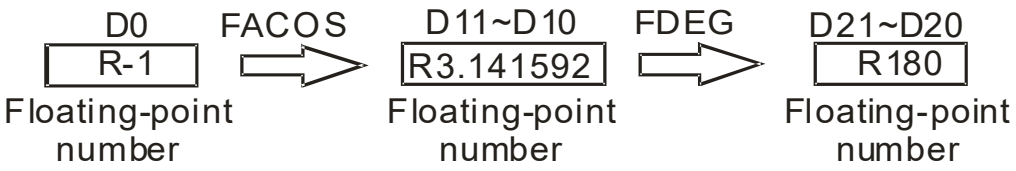
3. If the absolute value of the conversion result is less than the value that can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the FACOS instruction finds the arccosine of the floating-point number in (D1, D0) and stores it in (D11, D10). The FDEG instruction converts the arccosine value in (D11, D10) into degrees, and stores the conversion result in (D21, D20).





Note: "R" indicates REAL type data.

Additional remarks

1. The floating-point number specified by the operand **S** must be between –1.0 to 1.0. If the floating-point number is not in the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

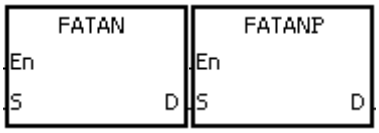
API	Instruction code			Operand	Function
1505		FATAN	P	**S, D**	Arctangent of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



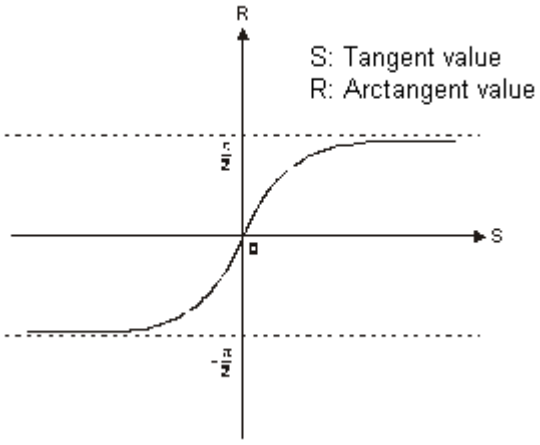
S : Source value

D : Arctangent value

Explanation

1. This instruction finds the arctangent of the value in **S** and stores it in **D**. Arctangent value=$\tan^{-1}$

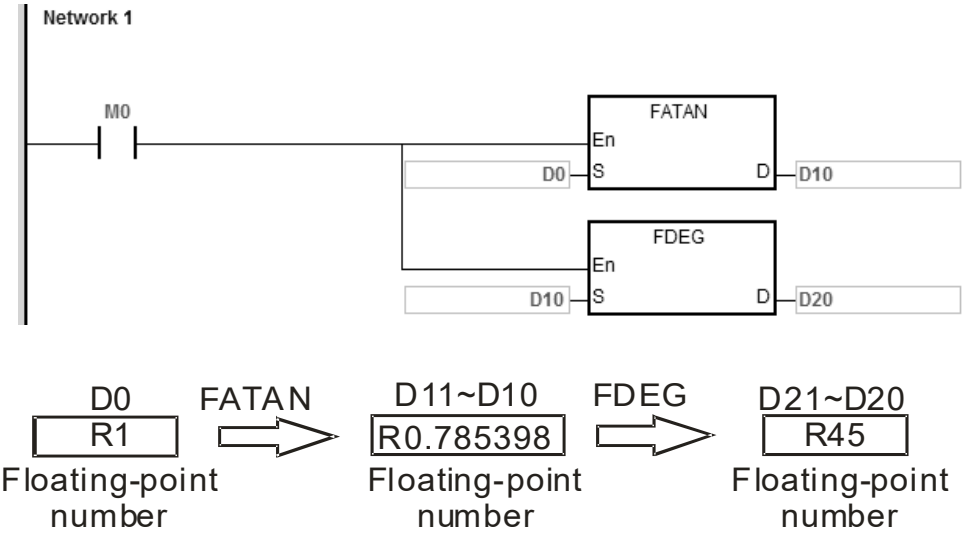
2. The following graph shows the relation between tangent and arctangent values.



3. If the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the FATAN instruction finds the arctangent of the floating-point number in (D1, D0) and stores it in (D11, D10). The FDEG instruction converts the arctangent value in (D11, D10) is converted into degrees, and stores the conversion result in (D21, D20).



Note: "R" indicates REAL type data.

Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1506		FSINH	P	**S, D**	Hyperbolic sine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

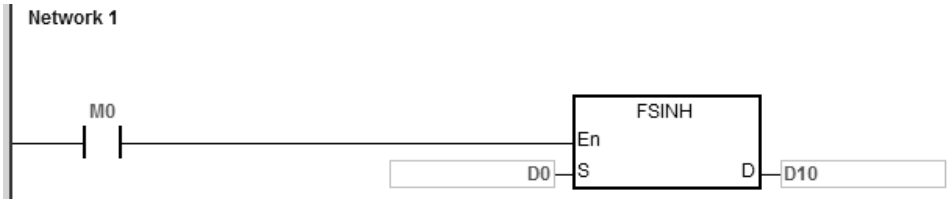
Symbol

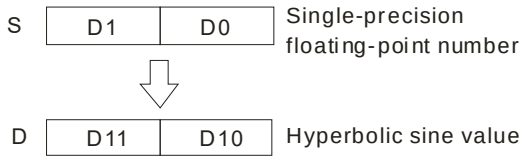
Explanation

1. This instruction finds the hyperbolic sine of the value in **S** and stores it in **D**. Hyperbolic sine value=$(e^{s}-e^{-s})/2$.
2. If the absolute value of the conversion result is larger than the value that can be represented by floating-point numbers, the value in **D** is 16#7F800000, and SM602 is ON.
3. If the absolute value of the conversion result is less than the value that can be represented by floating-point numbers, the value in **D** is 16#FF800000, and SM601 is ON.
4. If the conversion result is zero, SM600 is ON.

Example

1. When M0 is ON, the instruction finds the hyperbolic sine of the floating-point number in (D1, D0) and stores it in (D11, D10). The hyperbolic sine value in (D11, D10) is a floating-point number.





2. If the absolute value of the conversion result is larger than the value that can be represented by floating-point numbers, SM602 is ON.

3. If the absolute value of the conversion result is less than the value that can be represented by floating-point numbers, SM601 is ON.

4. If the conversion result is zero, SM600 is ON.

Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

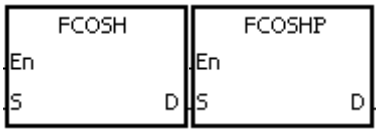
API	Instruction code			Operand	Function
1507		FCOSH	P	**S, D**	Hyperbolic cosine of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	●	●			●	●	●	●	●		○					○
D		●			●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S : Source value

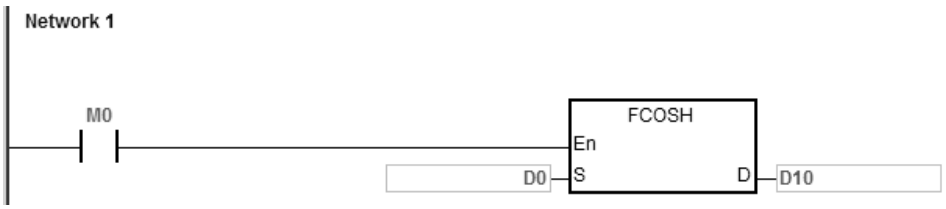
D : Hyperbolic cosine value

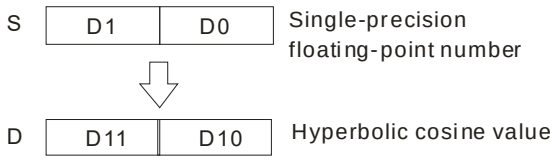
Explanation

1. This instruction finds the hyperbolic cosine of the value in **S** and stores it in **D**. Hyperbolic cosine value=$(e^{s}+e^{-s})/2$.
2. If the absolute value of the conversion result is larger than the value that can be represented by floating-point numbers, the value in **D** is 16#7F800000, and SM602 is ON.
3. If the absolute value of the conversion result is less than the value that can be represented by floating-point numbers, the value in **D** is 16#FF800000, and SM601 is ON.
4. If the conversion result is zero, SM600 is ON.

Example

1. When M0 is ON, the instruction finds the hyperbolic cosine of the floating-point number in (D1, D0) and stores it in (D11, D10). The hyperbolic cosine value in (D11, D10) is a floating-point number.





2. If the absolute value of the conversion result is larger than the value that can be represented by floating-point numbers, SM602 is ON.

3. If the absolute value of the conversion result is less than the value that can be represented by floating-point numbers, SM601 is ON.

4. If the conversion result is zero, SM600 is ON.

Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

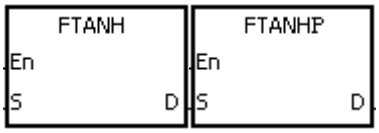
API	Instruction code			Operand	Function
1508		FTANH	P	**S, D**	Hyperbolic tangent of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S : Source value

D : Hyperbolic tangent value

Explanation

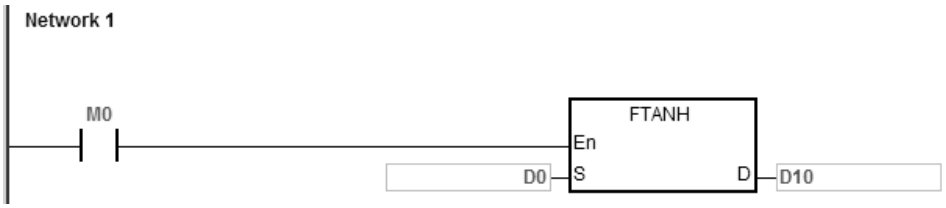
1. This instruction finds the hyperbolic tangent of the value in **S** and stores it in **D**.

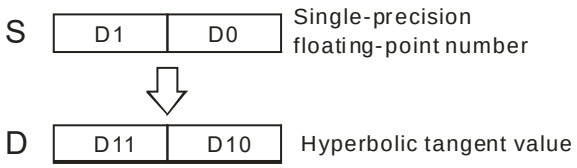
   Hyperbolic tangent value=$(e^{s}-e^{-s})/(e^{s}+e^{-s})$.

2. If the conversion result is 0, SM600 is ON.

Example

1. When M0 is ON, the instruction finds the hyperbolic tangent of the floating-point number in (D1, D0) and stores it in (D11, D10). The hyperbolic tangent value in (D11, D10) is a floating-point number.





2. If the conversion result is zero, SM600 is ON.

Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1509		FRAD	P	**S, D**	Converting degrees to radians

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value (in degrees)

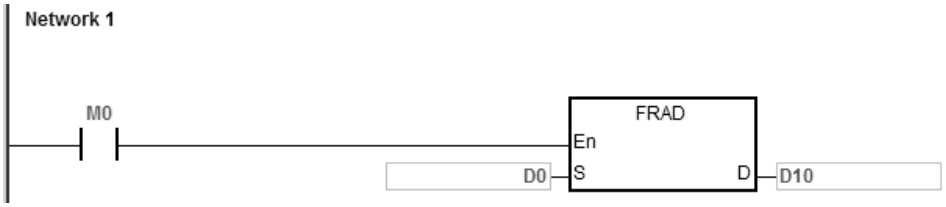
D : Conversion result (in radians)

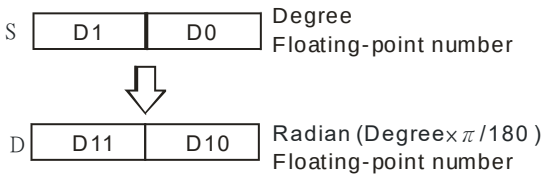
Explanation

1. This instruction converts the degrees value in **S** into radians, and stores it in **D**.
2. Radian = Degree×(π/180).
3. If the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the instruction converts the degree value in (D1, D0) to the radians value, and stores the conversion result in (D11, D10). The radian in (D11, D10) is a floating-point number.





Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1510		FDEG	P	**S, D**	Converting radians to degrees

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value (in radians)

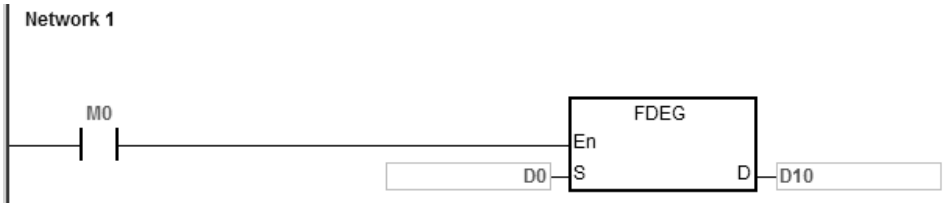
D : Conversion result (in degrees)

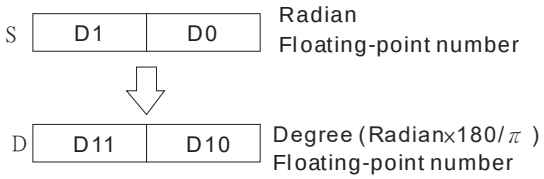
Explanation

1. This instruction converts the radians value in **S** to degrees, and stores it in **D**.
2. Degree = Radian×(180/π).
3. If the absolute value of the conversion result is large**r** than the value that can be represented by floating-point numbers, the value in **D** is 16#7F7FFFFF.
4. If the absolute value of the conversion result is less than the value that can be represented by floating-point numbers, the value in **D** is 16#7F7FFFFF.
5. If the conversion result is zero, SM600 is ON.

Example

When M0 is ON, the instruction converts the radians values in (D1, D0) to the degree value, and stores the conversion result in (D11, D10). The degree in (D11, D10) is a floating-point number.





Additional remarks

If the value in **S** exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
1511	D	SQR	P	**S, D**	Square root of a binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●				●	●	
		●	●		●	●	●				●	●	

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

S : Source value

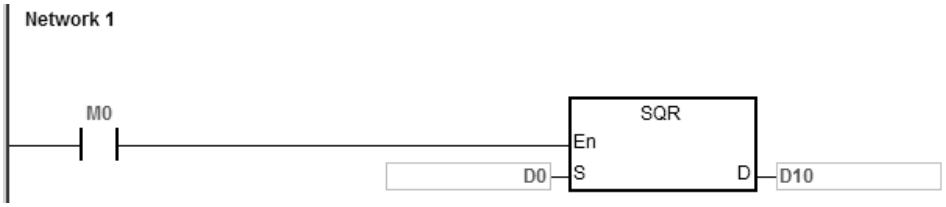
D : Device where the result is stored

Explanation

1. This instruction calculates the square root of the value in **S**, and stores the result in the device specified by **D**.
2. The operation result stored in **D** is an integer. If a floating-point number is rounded down to the nearest whole digit, SM601 is ON.
3. If the operation result stored in **D** is 0, SM600 is ON.
4. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example

When M0 is ON, the instruction calculates the square root of the value in D0, and stores the result in D10.



Additional remarks

The value in **S** only can be a positive value. If the value in **S** is a negative value, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1512		FSQR	P	**S, D**	Square root of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value

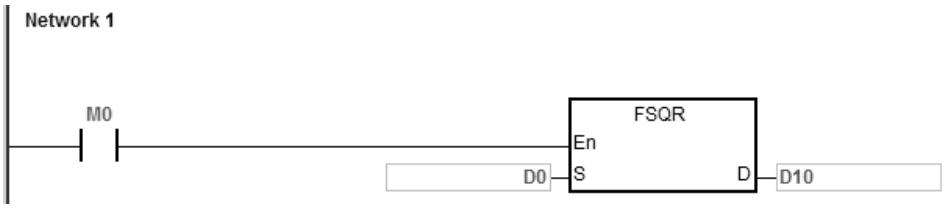
D : Device where the result is stored

Explanation

1. This instruction calculates the square root of the floating-point number in **S**, and stores the result in the device specified by **D**.
2. If the operation result stored in **D** is 0, SM600 is ON.

Example 1

When M0 is ON, the instruction calculates the square root of the floating-point number in (D1, D0), and stores the result in (D11, D10).



Additional remarks

The value in **S** only can be a positive value. If the value in **S** is a negative value, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1513		FEXP	P	**S, D**	Exponentiation of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value (exponent)

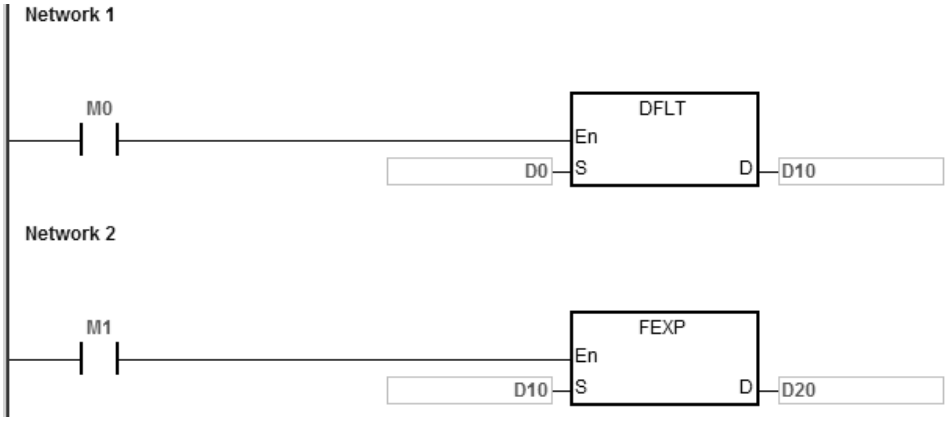
D : Device where the operation result is stored

Explanation

1. This function calculates the value of the number e raised to the power in **S.** Exponentiation involves two numbers, the base e which represents 2.71828, and the exponent in **S**.
2. EXP[**D**+1, **D**]=[**S**+1, **S**].
3. The number in **S** can be a positive number or a negative number. The device specified by **D** should be a 32-bit register, and the number in the device specified by **S** should be a floating-point number.
4. The value in the register specified by **D** is e^{S} (*e* is 2.71828, and **S** represents the source value).
5. If the absolute value of the conversion result is larger than the value that can be represented by floating-point numbers, the value in the register specified by **D** is 16#7F800000, and SM602 is ON.
6. If the operation result stored in **D** is 0, SM600 is ON.

Example

1. When M0 is ON, the DFLT instruction converts the value in (D1, D0) into a floating-point number, and stores the conversion result in (D11, D10).
2. When M1 is ON, the FEXP instruction performs the exponentiation with the value in (D11, D10), and stores the floating-point number result in (D21, D20).


Network 1
M0
DFLT
En
D0
S
D
D10
Network 2
M1
FEXP
En
D10
S
D
D20

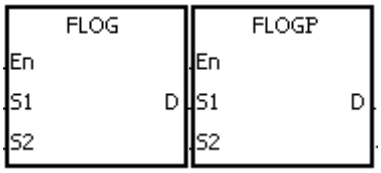
API	Instruction code			Operand	Function
1514		FLOG	P	**S_1, S_2, D**	Logarithm of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○					○
S_2					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Base for the logarithm

S_2 : Source value

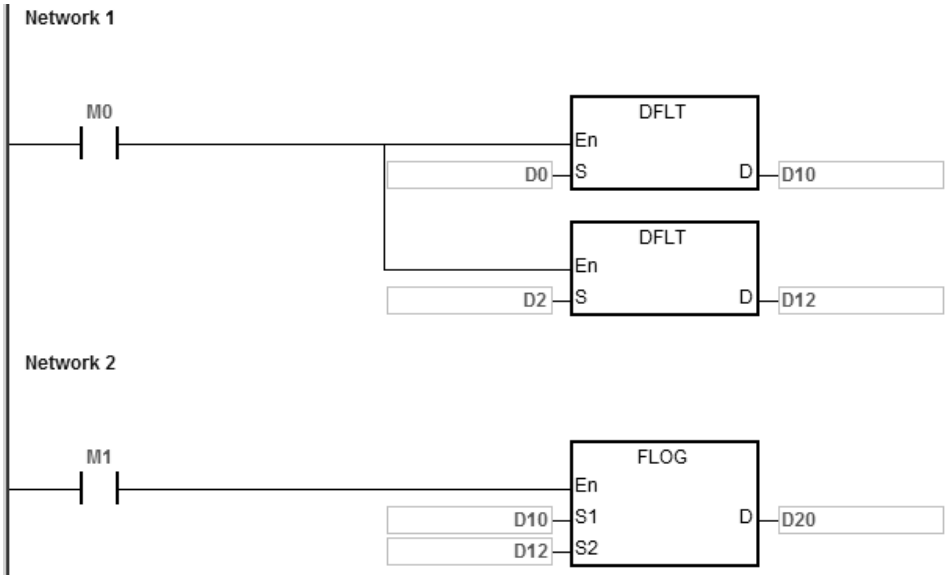
D : Device where the operation result is stored

Explanation

1. This instruction calculates the logarithm of the value in **S_2** with respect to the value in **S_1**, and stores the single-precision floating-point operation result in **D**.
2. The values in **S_1** and **S_2** only can be positive values.
3. **S_1^D=S_2.**→**D**=Log$_{S_1}S_2$.
4. Example: suppose the values in **S_1** and **S_2** are 5 and 125 respectively. Find $\log_5 125$ (log base 5 of the number 125).
5. **S_1^D=S_2.**→5**D**=125.→**D**=$\log_5 125$=3.
6. If the operation result stored in **D** is 0, SM600 is ON.

Example

1. When M0 is ON, the DFLT instruction converts the values in (D1, D0) and (D3, D2) into the floating-point numbers, and stores the conversion results in (D11, D10) and (D13, D12) respectively.
2. When M1 is ON, the FLOG instruction calculates the logarithm of the floating-point number in (D13, D12) with respect to the floating-point number in (D11, D10), and stores the operation result in (D21, D20).



Additional remarks

If the value in S_1 is less than or equal to 1, or if the value in S_2 is less or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
1515		FLN	P	**S, D**	Natural logarithm of a binary floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○					○
D					●	●	●	●			○					

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol

S : Source value

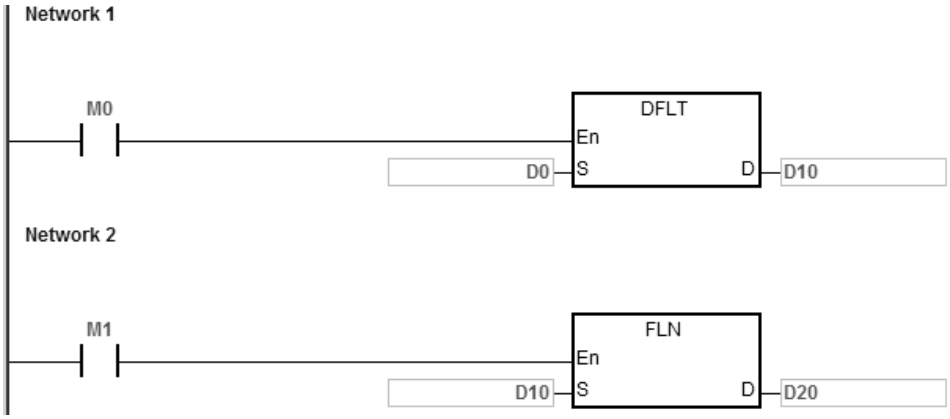
D : Device where the operation result is stored

Explanation

1. This instruction calculates the natural logarithm of the operand **S** in a single-precision floating-point operation.
2. The value in **S** only can be a positive value.
3. e^{D}=**S**.→The value in **D**=ln**S**.
4. If the operation result stored in **D** is 0, SM600 is ON.

Example

1. When M0 is ON, the DFLT instruction converts the value in (D1, D0) into the floating-point number, and stores the conversion result in (D11, D10).
2. When M1 is ON, the FLN instruction calculates the natural logarithm of the floating-point number in (D11, D10), and stores the operation result in (D21, D20).



Additional remarks

If the value in S is less than or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

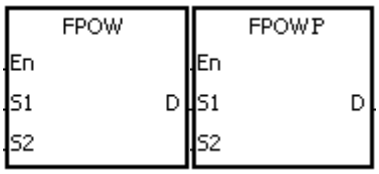
API	Instruction code			Operand	Function
1516		FPOW	P	**S_1, S_2, D**	Power of a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●							○
S_2					●	●	●	●	●							○
D					●	●	●	●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2									●				
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : Source value

S_2 : Exponent value

D : Device where the operation result is stored

Explanation

1. This instruction raises the single-precision floating-point number in **S_1** to the power of the value in **S_2**, and stores the single-precision floating-point operation result in **D**.

2. **D**=POW[**S_1**+1 · **S_1**]^[**S_2**+1 · **S_2**]

3. The value in **S_1** only can be a positive value, but the value in **S_2** can be a positive value or a negative value.

4. Suppose the values in **S_1** and **S_2** are 5 and 3 respectively: $D=5^3=125$.

5. If the absolute value of the operation result is larger than the value that can be represented by floating-point numbers, the value in **D** is 16#7F7FFFFF, and SM602 is ON.

6. If the absolute value of the operation result is less than the value that can be represented by floating-point numbers, the value in **D** is 16#FF800000, and SM601 is ON.

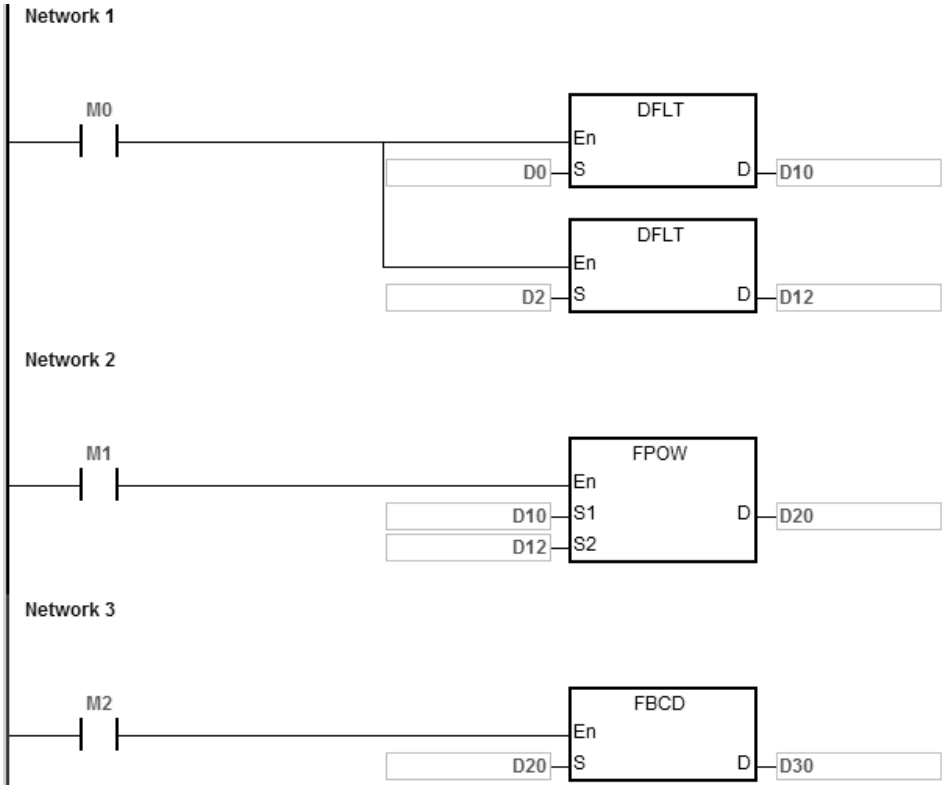
7. If the operation result stored in **D** is 0, SM600 is ON.

Example

1. When M0 is ON, the DFLT instruction converts the values in (D1, D0) and (D3, D2) into floating-point numbers, and stores the conversion results in (D11, D10) and (D13, D12) respectively.

2. When M1 is ON, the FPOW instruction raises the floating-point number in (D11, D10) to the power of the floating-point number in (D13, D12), and stores the operation result in (D21, D20).

3. When M2 is ON, the FBCD instruction converts the binary floating-point number in (D21, D20) into the binary-coded decimal floating-point number, and stores the conversion result in (D31, D30).



Additional remarks

If the value in S_1 is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

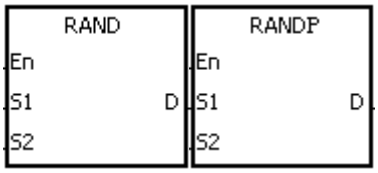
API	Instruction code			Operand	Function
1517		RAND	P	**S_1, S_2, D**	Generating a random number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○	○	○	○		
S_2					●	●		●	●		○	○	○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Minimum value

S_2 : Maximum value

D : Device where the result is stored

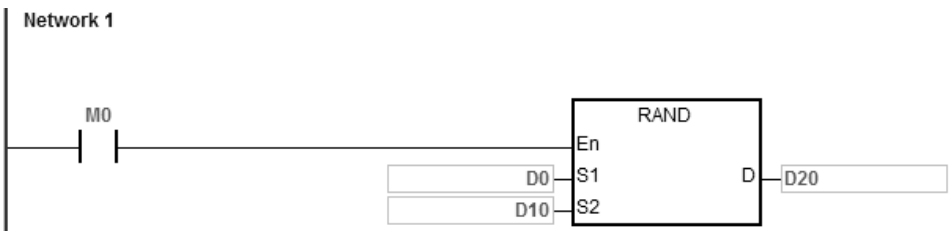
Explanation

1. This instruction generates a random number between the minimum value in **S_1** and the maximum value in **S_2**, and then stores the result in **D**.

2. If the value in **S_1** is larger than the value in **S_2**, the instruction takes the values in **S_1** and **S_2** as the maximum value and the minimum value respectively.

Example

When M0 is ON, the instruction generates a random number between the minimum value in D0 and the maximum value in D10, and stores the result in D20.



Additional remarks

The values in **S_1** and **S_2** must be between 0–32767. If the value in **S_1** or **S_2** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.17 Real-time Clock Instructions

6.17.1 List of Real-time Clock Instructions

The following table lists the Real-time Clock instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1600	TRD	–	✓	Reading the time
1601	TWR	–	✓	Writing the time
1602	T+	–	✓	Adding the time
1603	T-	–	✓	Subtracting the time
1604	HOUR	–	–	Running-time meter
1605	TCMP	–	✓	Comparing the time
1606	TZCP	–	✓	Time zone comparison
1607	DST	–	✓	Daylight saving time
1608	WWON	–	–	Setting up weekly working time setup

6.17.2 Explanation of Real-time Clock Instructions

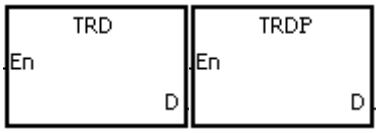
API	Instruction code			Operand	Function
1600		TRD	P	**D**	Reading the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



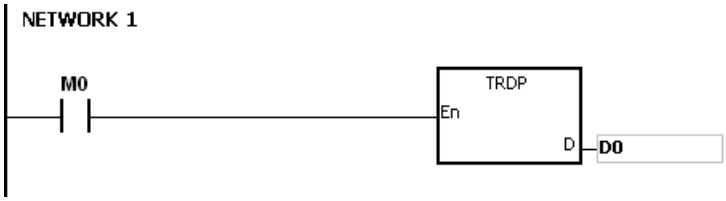
D : Device where the result is stored

Explanation

1. This instruction reads the current time from the built-in real time clock in the CPU module, and stores the current time in **D**.
2. The operand **D** occupies seven consecutive devices.
3. The built-in real-time clock provides the year, the week, the moth, the day, the minute, and the second. The data is stored in SR391–SR397.
4. The last two digits of the year number for A.D. are stored in SR391.

Example

When M0 is ON, the instruction reads the current time from the real-time clock into D0–D6. The value 1 in SR397 represents Monday, the value 2 represents Tuesday, and this continues to the value 7 represents Sunday.



Special data register	Item	Value		General data register	Item
SR391	Year (A.D.)	00-99	→	D0	Year (A.D.)
SR392	Month	1-12	→	D1	Month
SR393	Day	1-31	→	D2	Day
SR394	Hour	0-23	→	D3	Hour
SR395	Minute	0-59	→	D4	Minute
SR396	Second	0-59	→	D5	Second
SR397	Week	1-7	→	D6	Week

Additional remarks

1. If **D**+6 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. When SM220 is ON, the real-time clock is calibrated within ±30 seconds. If the value of the seconds read from the real-time clock is between 0–29, the instruction clears the seconds value to zero. If the value of the seconds read from the real-time clock is between 30–59, the instruction increments the value of the minute by one, and clears the seconds value to zero.

API	Instruction code			Operand	Function
1601		TWR	P	**S**	Writing the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●							

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Data source

Explanation

1. This instruction adjusts the built-in real-time clock in the CPU module by writing the correct current time in **S** into the built-in real-time clock.
2. The operand **S** occupies seven consecutive devices. (Use 24 hour time format)
3. The instruction instantly writes the new setting time into the real-time clock in the PLC.
4. Make sure that when the instruction executes, the new setting time in **S** is consistent with the actual time.
5. It is suggested to use it as a pulse instruction. If the contact is normally open, the instruction is executed to write the time constantly. But the PLC only writes the time at the first scan. If the built-in real-time clock needs to be updated, you can close the contact for a scan time and then execute this instruction again to update the clock.

Example

When M0 is ON, the instruction writes the correct current time into the built-in real-time clock in the PLC.

New setting time

General data register	Item	Value		Special data register	Item
D20	Year (A.D.)	00-99	→	SR391	Year (A.D.)
D21	Month	1-12	→	SR392	Month
D22	Day	1-31	→	SR393	Day
D23	Hour	0-23	→	SR394	Hour
D24	Minute	0-59	→	SR395	Minute
D25	Second	0-59	→	SR396	Second
D26	Week	1-7	→	SR397	Week

Real time clock

Additional remarks

1. If the value in **S** exceeds the range, an operation error occurs, the instruction is not executed, SM is ON, and the error code in SR is 16#2003.
2. If **S**+6 exceeds the device range, an operation error occurs, the instruction is not executed, SM is ON, and the error code in SR is 16#2003.
3. If you declare the operand **S** in ISPSoft, the data type is ARRAY [7] of WORD/INT.

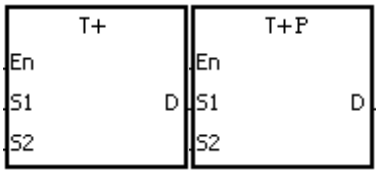
API	Instruction code			Operand	Function
1602		T+	P	**S_1, S_2, D**	Adding the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○					
S_2					●	●		●	●		○					
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Source device 1

S_2 : Source device 2

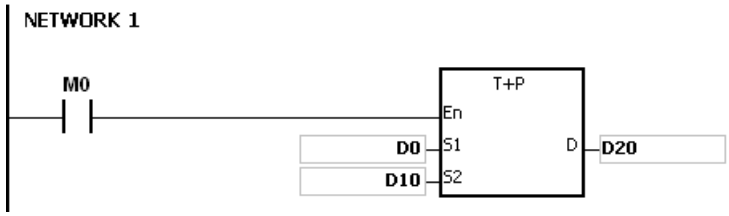
D : Device where the result is stored

Explanation

1. This instruction adds the value of the hour, the minute, and the second in the real-time clock specified by **S_2** to the value of the hour, the minute, and the second in the real-time clock specified by **S_1**, and then stores the sum in the register specified by **D**.
2. The operands **S_1**, **S_2**, and **D** each occupy three consecutive devices.
3. If the sum is larger than or equal to 24 hours, SM602 is ON, and the instruction subtracts 24 hours from the sum before storing the result in **D**.
4. If the sum is zero (zero hour zero minute zero second), SM600 is ON.

Example

When M0 is ON, the instruction adds the value of the hour, the minute, and the second in D10–D12 to the value of the hour, the minute, and the second in D0–D2, and stores the sum in D20–D22.



D0	8 (Hour)		D10	6 (Hour)		D20	14 (Hour)
D1	10 (Minute)	+	D11	40 (Minute)	→	D21	50 (Minute)
D2	20 (Second)		D12	6 (Second)		D22	26 (Second)

8 hour 10 minute 20 second 6 hour 40 minute 6 second 14 hour 50 minute 26 second

Additional remarks

1. If the value in S_1 or S_2 exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If S_1+2, S_2+2, or **D**+2 exceeds the device range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If you declare the operand S_1 in ISPSoft, the data type is ARRAY [3] of WORD/INT.
4. If you declare the operand S_2 in ISPSoft, the data type is ARRAY [3] of WORD/IN.
5. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of WORD/INT.

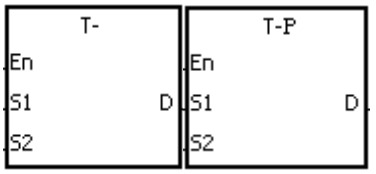
API	Instruction code			Operand	Function
1603		T-	P	**S_1, S_2, D**	Subtracting the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○					
S_2					●	●		●	●		○					
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Source device 1

S_2 : Source device 2

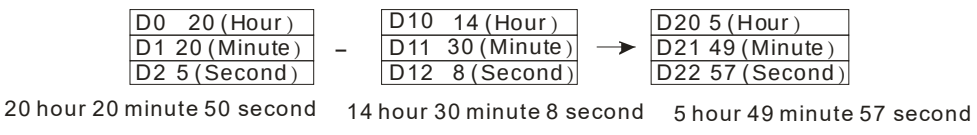
D : Device where the result is stored

Explanation

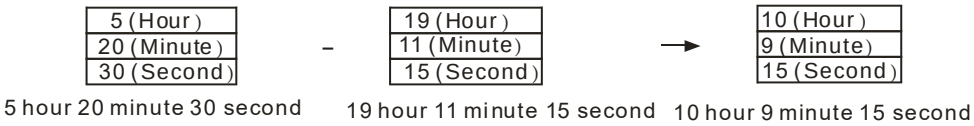
1. This instruction subtracts the value of the hour, the minute, and the second in the real-time clock specified by **S_2** from the value of the hour, the minute, and the second in the real-time clock specified by **S_1**, and stores the difference in the register specified by **D**.
2. The operands **S_1**, **S_2**, and **D** all occupy three consecutive devices.
3. If the difference is a negative, SM601 is ON, and the instruction adds 24 hours to the difference and then stores the result in **D**.
4. If the difference is zero (zero hour zero minute zero second), SM600 is ON.

Example

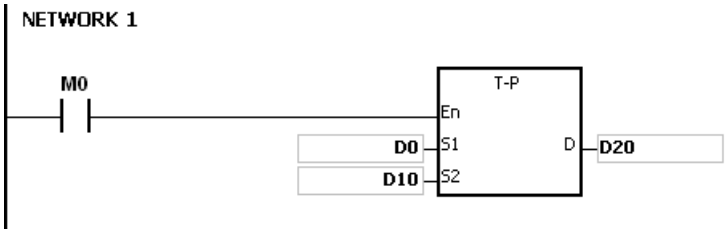
1. When M0 is ON, the instruction subtracts the value of the hour, the minute, and the second in D10–D12 from the value of the hour, the minute, and the second in D0–D2, and stores the difference in D20–D22.



2. If the difference is a negative, SM601 is ON.



5 hour 20 minute 30 second　19 hour 11 minute 15 second　10 hour 9 minute 15 second



Additional remarks

1. If the value in S_1 or S_2 exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If S_1+2, S_2+2, or **D**+2 exceeds the device range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If you declare the operand S_1 in ISPSoft, the data type is ARRAY [3] of WORD/INT.

4. If you declare the operand S_2 in ISPSoft, the data type is ARRAY [3] of WORD/INT.

5. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of WORD/INT.

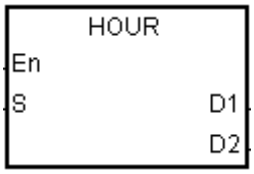
API	Instruction code	Operand	Function
1604	HOUR	**S, D_1, D_2**	Running-time meter

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		
D_1								●								
D_2		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D_1		●			●	●							
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	

Symbol



S : Time after which the output device is ON

D_1 : Current time

D_2 : Output device

Explanation

1. This instruction switches the output device specified by **D_2** to ON after the amount of time in **S**.

2. **S**: The time after which the output device is ON (Unit: Hour)

   The operand **S** used in the 16-bit instruction must be between 1–32,767.

3. **D_1**: The current time (Unit: Hour). The value in **D_1** must be between 0–32,767.

   D_1+1: The current time which is less than one hour (Unit: Second). The value in **D_1**+1 should be between 0–3,599.

   D_1+2 is for system use only. The value in it cannot be altered when the instruction is executed; otherwise, an error occurs.

   When the current time is 32,767 hour 3,599 second, the timer stops counting. After the values in **D_1** and **D_1**+1 are cleared to 0, the timer starts to count again.

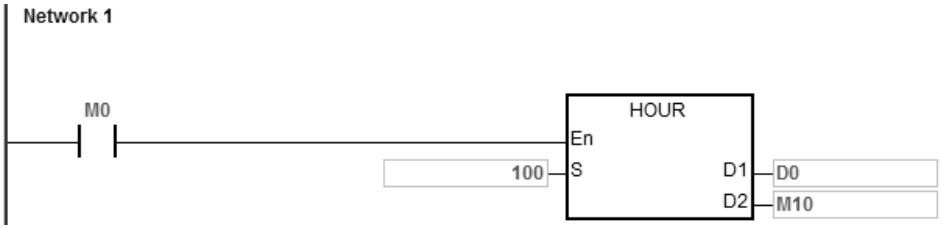
4. When the time for which the input contact has been ON reaches the setting time in **S**, the output device is ON. Before that the output device is not ON. This function allows you to manage the running time of the machine and maintenance.

5. After the output device is ON, the timer continues to count.

6. When using on-line editing, reset the conditional contact to initialize the instruction.

Example

When M0 is ON, the instruction timer starts to count. When the time for which M0 has been ON reaches 100 hours, M10 is ON. The current time is recorded in D0, and the current time which is less than one hour is recorded in D1. D2 is for system use. The value in it cannot be altered; otherwise, an error occurs.



Additional remarks

1. When **S** is less than or equal to 0, the instruction is not executed, and the state of the output device is unchanged.
2. If the value in $\mathbf{D_1}$ is less than **0**, the state of the output device is unchanged.
3. If $\mathbf{D_1}$+2 exceeds the device range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If you declare the operand $\mathbf{D_1}$ in ISPSoft, the data type is ARRAY [3] of WORD/INT.

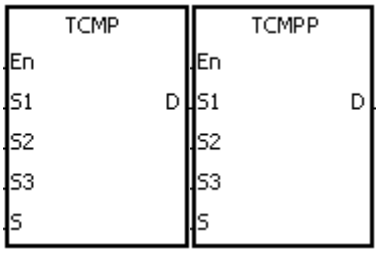
API	Instruction code			Operand	Function
1605		TCMP	P	**S_1, S_2, S_3, S, D**	Comparing the time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○	○	○	○		
S_2					●	●		●	●		○	○	○	○		
S_3					●	●		●	●		○	○	○	○		
S					●	●		●								
D		●	●	●				●		○						

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Hour for the setting time

S_2 : Minute for the setting time

S_3 : Second for the setting time

S : Current time

D : Comparison result

Explanation

1. This instruction compares the value of the hour, the minute, and the second specified by **S_1**–**S_3** with the value of the hour, the minute, and the second in the devices starting from the device specified by **S**, and stores the comparison result in **D**.

2. The hour of the current time is in the device specified by **S**, and the value of the hour between 0–23. The minute of the current time is in the device specified by **S**+1, and the value of the minute must be between 0–59. The second of the current time is in the device specified by **S**+2, and the value of the second must be between 0–59.

3. The operand **D** occupies three consecutive devices. The comparison result is stored in **D**, **D**+1, and **D**+2.

4. In general, use the TRD instruction (API 1600) to read the current time from the real-time clock first, and then use the TCMP instruction to compare the time.

5. If the setting time in **S_1**-**S_3** is larger than the current time in **S**, **D** is ON, **D**+1 is OFF, and **D**+2 is OFF.

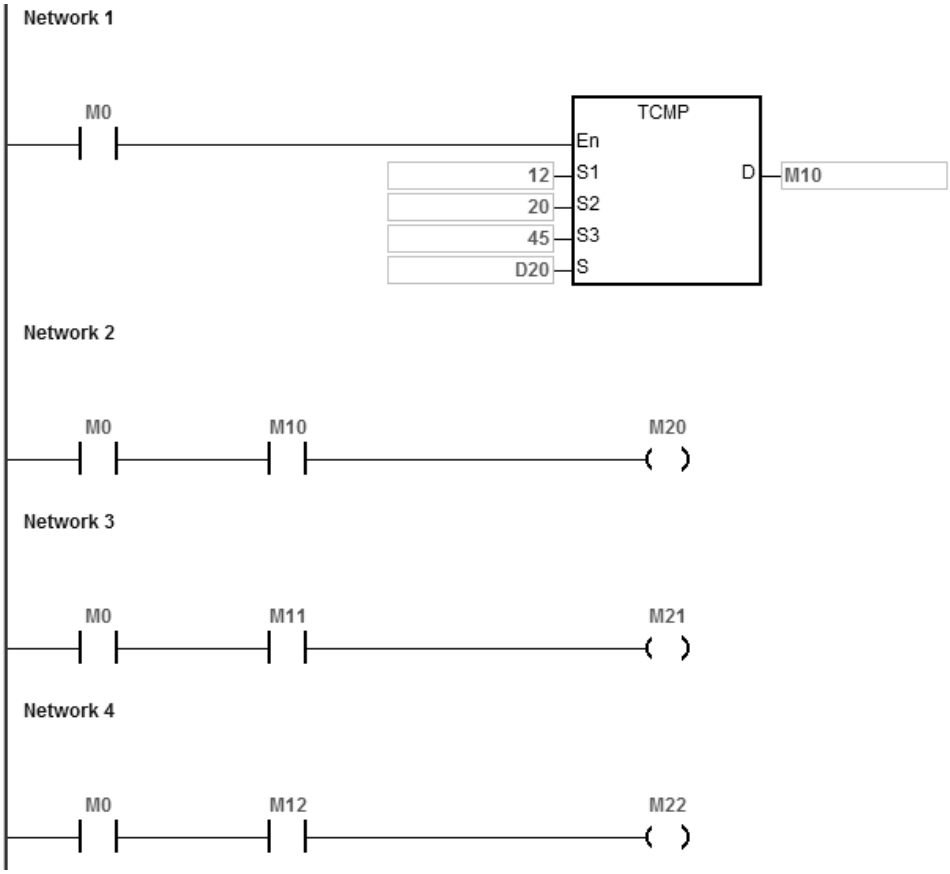
6. If the setting time in **S**$_1$-**S**$_3$ is equal to the current time in **S**, **D** is OFF, **D**+1 is ON, and **D**+2 is OFF.

7. If the setting time in **S**$_1$-**S**$_3$ is less than the current time in **S**, **D** is OFF, **D**+1 is OFF, and **D**+2 is ON.

Example

1. When M0 is ON, the instruction compares the setting time 12 hour 20 minute 45 second with the current time in D20–D22, and stores the comparison result in M10–M12. When M0 switches from ON to OFF, the instruction is not executed, and the states of M10, M11, and M12 remain the same as they were before M0 switched to ON.

2. If you want to get the comparison result ≧, ≦, or ≠, you can connect M10–M12 in series or in parallel.



Additional remarks

1. If **S**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **S** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the values in $\mathbf{S}_1$–$\mathbf{S}_3$ exceed the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

5. If you declare the operand **S** in ISPSoft, the data type is ARRAY [3] of WORD.

6. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.

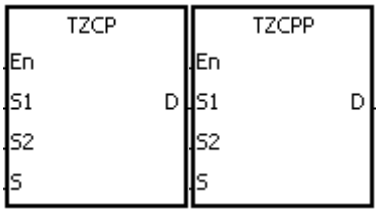
API	Instruction code			Operand	Function
1606		TZCP	P	**S_1 · S_2 · S · D**	Time zone comparison

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							
S_2					●	●		●	●							
S					●	●		●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Lower limit time

S_2 : Upper limit time

S : Current time

D : Comparison result

Explanation

1. This instruction compares the current time specified by **S** with the lower limit time specified by **S_1**, and with the upper limit time specified by **S_2**, and stores the comparison result in **D**.

2. The hour of the lower limit time is in the device specified by **S_1**, the minute of the lower limit time is in the device specified by **S_1**+1, and the second of the lower limit time is in the device specified by S_1+2.

3. The hour of the upper limit time is in the device specified by **S_2**, the minute of the upper limit time is in the device specified by **S_2**+1, and the second of the upper limit time is in the device specified by **S_2**+2.

4. The hour of the current time is in the device specified by **S**, the minute of the current time is in the device specified by **S**+1, and the second of the current time is in the device specified by **S**+2.

5. The time in the device specified by **S_1** must be less than the time in the device specified by **S_2**. If the time in the device specified by **S_1** is larger than the time in the device specified by **S_2**, the instruction takes the time in the device specified by **S_1** as the upper limit time during the execution of the instruction.

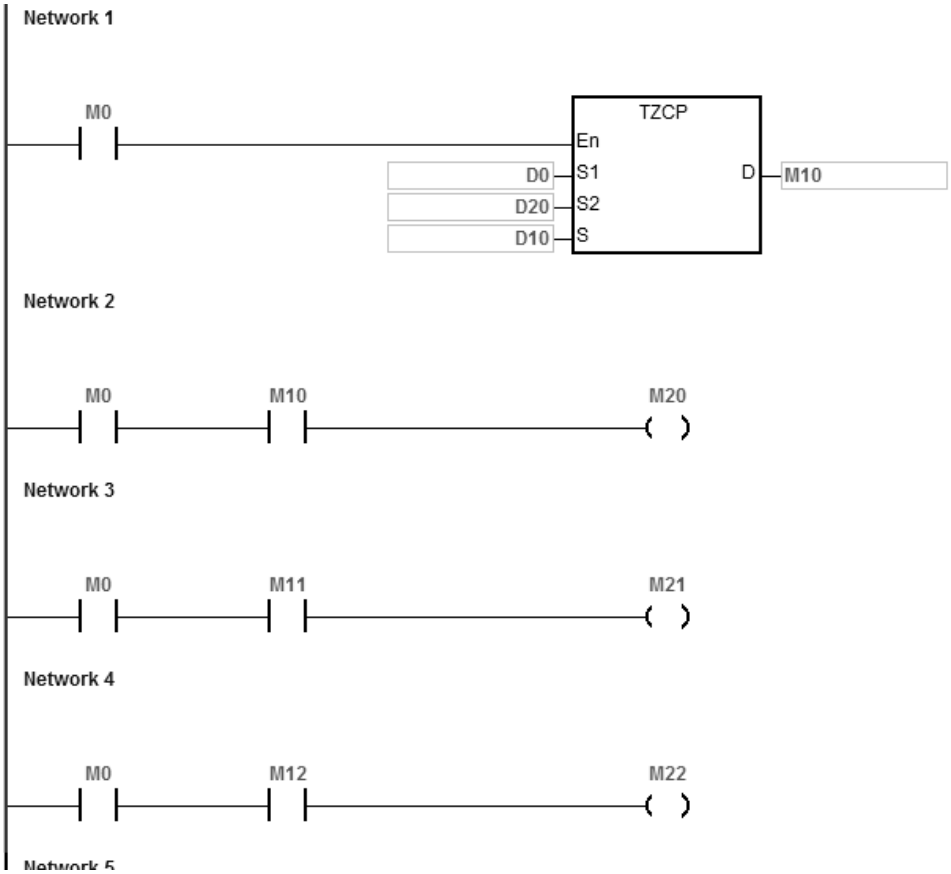
6. In general, use the TRD instruction (API 1600) to read the current time from the real-time clock first, and then use

the TZCP instruction to compare the time.

7. If the current time in the device specified by **S** is less than the lower limit time in the device specified by $\mathbf{S_1}$, and is less than the upper limit time in the device specified by $\mathbf{S_2}$, **D** is ON. If the current time in the device specified by **S** is larger than the lower limit time in the device specified by $\mathbf{S_1}$, and is larger than the upper limit time in the device specified by $\mathbf{S_2}$, D+2 is ON; otherwise D+1 is ON.

Example

When M0 is ON, the TZCP instruction is executed. M10, M11, or M12 is ON. When M0 is OFF, the instruction is not executed, the state of M10, the state of M11, and the state of M12 remain the same as before M0 switched to ON.



Additional remarks

1. If $\mathbf{S_1}$+2, $\mathbf{S_2}$+2, **S**+2, or **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the values in $\mathbf{S_1}$, $\mathbf{S_2}$, and **S** exceed the range, an operation error occurs, the instruction is not executed, SM0 is ON,

and the error code in SR0 is 16#2003

3. If you declare the operand $\mathbf{S}_1$ in ISPSoft, the data type is ARRAY [3] of WORD/INT.

4. If you declare the operand $\mathbf{S}_2$ in ISPSoft, the data type is ARRAY [3] of WORD/INT.

5. If you declare the operand **S** in ISPSoft, the data type is ARRAY [3] of WORD/INT.

6. If you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of BOOL.

API	Instruction code			Operand	Function
1607		DST	P	**S, S_1, S_2, S_3, S_4, S_5, D**	Daylight saving time

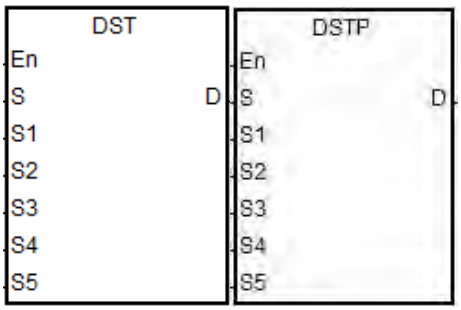
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●					○	○		
S_1					●	●		●					○	○		
S_2					●	●		●					○	○		
S_3					●	●		●					○	○		
S_4					●	●		●					○	○		
S_5					●	●		●					○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S_5		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

6

Symbol



S : Daylight saving time function codes

S_1 : Month of the daylight saving start time

S_2 : Date of the daylight saving start time

S_3 : Month of the daylight saving end time

S_4 : Date of the daylight saving end time

S_5 : Change due to daylight saving time (minutes)

D : The state of the daylight saving function

Explanation

1. Operands used in this instruction are described below:

   S: Daylight saving time function codes

Function codes	Description
0	Disable daylight saving time
1	Enable daylight saving time mode 1
2	Read daylight saving time
3	Enable daylight saving time mode 1
4	Disable daylight saving time (set by the system)
5	Daylight saving time mode 1 enabled (set by the system)
7	Daylight saving time mode 2 enabled (set by the system)
6, 8~	Reserved or viewed as reading daylight saving time

   Note 1: When the code in **S** is 4, 5 or 7, the execution of instruction is of no use.

   Note 2: Read more for information on the various modes in the following sections.

   S_1: setting for the month to start daylight saving time

   S_2: setting for the date to start daylight saving time

   S=1 (daylight saving time mode 1 enabled), **S_2**: settings for the date to start daylight saving time

   S=3 (daylight saving time mode 2 enabled), **S_2**: settings for the week to start daylight saving time, **S_2**+1: on which weekday of **S_2**

   S_3: setting for the month to end daylight saving time

   S_4: settings for the date to end daylight saving time

   S=1 (daylight saving time mode 1 enabled), **S_4**: settings for the date to end daylight saving time

   S=3 (daylight saving time mode 2 enabled), **S_4**: settings for the week to end daylight saving time, **S_2**+1: on which weekday of **S_4**

   S_5: settings for the change due to daylight saving time; unit: minute

   D: stores the state of the daylight saving time; when the value in **D** is OFF, daylight saving time is disabled. When the value in **D** is ON, daylight saving time is enabled.

2. Descriptions on the value in function code S for daylight saving time functions

D.S.T State	S Function Code	Description
Disabled	0	Disabled daylight saving time function

Enabled	1, 3	Enabled daylight saving time function
Read	2	Read the daylight saving time setting

- **Disabled daylight saving time function (refer to example 1 below)**

  When the operand **S** is 0, the function of daylight saving time is disabled. When **S** is set to disable the daylight saving time, the values in **S_1**–**S_5** are irrelevant and the operand D shows the daylight saving time state as OFF.

- **Enabled daylight saving time function (refer to example 2 and 3)**

  When the value in S is 1 or 3, daylight saving time function is enabled: **S_1 and S_2**: setting for the month to start daylight saving time; **S_3 and S_4**: setting for the month to end daylight saving time; **S_5**: settings for the change due to daylight saving time; unit: minute; the operand D shows the daylight saving time state. When the function of daylight saving time is enabled and the system runs for the first time during the start time (**S_1**, **S_2**), the system time adds the value set in **S_5** once. When the function of daylight saving time is disabled and the system runs for the first time during the end time (**S_1**, **S_2**), the system time subtracts the value set in **S_5** once.

Modes for daylight saving

Mode	S Function Code	Rules
Mode 1	1	By month and date
Mode 2	3	By month and week

Mode 1 (S=1): enabled by month and date (refer to example 2)

Operand	Description
S_1	The month to start daylight saving time Range: 1-12
S_2	The date to start daylight saving time Range: 1-31
S_3	The month to end daylight saving time Range: 1-12
S_4	The date to end daylight saving time Range: 1-31
S_5	Time that changed due to daylight saving time; unit: minute Range: 1-1439 (within 1 day)

Note 1: If this function is enabled, the value in **D** is ON.

Note 2: If the date is set incorrectly, the daylight saving function cannot be enabled. The SM0 is ON, and the error code in SR0 is 16#200B. For example if a non-existed date is set, such as April 31, or the

starting date is set smaller than the ending date in a calendar year, for example starting date is October 1 and ending date is April 01.

Note 3: If **S_5** is set out of range, the daylight saving function cannot be enabled. The SM0 is ON, and the error code in SR0 is 16#200B.

Mode 2 (S=3): enabled by week and weekday (refer to example 3)

Operand	Description
S_1	The month to start daylight saving time Range: 1-12
S_2 **S_2**+1	**S_2**: settings for the week to start daylight saving time; range: 1-4 **S_2**+1: on which weekday of the **S_2**; range: 1-7 (Monday: 1, Tuesday: 2..., Sunday: 7)
S_3	The month to end daylight saving time Range: 1-12
S_4 **S_4**+1	**S_4**: settings for the week to end daylight saving time; range: 1-4 **S_4**+1: on which weekday of the **S_4**; range: 1-7 (Monday: 1, Tuesday: 2..., Sunday: 7)
S_5	Time that changed due to daylight saving time; unit: minute Range: 1-1439 (within 1 day)

Note 1: If this function is enabled, the value in **D** is ON.

Note 2: The setting range for **S_2** and **S_4** is 1— 4 or -1— -4. The value -1 indicates the last week of the month and -2 indicates the last 2nd week. If the value in **S_2** is -2 and **S_2**+1 is 7, it indicates the last 2 Sunday of the month. If the date is set incorrectly, the daylight saving function cannot be enabled. The SM0 is ON, and the error code in SR0 is 16#200B.

Note 3: If the value in **S_2**+1 / **S_4**+1 is out of range, the default setting value is 7, indicating Sunday.

Note 4: If **S_5** is set out of range, the daylight saving function cannot be enabled. The SM0 is ON, and the error code in SR0 is 16#200B.

Note 5: If the device for operand **S_2** and **S_4** is K or 16#, the values are not saved, the SM0 is ON, and the error code in SR0 is 16#2003.

- **Read the daylight saving time function (refer to example 1-3)**

When the operand **S** is 2, the function of daylight saving time is being read. **S_1 and S_2**: setting for the month to start daylight saving time; **S_3 and S_4**: setting for the month to end daylight saving time; **S_5**: settings for the change due to daylight saving time; unit: minute. When **S** is set to read the state of the daylight saving function and the output state of **D** is ON, the PLC saves the setting values in the operands **S_1**–**S_5**. The device is set to **D** while **S** is set to read. If the device is set to K or 16#, the values are not saved, the SM0 is ON, and the error code in SR0 is 16#2003.

Adds 4 to the function codes in **S**, after the daylight saving state is read. For example, after the daylight saving state is read, the function codes 0, 1, 3 become 4, 5 and 7.

When the DST state is OFF, the operand and descriptions are shown below.

Operand	Description
S	Function code is 4, indicating the DST state is OFF.
S_1- S_5	Invalid operand
D	DST state is OFF.

When the DST state is ON and in mode 1, the operand and descriptions are shown below.

Operand	Description
S	Function code is 5, indicating the DST state is ON and in mode 1.
S_1	The month to start daylight saving time
S_2	The date to start daylight saving time
S_3	The month to end daylight saving time
S_4	The date to end daylight saving time
S_5	Time that changed due to daylight saving time; unit: minute
D	The DST state is ON (enabled).

When the DST state is ON and in mode 2, the operand and descriptions are shown below.

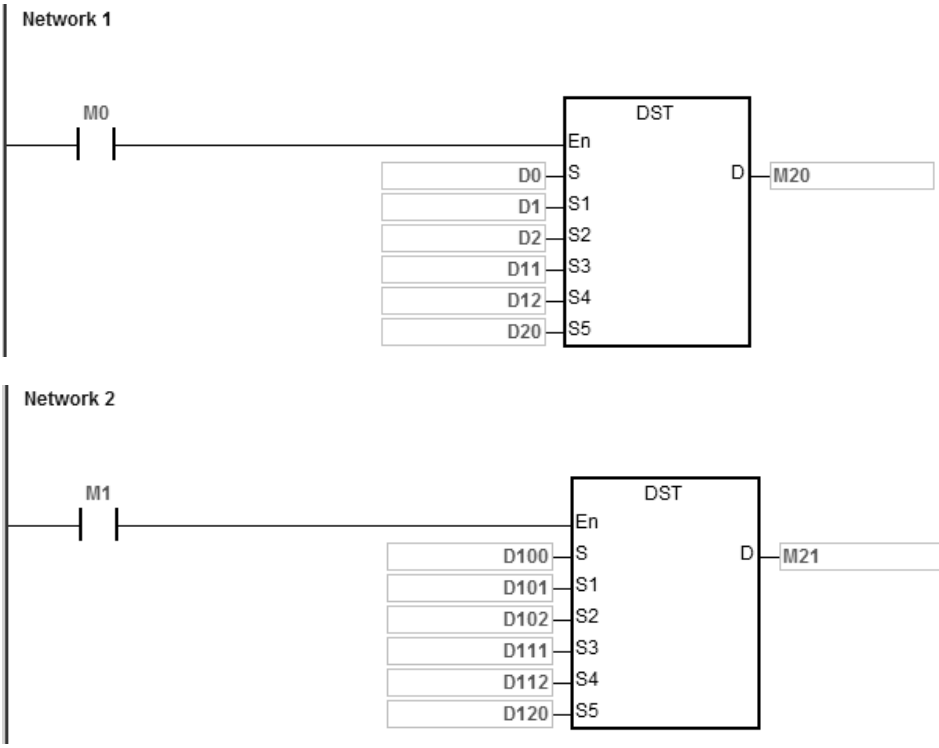
Operand	Description
S	Function code is 7, indicating the DST state is ON and in mode 2.
S_1	The month to start daylight saving time
S_2 **S_2**+1	**S_2**: settings for the week to start daylight saving time **S_2**+1: on which weekday of the **S_2**
S_3	The month to end daylight saving time
S_4 **S_4**+1	**S_4**: settings for the week to end daylight saving time **S_4**+1: on which weekday of the **S_4**
S_5	Time that changed due to daylight saving time; unit: minute
Operand	The DST state is ON (enabled).

3. This instruction is to enable / disable the daylight saving time function. Whether the contact is normally open or close will not affect the daylight saving time setting. (refer to example 2 for more details on how to switch the contact M0 OFF=>ON) You can reset the daylight saving time by executing the instruction again. There is no need to disable and then enable this function to reset the daylight saving time.
4. When setting the daylight saving time to start on April 1st and to end on September 1st, and the duration is 60 minutes; the real-time clock goes like below.

Daylight saving time function disabled	Daylight saving time function enabled
1st March, 3 o'clock	1st March, 3 o'clock
31st March, 3 o'clock	31st March, 3 o'clock
1st April, 3 o'clock	1st April, 4 o'clock
1st May, 3 o'clock	1st May, 4 o'clock
1st June, 3 o'clock	1st June, 4 o'clock
1st July, 3 o'clock,	1st July, 4 o'clock
1st August, 3 o'clock	1st August, 4 o'clock,
31st August, 3 o'clock	31st August, 4 o'clock
1st September, 3 o'clock	1st September, 3 o'clock

Example 1

Disable DST function and read the DST state.



Setting values and descriptions:

Device	Setting Value	Description
D0	0	Disable DST function
D1	X	Invalid operand
D2	X	Invalid operand
D11	X	Invalid operand
D12	X	Invalid operand

D20	X	Invalid operand

Enable contact M0

M20=OFF, indicating DST function is disabled.

D100=K2, indicating DST state is being read.

Enable contact M1

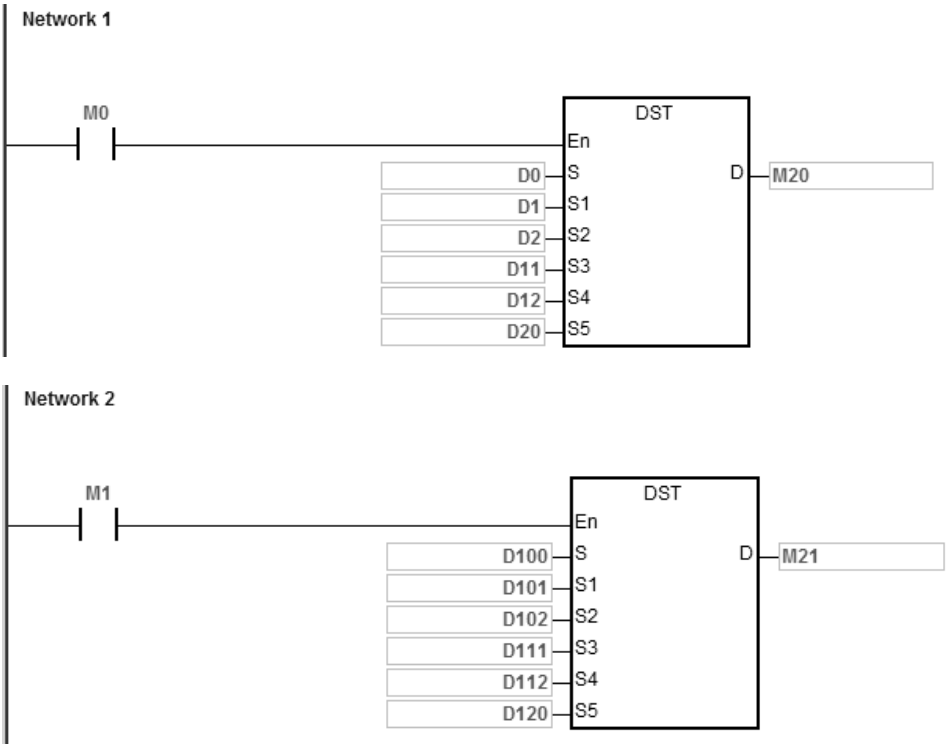
Setting values and descriptions:

Device	Setting Value	Description
D100	4	Function code is 4, indicating the DST state is OFF.
D101	X	Invalid operand
D102	X	Invalid operand
D111	X	Invalid operand
D112	X	Invalid operand
D120	X	Invalid operand
M21	OFF	Node state is OFF.

Example 2

Enable DST function and read the DST state.

Set the DST to start on 1st April and to end on 3rd September and the duration is 60 minutes.



Setting values and descriptions:

Device	Setting Value	Description
D0	1	The DST state is ON and in mode 1.
D1	4	Starting month: April
D2	1	Starting date: the 1st
D11	9	Ending month: September
D12	3	Ending date: the 3rd
D20	60	Duration: 60 minutes

Enable contact M0

M20=ON, indicating DST function is enabled.

The PLC system time adds 60 minutes when the date April 1st arrives, and subtracts 60 minutes when the date September 3rd arrives to end daylight saving time.

D100=K2, indicating DST state is being read.

Enable contact M1

Setting values and descriptions:

Device	Setting Value	Description
D100	5	Function code is 5, indicating the DST state is ON and in mode 1.
D101	4	Starting month: April
D102	1	Starting date: the 1st
D111	9	Ending month: September
D112	3	Ending date: the 3rd
D120	60	Duration: 60 minutes
M21	ON	Node state is ON.

Use the instruction DST or HWCONFIG in ISPSoft to read the daylight saving state. The HWCONFIG converts the result from week number to the corresponding dates and months automatically.

Enable the contact M0 OFF=>ON again; this act has no impact on the DST. The DST does not reset.

Setting values and descriptions:

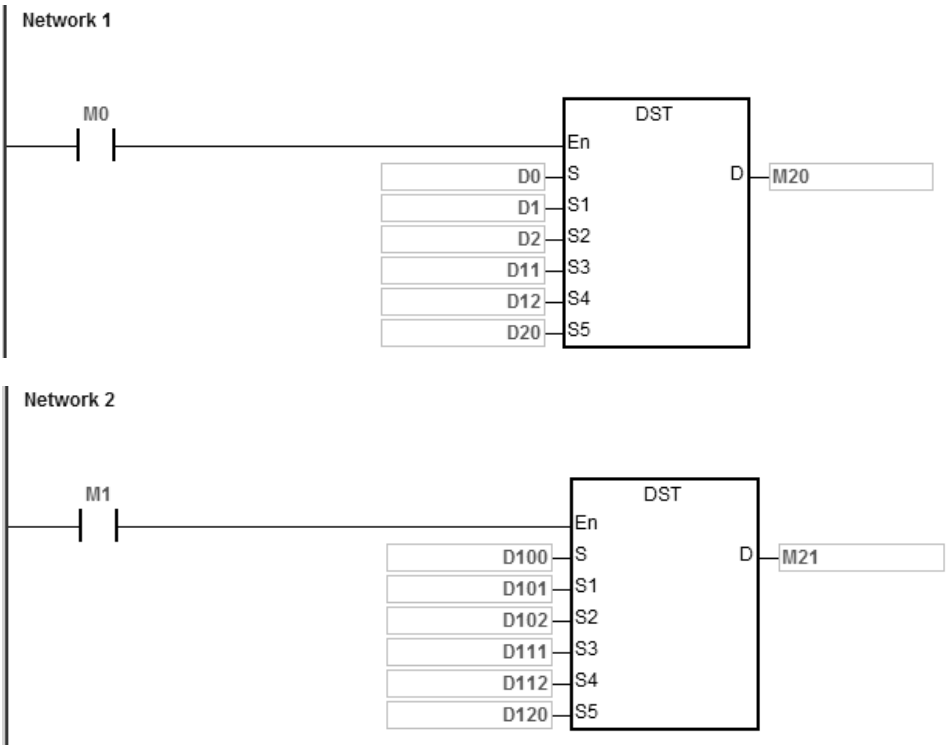
Device	Setting Value	Description
D100	5	Function code is 5, indicating the DST state is ON and in mode 1.
D101	4	Starting month: April
D102	1	Starting date: the 1st
D111	9	Ending month: September
D112	3	Ending date: the 3rd
D120	60	Duration: 60 minutes
M21	ON	Node state is ON.

If the daylight saving time starts from 3 o'clock of 1st April, 60 minutes is added; the real-time clock shows 4 o'clock of 1st April. No matter how many times the contact M0 is disabled or enabled, the real-time clock keeps the same daylight saving time.

Example 3

Enable DST function and in mode 2.

Set the DST to start from the 2nd Wednesday of May and to end on 3rd Friday of September and the duration is 60 minutes.



Setting values and descriptions:

Device	Setting Value	Description
D0	3	The DST state is ON and in mode 2.
D1	5	Starting month: May
D2	2	Starting week number: the 2nd week
D3	3	Starting day: Wednesday
D11	9	Ending month: September
D12	3	Ending week number: the 3rd week
D13	5	Ending day: Friday
D20	60	Duration: 60 minutes

Enable contact M0

M20=ON, indicating DST function is enabled.

For the year 2017, the 2nd Wednesday of May is 10th May and the 3rd Friday of September is 15th September. The PLC system time adds 60 minutes when the date May 10th arrives and subtracts 60 minutes when the date September 15th arrives to end daylight saving time.

D100=K2, indicating DST state is being read.

Enable contact M1

Setting values and descriptions:

Device	Setting Value	Description
D100	7	Function code is 4, indicating the DST state is ON and in mode 2.
D101	5	Starting month: May
D102	2	Starting week number: the 2nd week
D103	3	Starting day: Wednesday
D111	9	Ending month: September
D112	3	Ending week number: the 3rd week
D113	5	Ending day: Friday
D120	60	Duration: 60 minutes
M21	ON	Node state is ON.

Use the instruction DST or HWCONFIG in ISPSoft to read the daylight saving state. The HWCONFIG converts the result from week number to the corresponding dates and months automatically.

API	Instruction code	Operand	Function

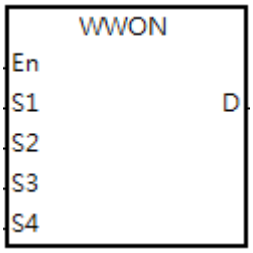
1608		WWON		S_1, S_2, D	Setting up weekly working time

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								○								
S_2								○								
S_3								○								
S_4								○								
D		○	○	○												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : The hour to start working (occupies 7 consecutive devices)

S_2 : The minute to start working (occupies 7 consecutive devices)

S_3 : The hour to stop working (occupies 7 consecutive devices)

S_4 : The minute to stop working (occupies 7 consecutive devices)

D : Output control

Explanation

1. This instruction allows you to set the time to start working for the week. **S_1**–**S_1**+6 allows you to set the time on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices. You can use the variables in an ARRAY to declare the operands.

2. **S_2**–**S_2**+6 **S_1**–**S_1**+6 allows you to set the minutes to start working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices. You can use the variables in an ARRAY to declare the operands.

3. **S_3**–**S_3**+6 allows you to set the hour to stop working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices. You can use the variables in an ARRAY to declare the operands.

4. **S_4**–**S_4**+6 allows you to set the minutes to stop working on Sunday / Monday / Tuesday / Wednesday / Thursday / Friday / Saturday respectively. This operand occupies 7 consecutive devices. You can use the variables in an

ARRAY to declare the operands.

5. When the hour value in **S1** is larger than the value set in **S3**, it means the time to stop working is the next day. For example, when you set the time to start working at 18:00 on Monday and the time to stop working at 6:00, it means the time to stop working is at 6:00 Tuesday.

	Start working time						Stop working time					
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute	
Sunday	**S1**	24		**S2**	00		**S3**	24		**S4**	00	
Monday	**S1**+1	18		**S2**+1	00		**S3**+1	06		**S4**+1	00	

6. The setting value for the hour is between 0–23. When the setting value is out of range, this function is not enabled. The setting value for the minute is between 0–59. When the setting value is out of range, this function is enabled but uses 0 as the setting value.

7. When it is required to set the work time to be more than 1 day, you can set the hour as 24, which means the system does not check the start working time and the stop working time. For example, to set the start working time to 8 am Monday and the stop working time to 8pm Tuesday, use **S1**+1=8, **S3**+1=24, **S1**+2 =24 and **S3**+2=20. See the formula in the following table.

	Start working time						Stop working time					
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute	
Sunday	**S1**	24		**S2**	00		**S3**	24		**S4**	00	
Monday	**S1**+1	08		**S2**+1	00		**S3**+1	24		**S4**+1	00	
Tuesday	**S1**+2	24		**S2**+2	00		**S3**+2	20		**S4**+2	00	

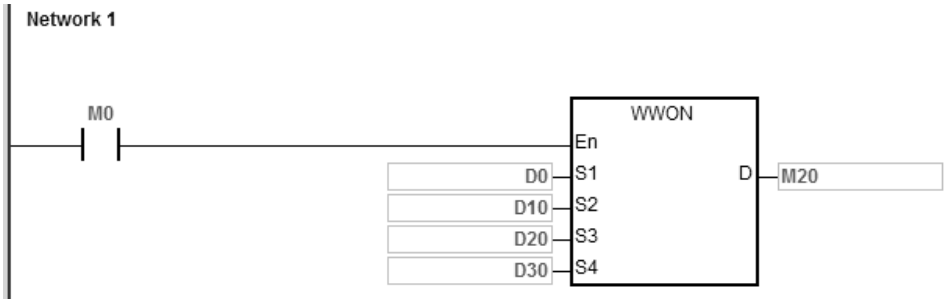
8. This instruction should work with the real-time clock in the PLC. Before operating, make sure the PLC battery is securely installed and working correctly.

9. There is no limit on the number of times you can execute the instruction but the output control device **D** cannot be used repeatedly. If you use the device **D** repeatedly, only the last output result from the WWON instruction is executed.

10. If more than 1 set of work hours are needed, use the WWON instruction repeatedly as required. Note that you cannot use the output control device **D** repeatedly.

Example 1

Set a working time from 8:00 to 18:00 from Monday to Friday and no work on Saturday and Sunday.



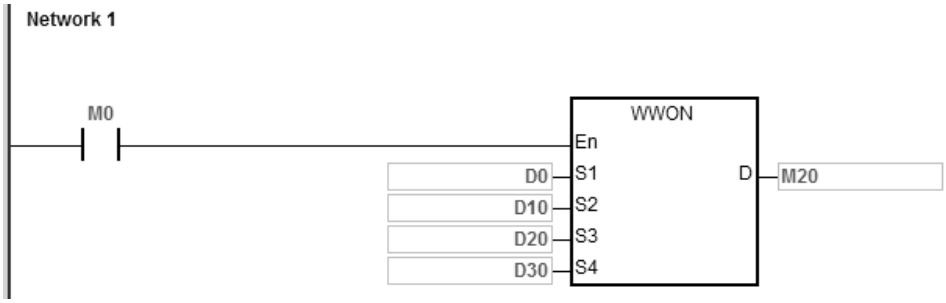
The following table lists the settings for the device **D**.

	Start working time					Stop working time				
Day	Start	Hour		Start	Minute	Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00	D20	24		D30	00
Monday	D1	08		D11	00	D21	18		D31	00
Tuesday	D2	08		D12	00	D22	18		D32	00
Wednesday	D3	08		D13	00	D23	18		D33	00
Thursday	D4	08		D14	00	D24	18		D34	00
Friday	D5	08		D15	00	D25	18		D35	00
Saturday	D6	24		D16	00	D26	24		D36	00

When M0 is ON, M20 is ON from 8:00 to 18:00 from Monday to Friday; for other times the M20 is OFF.

Example 2

Set a working time from 18:00 Monday to 08:00 Tuesday and from 18:00 Tuesday to 08: 00 Wednesday. Follow this pattern to 08:00 Saturday and no work on Sunday.



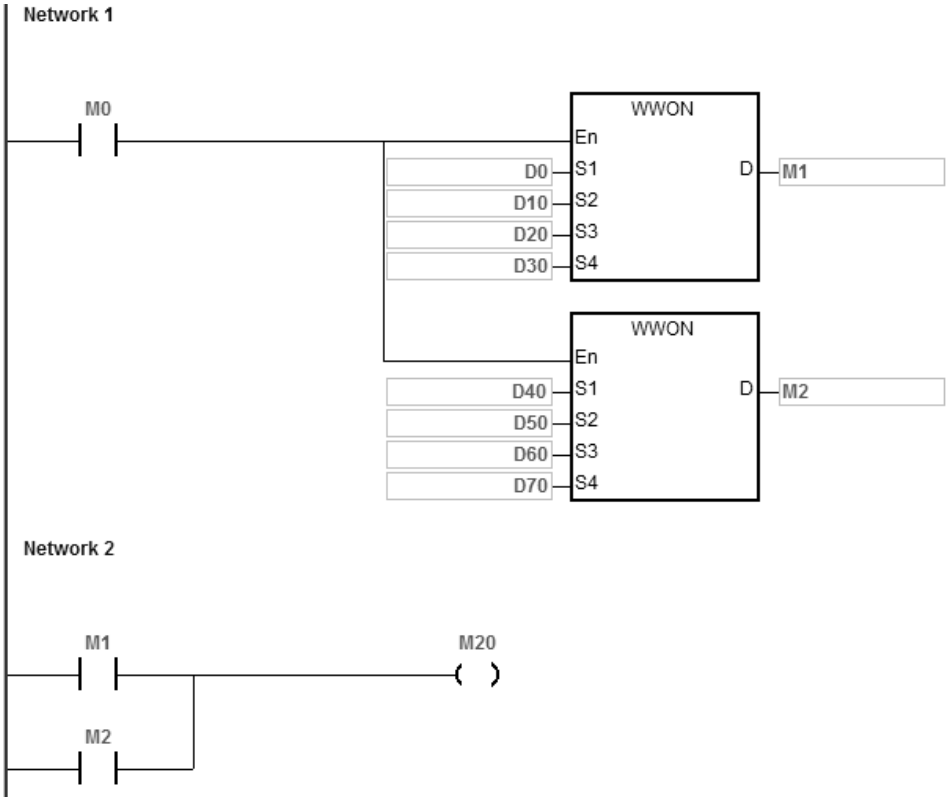
The following table lists the settings for the device **D**.

	Start working time					Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00		D20	24		D30	00
Monday	D1	18		D11	00		D21	08		D31	00
Tuesday	D2	18		D12	00		D22	08		D32	00
Wednesday	D3	18		D13	00		D23	08		D33	00
Thursday	D4	18		D14	00		D24	08		D34	00
Friday	D5	18		D15	00		D25	08		D35	00
Saturday	D6	24		D16	00		D26	24		D36	00

When M0 is ON, M20 is ON from 18:00 to 8:00 the next day from Monday to Friday and for other times the M20 is OFF.

Example 3

Set a working time from 08:00 to 12:00 and from 14:00 to 17:30 from Monday to Friday. No work on Saturday and Sunday.



The following table lists the settings in the morning for the device **D**.

	Start working time					Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D0	24		D10	00		D20	24		D30	00
Monday	D1	08		D11	00		D21	12		D31	00
Tuesday	D2	08		D12	00		D22	12		D32	00
Wednesday	D3	08		D13	00		D23	12		D33	00
Thursday	D4	08		D14	00		D24	12		D34	00
Friday	D5	08		D15	00		D25	12		D35	00
Saturday	D6	24		D16	00		D26	24		D36	00

The following table lists the settings in the afternoon for the device **D**.

	Start working time					Stop working time				
Day	Start	Hour		Start	Minute		Stop	Hour		Stop	Minute
Sunday	D40	24		D50	00		D60	24		D70	00
Monday	D41	14		D51	00		D61	17		D71	30
Tuesday	D42	14		D52	00		D62	17		D72	30
Wednesday	D43	14		D53	00		D63	17		D73	30
Thursday	D44	14		D54	00		D64	17		D74	30
Friday	D45	14		D55	00		D65	17		D75	30
Saturday	D46	24		D56	00		D66	24		D76	00

When M0 is ON, M20 is ON from 08:00 to 12:00 and 14:00 to 17:30 from Monday to Friday and for other times the M20 is OFF.

6.18 Peripheral Instructions

6.18.1 List of Peripheral Instructions

The following table lists the Peripheral instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1700	TKY	DTKY	–	Ten-key keypad
1701	HKY	DHKY	–	Sixteen-key keypad
1702	DSW	–	–	DIP switch
1703	ARWS	–	–	Arrow keys
1704	SEGL	–	–	Seven-segment display with latches

6.18.2 Explanation of Peripheral Instructions

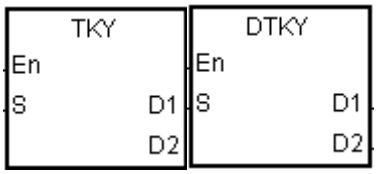
API	Instruction code			Operand	Function
1700	D	TKY		**S, D_1, D_2**	Ten-key keypad

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S	○															
D_1					●	●	●	●								
D_2		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●												
D_1		●	●		●	●	●						
D_2	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



S : First input device

D_1 : Device where the value is stored

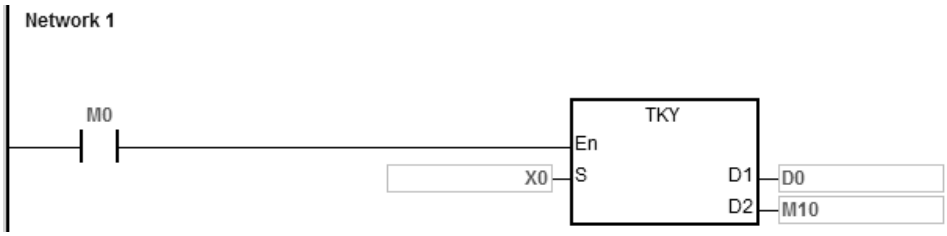
D_2 : Output signal

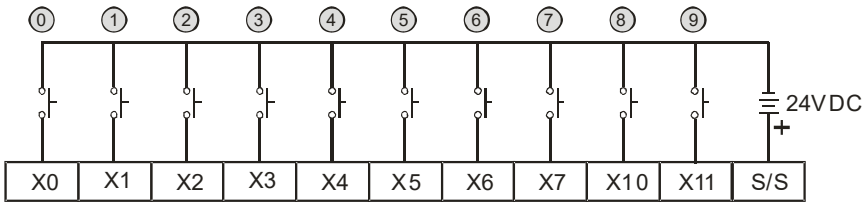
Explanation

1. The ten external inputs starting from the input specified by **S** represents 0–9 in the decimal system. They are connected to ten keys. You can enter a four-digit decimal value 0–9,999 (16-bit instruction) or an eight-digit decimal value 0–99,999,999 (32-bit instruction) by pressing the keys in order. The instruction stores the decimal value in **D_1**, and stores the output signals in **D_2**.
2. The operand **S** occupies ten bits.
3. The operand **D_2** occupies eleven bits. Please do not change the states of the bits during the execution of the instruction.
4. When the conditional contact is not enabled, the eleven bits starting from the bit specified by **D_2** are OFF.
5. When using on-line editing, please reset the conditional contact to initialize the instruction.
6. You can use the 32-bit counter only when **D_1** uses 32-bit instructions.

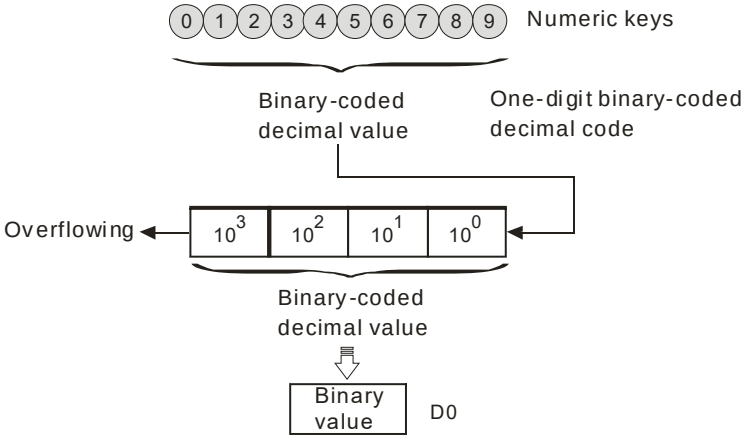
Example

1. The ten external inputs starting from X0 are connected to ten keys that represent 0–9 in the decimal system. When M0 is ON, the instruction stores the value that you enter as a binary value in D0, and stores the output signals in M10–M19.





Note: The digital input module is used in this example.



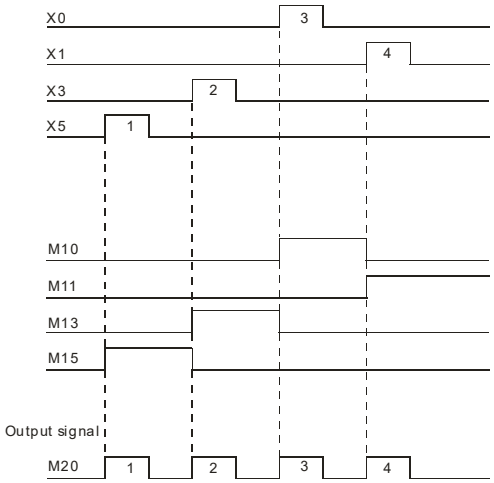
2. If the keys connected to X5, X3, X0, and X1 are pressed in the order shown in the timing chart, the instruction stores the result 5,301 in D0. The maximum value that can be stored in D0 is 9,999. If the value exceeds four digits, the first digit from the left overflows.

3. After the key connected to the X2 is pressed and before other keys are pressed, M12 is ON. The same applies to the other keys.

4. When a key connected to the input within the range between X0~X7 and X10~X11 is pressed, the corresponding output within the range between M10 and M19 is ON.

5. When one of the keys is pressed, M20 is ON.

6. When the conditional contact M0 is switched OFF, the value stored in D0 is unchanged; however, M10–M20 are switched OFF.



Additional remarks

1. If you declare the operand **S** in ISPSoft, the data type is ARRAY [10] of BOOL.

2. If you declare the operand **D_2** in ISPSoft, the data type is ARRAY [11] of BOOL.

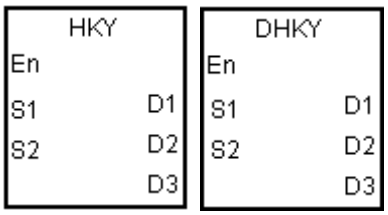
API	Instruction code			Operand	Function
1701	D	HKY		**S_1, S_2, D_1, D_2, D_3**	Sixteen-key keypad

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1		○														
D_2					●	●	●	●								
D_3		○	○	○				○								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1	●												
D_2		●	●		●	●	●						
D_3	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	ES3

Symbol



S_1 : First input device

S_2 : For system use only

D_1 : First output device

D_2 : Device where the value is stored

D_3 : Output signal

Explanation

1. The four external inputs starting from the input specified by **S** are connected to the four external outputs starting from the output specified by **D_1** to form a 16-key keypad. The instruction stores the value that you enter by pressing the keys in **D_2**, and stores the output signals in **D_3**. If you press several keys simultaneously, the value that is smaller is stored.

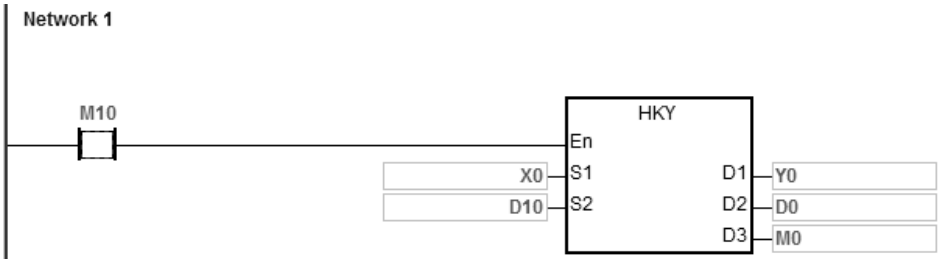
2. The value that you enter by pressing the keys is temporarily stored in **D_2**. For the 16-bit HKY instruction, the maximum value that can be stored in **D_2** is 9,999. If the value exceeds four digits, the first digit from the left overflows. For the 32-bit DHKY instruction, the maximum value that can be stored in **D_2** is 9,999. If the value exceeds eight digits, the first digit from the left overflows.

3. After the instruction completes, SM692 is ON. That is, SM692 is ON for a scan cycle after the execution of the matrix scan is complete.

4. You can use the 32-bit counter only when **D_2** uses 32-bit instructions.

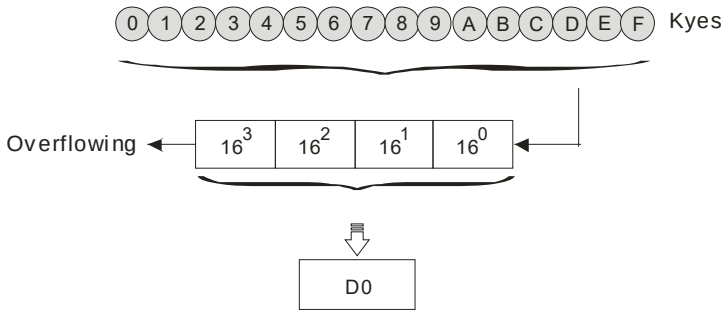
Example

1. The four external inputs X0–X3 are connected to the four external outputs Y0–Y3 to form a 16-key keypad. When X10 is ON, the instruction stores the value that you enter as a binary value in D0, and stores the output signals in M0–M7.

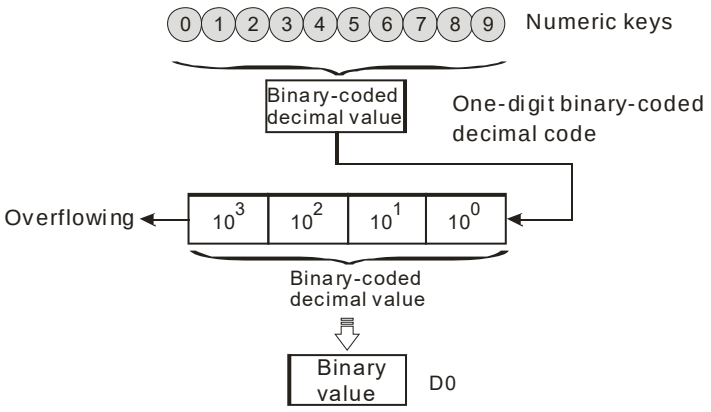


The function of SM691:

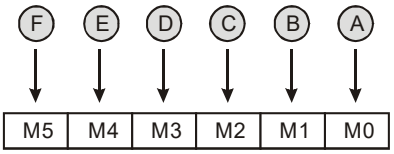
- If SM691 is ON, the 16-bit instruction takes 0–F as hexadecimal values.
  - Numeric keys:



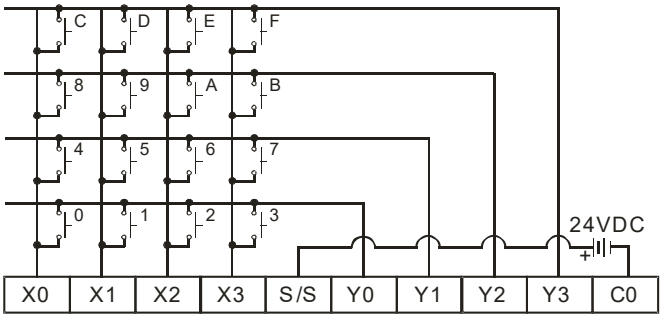
- If SM691 is OFF, the 16-bit instruction takes A–F as function keys.
  - Numeric keys:



- Function keys:
  - When A is pressed, M0 stays ON. When D is pressed, M0 switches OFF, and M3 stays ON.
  - If several function keys are pressed, the key which is pressed first has priority.



2. Output signals:
   - When a key within the range between A and F is pressed, M6 is ON.
   - When a key within the range between 0 and 9 is pressed, M7 is ON.
3. When the conditional contact M10 switches to OFF, the value that was stored in D0 is unchanged. However, M0–M7 are switched OFF.
4. The external wiring:



Note: The transistor output module is used in this example.

Additional remarks

1. When this instruction is executed, a too long or a too short scan cycle time will cause the state of the switches to be read incorrectly. Use the following tips to solve the issue.
   - When the scan cycle is too short, the I/O may not be able to respond in time and cannot read the correct states of the inputs. You can set a fixed scan time to solve this issue.
   - When the scan cycle is too long, the switch may become slow to react. You can write this instruction to a timer interrupt task to set a fixed time to execute this instruction.
2. If you declare the operand **S** in ISPSoft, the data type is ARRAY [4] of BOOL.
3. If you declare the operand **D1** in ISPSoft, the data type is ARRAY [4] of BOOL.
4. If you declare the operand **D3** in ISPSoft, the data type is ARRAY [8] of BOOL.

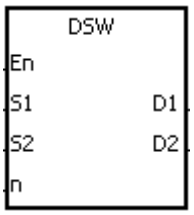
API	Instruction code	Operand	Function
1702	DSW	**S_1, S_2, D_1, D_2, n**	DIP switch

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1		○														
D_2					●	●		●								
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1	●												
D_2		●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : First input device

S_2 : For system use only

D_1 : First output device

D_2 : Device where the value is stored

n : Number of DIP switches

Explanation

1. The four or eight external inputs starting from the input specified by **S_1** are connected to the four external outputs starting from the output specified by **D_1** to form a four-digit DIP switch or two four-digit DIP switches. The instruction stores the value from the DIP switch in **D_2**. Whether there is one four-digit DIP switch or two four-digit DIP switches depends on **n**.

2. If **n** is 1, the operand **D_2** occupies one register. If **n** is 2, the operand **D_2** occupies two registers.

3. **S_2** and **S_2**+1, are for system use only, and occupy two devices. Please do not alter the values in these devices.

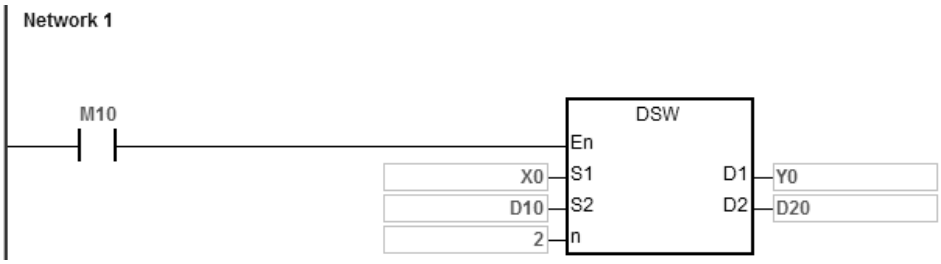
4. After the instruction completes, SM694 is ON for a scan cycle.

5. When the conditional contact is not enabled, the four external outputs starting from the output specified by **D_1** stay OFF.

6. When using on-line editing, please reset the conditional contact to initialize the instruction.

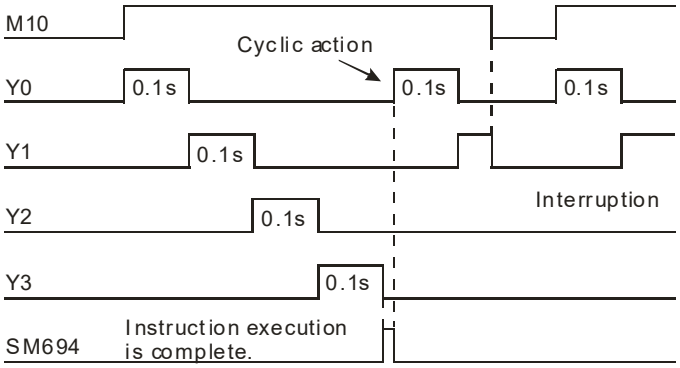
Example

1. X0–X3 are connected to Y0–Y3 to form the first DIP switch, and X4–X7 are connected to Y0–Y3 to form the second DIP switch. When M10 is ON, the instruction converts the value that you enter with the first DIP switch into the binary value, and stores the conversion result in D20. The instruction converts the value that you enter with the second DIP switch into the binary value, and stores the conversion result in D21.

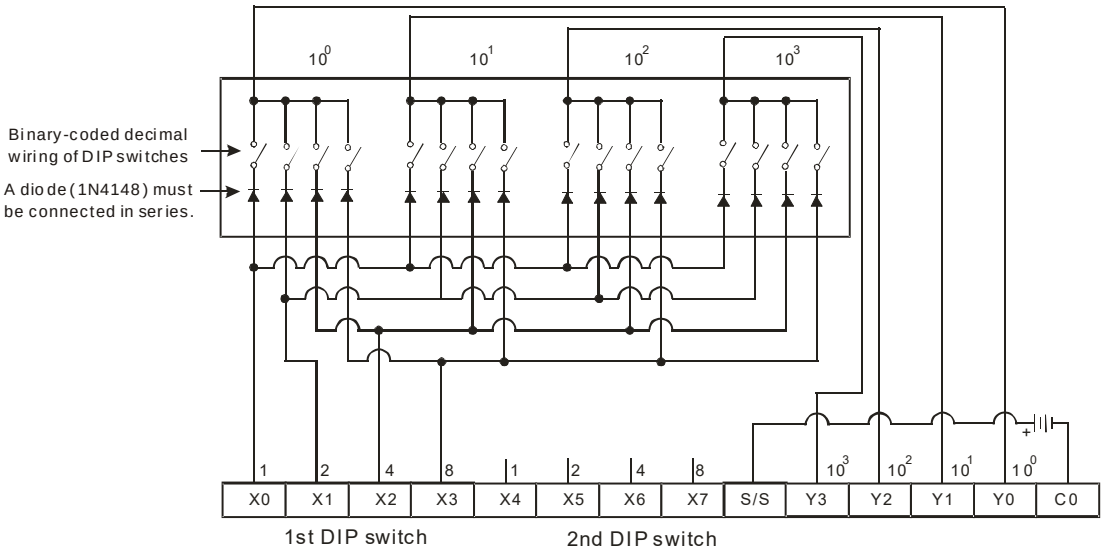


2. When M10 is ON, Y0–Y3 are ON cyclically. After the instruction completes, SM694 is ON for a scan cycle.

3. The following graphic shows the outputs. Y0–Y3 must be transistors.



4. The following graphic shows the DIP switches.



Note: The transistor output module is used in this example.

Additional remarks

1. If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If you declare the operand $\mathbf{D_1}$ in ISPSoft, the data type is ARRAY [4] of BOOL.

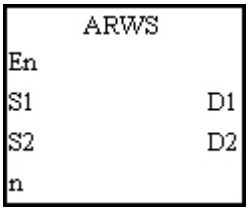
API	Instruction code			Operand	Function
1703		ARWS		S_1, S_2, D_1, D_2, n	Arrow keys

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
D_1					●	●		●								
D_2		○														
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2		●			●	●							
D_1		●			●	●							
D_2	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



- S_1 : First input device
- S_2 : For system use only
- D_1 : Device where the setting value is stored
- D_2 : First output device
- n : Positive/Negative logic

Explanation

1. This instruction defines S_1 as the down arrow, S_1+1 as the up arrow, S_1+2 as the right arrow, and S_1+3 as the left arrow. The instruction stores the setting value in D_1, and the value must be between 0–9,999.
2. The operand S_1 occupies four consecutive bit devices.
3. S_2 is for system use only. Please do not alter the value in it.
4. The operand D_2 occupies eight consecutive bit devices.
5. When the conditional contact is disabled, the eight bit devices starting from the bit device specified by D_2 stay OFF.
6. The operand n must be between 0–3.
7. When using on-line editing, please reset the conditional contact to initialize the instruction.

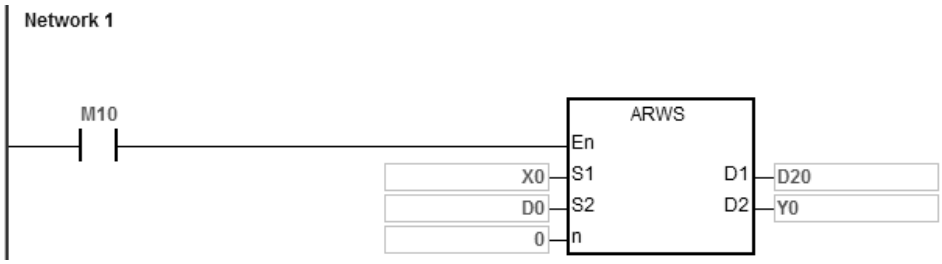
Example

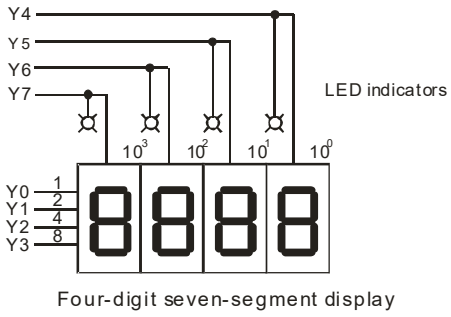
1. The instruction defines X0 as the down arrow, X1 as the up arrow, X2 as the right arrow, and X3 as the left arrow. The instruction stores the setting value in D20, and the setting value must be between 0–9,999.

2. When M10 is ON, the digit in the place 10^3 is selected. If the left arrow is pressed, the places are selected in sequence ($10^3 \rightarrow 10^0 \rightarrow 10^1 \rightarrow 10^2 \rightarrow 10^3 \rightarrow 10^0$).

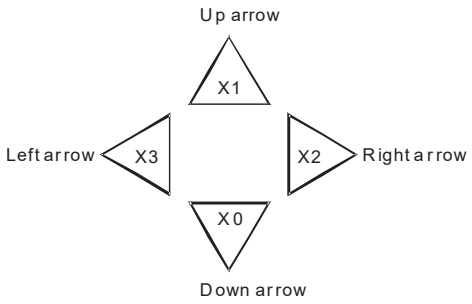
3. If you press right arrow, the places are selected in sequence ($10^3 \rightarrow 10^2 \rightarrow 10^1 \rightarrow 10^0 \rightarrow 10^3 \rightarrow 10^2$). The LED indicators for the corresponding places are connected to Y4–Y7. When the digits in the places are selected in sequence, the LED indicators are ON in sequence.

4. If you press the up arrow, the digit in the place selected changes (0→1→2→…8→9→0→1). If you press the down arrow, the digit in the place selected changes (0→9→8→…1→0→9). The new digit is shown on seven-segment display.





Four-digit seven-segment display



The four keys are used to select the place and change the digits.

Additional remarks

1. If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

2. If you declare the operand $\mathbf{S_1}$ in ISPSoft, the data type is ARRAY [4] of BOOL.

3. If you declare the operand $\mathbf{D_2}$ in ISPSoft, the data type is ARRAY [8] of BOOLL.

API	Instruction code	Operand	Function
1704	SEGL	**S_1, S_2, D, n**	Seven-segment display with latches

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●		○	○				
S_2								●								
D		○														
n					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

S_1 : Source device

S_2 : For system use only

D : First output device

n : Positive/Negative logic

Explanation

1. The eight external outputs starting from the output specified by **D** are connected to a four-digit seven-segment display; or the twelve external outputs starting from the output specified by **D** are connected to two four-digit seven-segment displays. Every place is equipped with a driver that converts a binary-coded decimal value into seven-segment data, and every driver is equipped with a latch that can be used to store state information.

2. The value in **S_1** is the value to show on first seven-segment display, and the value in **S_1**+1 is the value to show on second seven-segment display.

3. **S_2** is for system use only. Please do not alter the value in it.

4. The operand **n** must be between 0–7. Please refer to the Additional remark for more information.

5. Whether there is one four-digit seven-segment display or two four-digit seven-segment displays, and whether an output is a positive logic output or a negative logic output depends on **n**.

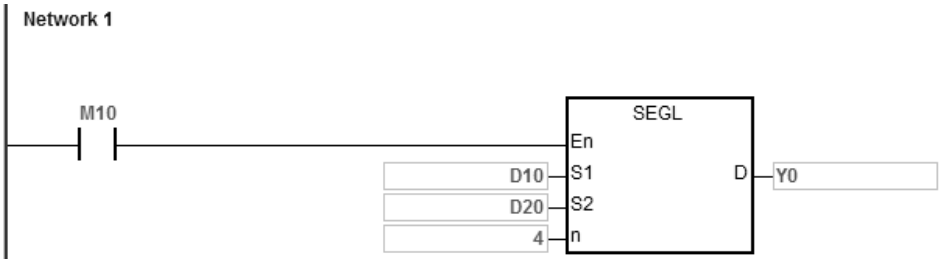
6. If there is one four-digit seven-segment display, eight outputs are occupied. If there are two four-digit seven-segment displays, twelve outputs are occupied.

7. When the instruction is executed, the outputs are ON cyclically. If the conditional contact switches from OFF to ON during the execution of the instruction, the outputs are ON cyclically again.

8. After the execution of the instruction is complete, SM693 is ON for a scan cycle.

Example

1. When M10 is ON, the instruction is executed. Y0–Y7 form a circuit. The instruction converts the value in D10 into the binary-coded decimal value, and shows the conversion result on first seven-segment display. The instruction converts the value in D11 into the binary-coded decimal value, and shows the conversion result on second seven-segment display. If the value in D10 or D11 exceeds 9,999, an operation error occurs.



2. When M10 is ON, Y4–Y7 are ON cyclically. It takes twelve scan cycles for Y0.4–Y0.7 to be ON. After the instruction completes, SM693 is ON for a scan cycle.

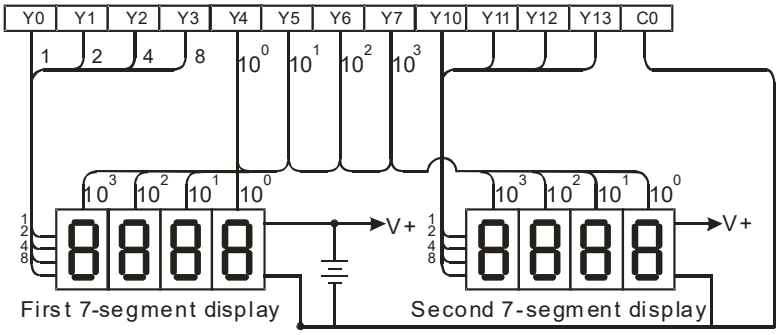
3. If there is one four-digit seven-segment display, **n** is between 0–3.

   - Connect the pins 1, 2, 4, and 8 in parallel, then connect them to Y0–Y3 on the PLC, and connect the latches to Y4–Y7 on the PLC.
   - When M10 is ON, the instruction is executed. Y4–Y7 are ON cyclically, and the value in D10 is shown on seven-segment display.

4. If there are two four-digit seven-segment displays, **n** is between 4–7.

   - Connect the pins 1, 2, 4, and 8 in parallel, then connect them to Y10–Y13 on the PLC, and connect the latches to Y4–Y7 on the PLC.
   - The value in D10 is shown on first seven-segment display, and the value in D11 is shown on second seven-segment display. If the values in D10 and D11 are 1234 and 4321 respectively, 1234 is shown on second seven-segment display.

5. The wiring:



Note: The transistor output module is used in this example.

Additional remarks

- Whether an output is a positive output or a negative output, and whether there is one four-digit seven-segment display or two four-digit seven-segment displays depend on **n**.
- The outputs on the PLC should be NPN transistors whose collectors are open collectors. In addition, an output has to connect a pull-up resistor to the DC power supply (less than 30 VDC). Therefore, when an output is ON, a signal of low potential is output.
- The following table shows the negative logic.

Binary-coded decimal value				Output (Binary-coded decimal code)				Signal			
b_3	b_2	b_1	b_0	8	4	2	1	A	B	C	D
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1	1	1	1	0
0	0	1	0	0	0	1	0	1	1	0	1
0	0	1	1	0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	0	1	0	1	1
0	1	0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0	1	0	0	1
0	1	1	1	0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	1	0	1	1	0

- The following table shows the positive logic.

Binary-coded decimal value				Output (Binary-coded decimal code)				Signal			
b_3	b_2	b_1	b_0	8	4	2	1	A	B	C	D
0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	0	0	1	0
0	0	1	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1	0	1	0	0
0	1	0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1	0	1	1	0
0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	1	1	0	1	0	0	1

- The following table shows the latch.

Positive logic		Negative logic	
Latch	Signal	Latch	Signal
1	0	0	1

- The following table shows the setting value of the parameter **n**.

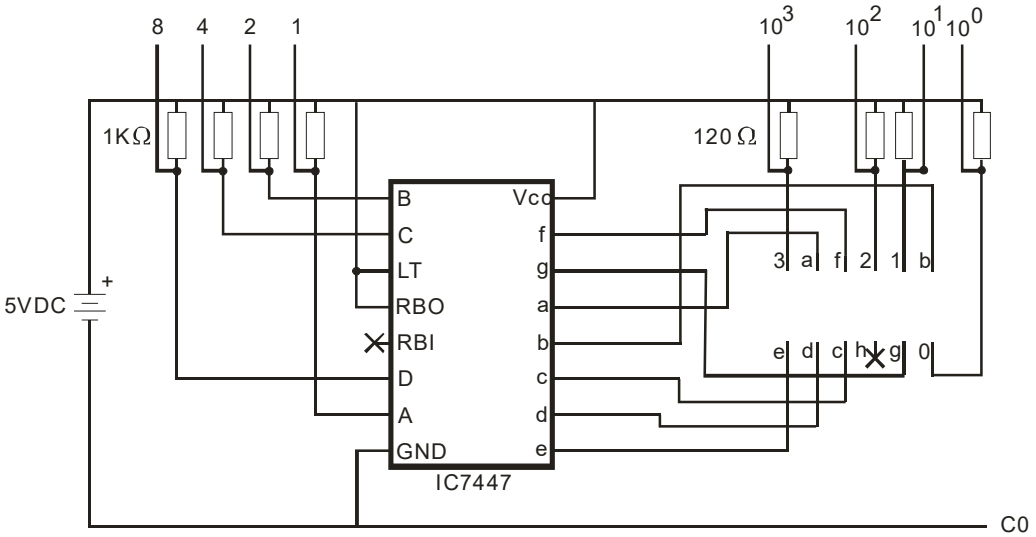
Number of seven-segment displays	One				Two			
Output (Binary-coded decimal code)	+		-		+		-	
Latch	+	-	+	-	+	-	+	-
n	0	1	2	3	4	5	6	7

‘+’: Positive logic

‘-’: Negative logic

- You can edit the parameters in **n** to modify the logics for the output transistor and the input of the seven-segment display.

- The following graphic shows the connection of the common-anode four-digit seven-segment display with IC 7447.



6.19 Communication Instructions

6.19.1 List of Communication Instructions

The following table lists the Communication instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1806	LRC	–	–	Longitudinal parity check
1807	CRC	–	–	Cyclic Redundancy Check
1808	MODRW	–	–	Reading and Writing Modbus data
1812	COMRS	–	–	Sending and receiving communication data
1813	COMDF	–	✓	Setting the communication format for a serial communication port
1814	VFDRW	–	–	Serial communication instruction exclusively for Delta AC motor drives
1815	ASDRW	–	–	Serial communication instruction exclusively for Delta servo drives
1816	CCONF	–	✓	Setting the parameters in the data exchange table for a communication port
1817	MODRWE	–	–	Reading and writing Modbus data without using any flags
1818	DNETRW	–	–	Reading and writing DeviceNet communication data
1819	CANRS	–	–	User-defined CAN communication sending and receiving
1820	DMVSH	–	–	Enabling Delta DMV detection and communication

6.19.2 Explanation of Communication Instructions

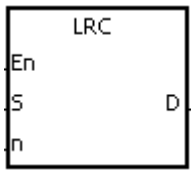
API	Instruction code	Operand	Function
1806	LRC	**S, n, D**	Longitudinal parity check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●	●							
n								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : First device to which the LRC is applied

n : Number of bytes

D : First device where the operation result is stored

Explanation

1. This instruction performs a longitudinal parity check on **n** bytes in the device specified by **S**. Please refer to the Additional remark for this instruction (below) for more information about the LRC check code.

2. The operand **n** must be an even number, and must be between 1–1000. If **n** is not in the range, an operation error occurs, the instruction is not executed, SM0 and SM1 are ON, and the error code in SR0 is 16#200B.

3. The 16-bit conversion mode: When SM606 is OFF, the instruction divides the hexadecimal data in the device specified by **S** into the high 8-bit data and the low 8-bit data. The instruction applies the LRC to every byte, and stores the operation result in the high 8-bit and the low 8-bit in the device specified by **D**. The number of bytes depends on **n**.

4. The 8-bit conversion mode: When SM606 is ON, the instruction divides the hexadecimal data in the device specified by **S** into the high 8-bit data (invalid data) and the low 8-bit data. The instruction applies the LRC to every byte, and stores the operation result in the low 8-bit in the two registers. The number of bytes depends on **n**. The values of the high 8 bits in the two registers are 0.

Example

1. The PLC is connected to the VFD-S series AC motor drive (ASCII modeSM210 is OFF; 8-bit mode: SM606 is ON.). The PLC sends the command, and reads the data in the six devices at the addresses starting from 16#2101 in the VFD-S series AC motor drive.

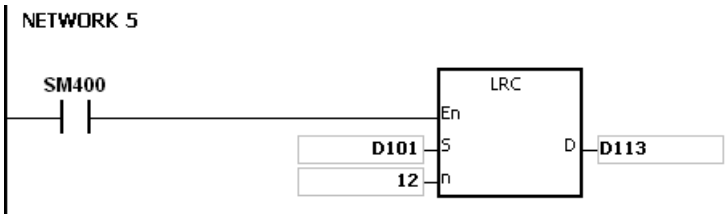
   PLC⇨VFD-S

   The PLC sends " : 01 03 2101 0006 D4 CR LF".

The PLC sends the data in the following table.

Register	Data		Description	
D100 Low 8 bits	' : '	16#3A	STX	
D101 Low 8 bits	'0'	16#30	ADR 1	AD (1, 0) is the station address of the AC motor drive.
D102 Low 8 bits	'1'	16#31	ADR 0	
D103 Low 8 bits	'0'	16#30	CMD 1	CMD (10) is the command code.
D104 Low 8 bits	'3'	16#33	CMD 0	
D105 Low 8 bits	'2'	16#32	Initial data address	
D106 Low 8 bits	'1'	16#31		
D107 Low 8 bits	'0'	16#30		
D108 Low 8 bits	'1'	16#31		
D109 Low 8 bits	'0'	16#30	Number of data (counted by the word)	
D110 Low 8 bits	'0'	16#30		
D111 Low 8 bits	'0'	16#30		
D112 Low 8 bits	'6'	16#36		
D113 Low 8 bits	'D'	16#44	LRC CHK 0	LRC CHK (01) is the error checking code.
D114 Low 8 bits	'4'	16#34	LRC CHK 1	
D115 Low 8 bits	CR	16#0D	END	
D116 Low 8 bits	LF	16#0A		

LRC CHK (01) above is the error checking code. You can use the LRC instruction to calculate it (8-bit mode: SM606 is ON).



LRC check code: 16#01+16#03+16#21+16#01+16#00+16#06=16#2C

The two's complement of 16#2C is 16#D4. 'D' (16#44) is stored in the low 8-bit in D113, and '4' (16#34) is stored in the low 8-bit in D114.

Additional remarks

1. The following table lists the format of the communication data in the ASCII mode.

STX	' : '	The start-of-text character is ' : ' (16#3A).
Address Hi	' 0 '	Communication address:
Address Lo	' 1 '	The 8-bit address is composed of two ASCII codes.
Function Hi	' 0 '	Function code:
Function Lo	' 3 '	The 8-bit function code is composed of two ASCII codes.
DATA (n-1)	' 2 '	Data: The n×8-bit data is composed of 2n ASCII codes.
.....	' 1 '	
	' 0 '	
	' 2 '	
	' 0 '	
DATA 0	' 0 '	
	' 0 '	
	' 2 '	
LRC CHK Hi	' D '	LRC check code:
LRC CHK Lo	' 7 '	The 8-bit check code is composed of two ASCII codes.
END Hi	CR	End-of-text character:
END Lo	LF	END Hi=CR (16#0D), END Lo=LF (16#0A)

2. LRC check code: The instruction adds up the values starting from the communication address to the data, then the instruction calculates the two's complement of the sum that is the LRC check code.

   Example 16#01+16#03+16#21+16#02+16#00+16#02=16#29

   The two's complement of 16#29 is 16#D7.

API	Instruction code	Operand	Function
1807	CRC	**S, n, D**	Cyclic Redundancy Check

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●	●							
n								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol

S : First device to which the CRC is applied

n : Number of bytes

D : First device in which the operation result is stored

Explanation

1. This instruction performs a cyclic redundancy check on **n** bytes starting with the device specified by **S**. Please refer to the Additional remark for this instruction (below) for more information about the CRC check code.

2. The operand **n** must be between 1–1000. If **n** is not in the range, an operation error occurs, the instruction is not executed, SM0 and SM1 are ON, and the error code in SR0 is 16#200B.

3. The 16-bit conversion mode: When SM606 is OFF, the instruction divides the hexadecimal data in the device specified by **S** into the high 8-bit data and the low 8-bit data. The instruction applies the CRC to every byte, and the stores the operation result in the high 8-bit and the low 8-bit in the device specified by **D**. The number of bytes depends on **n**.

4. The 8-bit conversion mode: When SM606 is ON, the instruction divides the hexadecimal data in the device specified by **S** into the high 8-bit data (invalid data) and the low 8-bit data. The instruction applies the CRC to every byte, and stores the operation result in the low 8-bit in the two registers. The number of bytes depends on **n**.

Example

1. The PLC is connected to the VFD-S series AC motor drive (RTU modeSM210 is ON; 16-bit mode: SM606 is ON.). The value 16#12, to be written into the device at 16#2000 in the VFD-S series AC motor drive, is written into the device in the PLC first.

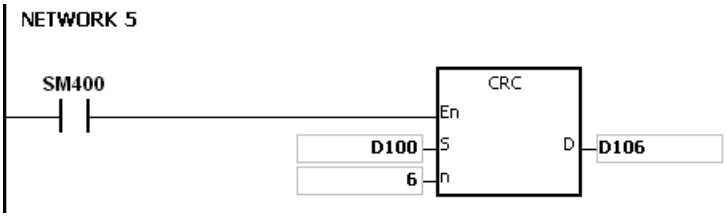
   PLC⇨VFD-S

   The PLC sends 01 06 2000 0012 02 07.

   The PLC sends the data as the following table shown.

Register	Data	Description
D100 Low 8 bits	16#01	Address
D101 Low 8 bits	16#06	Function
D102 Low 8 bits	16#20	Data address
D103 Low 8 bits	16#00	
D104 Low 8 bits	16#00	Data
D105 Low 8 bits	16#12	
D106 Low 8 bits	16#02	CRC CHK 0
D107 Low 8 bits	16#07	CRC CHK 1

   CRC CHK (01) above is the error checking code. You can calculate it with the CRC instruction (8-bit mode: SM606 is ON).



CRC check code: 16#02 is stored in the low 8-bit in D106, and 16#07 is stored in the low 8-bit in D107.

Additional remarks

1. The following table shows the format of the communication data in RTU mode.

START	Time interval
Address	Communication address: 8-bit binary address
Function	Function code: 8-bit binary code
DATA (n-1)	Data: n×8-bit data
…….	
DATA 0	
CRC CHK Low	CRC check code: The 16-bit check code is composed of two 8-bit binary codes.
CRC CHK High	
END	Time interval

2. CRC check code: The check code starts from the address to the data. The operation rule is shown in the following table.

   Step 1: Suppose the data in the 16-bit register (the register where the CRC check code is stored) is 16#FFFF.

   Step 2: The logical operator XOR takes the first 8-bit message and the low 8-bit data in the 16-bit register, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit register.

   Step 3: The values of the bits in the 16-bit registers are shifted by one bit to the right. The value of the highest bit becomes 0.

   Step 4: If the value of the right-most bit that is shifted to the right is 0, the data from step 3 is stored in the 16-bit register. Otherwise, the logical operator XOR takes 16#A001 and the data in the 16-bit register, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit register.

   Step 5: Repeat step 3 and step 4, and perform the operation on the 8-bit message.

   Step 6: Repeat step 2–5, and then get the next 8-bit message. Perform the operations on all messages. The final result in the 16-bit register is the CRC check code. Notice that the low 8-bit data in the 16-bit register is interchanged with the high 8-bit data in the 16-bit register before the CRC check code is put into the check code of the message.

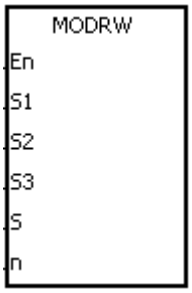
API	Instruction code	Operand	Function
1808	MODRW	**S_1, S_2, S_3, S, n**	Reading and Writing Modbus data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S								●								
n								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S	●	●			●	●							
n		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Unit address

S_2 : Function code

S_3 : Device address

S : Register for reading/writing the data

n : Data length

Explanation

1. The operand **S_1** must be between 0–254; 0 is the broadcasting mode.

2. **S_2** is the function code.

   The following table shows an example.

Function code	Description	Data length	Devices that support devices for slaves
16#01	PLC reads the data from several bit devices.	1-1600	X, Y, M, SM, S, T, C, HC
16#02	PLC reads the data from several bit devices.	1-1600	X, Y, M, SM, S, T, C, HC
16#03	PLC reads the data from several word devices.	1-100	X, Y, SR, D, T, C, HC, E

Function code	Description	Data length	Devices that support devices for slaves
16#04	PLC reads the data from several word devices.	1-100	X
16#05	PLC writes the state into a bit device.	1	Y, M, SM, S, T, C, HC
16#06	PLC writes the data into a word device.	1	Y, SR, D, T, C, HC, E
16#0F	PLC writes the states into several bit devices.	1-1600	Y, M, SM, S, T, C, HC
16#10	PLC writes the data into several word devices.	1-100	Y, SR, D, T, C, HC, E

The instruction supports only the function codes mentioned above, and cannot execute other function codes. Please refer to the examples below.

3. **S_3** is the device address. If the address is invalid for the designated communication device, the communication device responds with an error message. For example, the device address 16#8000 is invalid in the DVP-ES2.

4. **S** is the register involved in the reading/writing the data.
   The data to be written into the external equipment is stored in the register in advance.

   The data to be read from the external equipment is stored in the register.

5. **N** is the length of the data

   For word-type communication function codes, the data length cannot exceed 100 words.
   For bit-type (BOOL) communication function codes, the data length is between 1–1600 bits.

6. The following table shows how the functions of **S_3**, **S**, and **n** vary with the function code used.

Function code	S_3	S	n
16#01	Address from where the data is read	Register where the data read is stored	Length of data read
16#02	Address from where the data is read	Register where the data read is stored	Length of data read
16#03	Address from where the data is read	Register where the data read is stored	Length of data read
16#04	Address from where the data is read	Register where the data read is stored	Length of data read
16#05	Address into where the	Status value written	No meaning

Function code	S_3	S	n
	data is written		
16#06	Address into where the data is written	Register where the data written is stored	No meaning
16#0F	Address into where the data is written	Register where the data written is stored	Length of data written
16#10	Address into where the data is written	Register where the data written is stored	Length of data written

7. There is no limitation on the number of times you can use this instruction; however only one instruction can be executed on the same COM port at a time. You need to decide and use the sending flag for the COM to be used before executing this instruction. Otherwise, the data from 2 different communication COMs may be mixed up.

8. If a communication timeout occurs, the timeout flags are ON. After you solve the problem, you must reset timeout flags to OFF. When using this instruction, the timeout value cannot 0. Set the value between 100–32767ms; when the value is set to 0, it is processed as 200 ms.

9. In Modbus ASCII mode, you need to set up only the data (non-ASCII mode) for transmission. The instruction converts the non-ASCII mode to the ASCII mode, consisting of the head code (:), the converted ASCII code, checksum (LRC) and tail code (CRLF). The instruction stores the data received in ASCII character in the internal register. The PLC automatically converts the data into the hexadecimal value, and if the communication data is correct, stores the conversion result in **S**. and sets the completion flag SM to ON.

10. In Modbus RTU mode, you need to set up only the data for transmission. The instruction adds the checksum (CRC) and the stores the data received in ASCII character in the internal register. The PLC automatically converts the data into the hexadecimal value, and if the communication data is correct, stores the conversion result stored in **S**.

11. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Communication protocol setup example

1. The following examples use PLC communication port 1 and special registers to demonstrate how to setup a communication protocol.

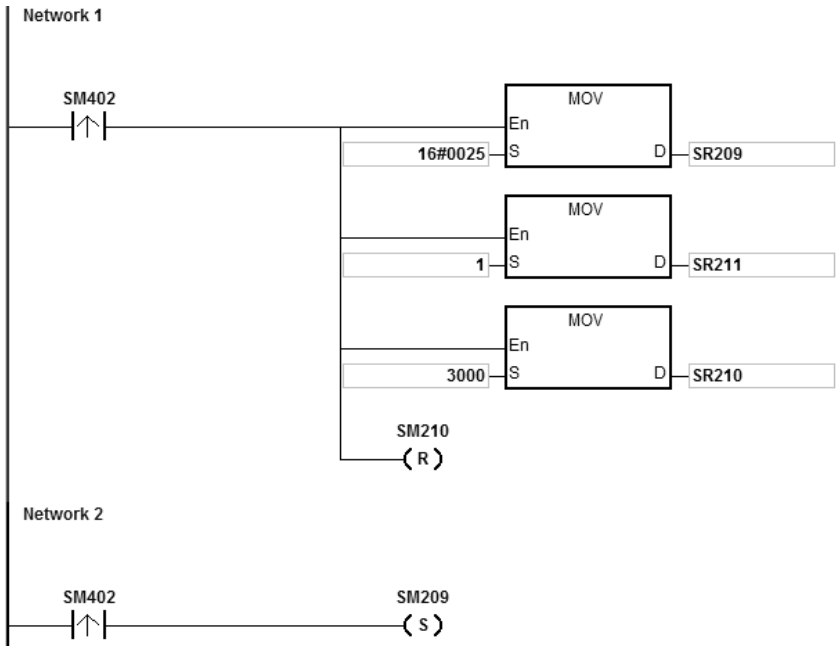
2. You can set up the PLC communication port with HWCONFIG in ISPSoft, or with the relative special registers, or you can use the COMDF instruction (API 1813) to set up the communication. Please refer to the ISPSoft manual for setting it up in HWCONFIG. For communication register setups (SM, SR), please refer to section 6.19.3 for more details.

3. The communication setup for this example is RS485 ASCII , 9600, 8, E, 1 (SR209=16#0025).

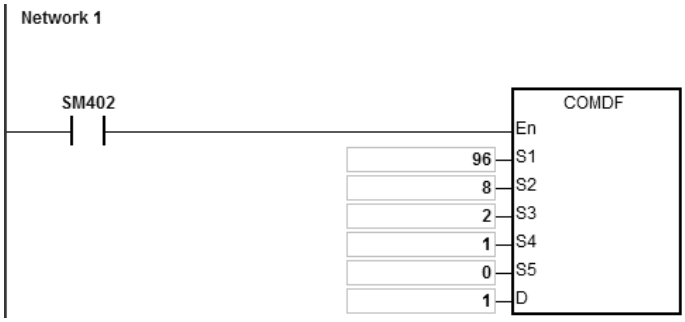
4. Set the communication timeout to 3000ms (SR210=3000).

5. Set the communication mode to ASCII mode (SM210=OFF).

6. Enable the communication protocol (SM209=ON).



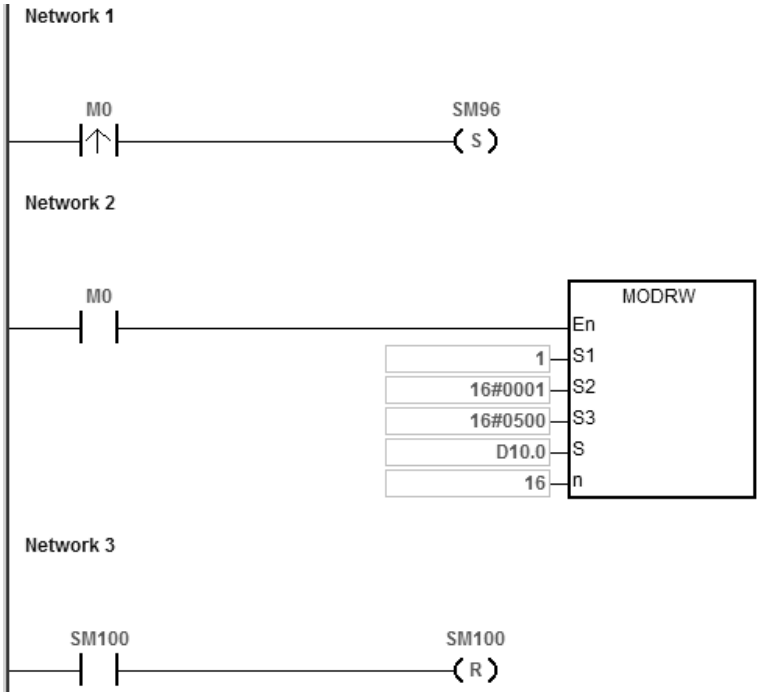
If you set up the communication port with the COMDF instruction (API 1813), you can ignore this step.



If you set up the communication port in ISPSoft - > HWCONFIG - > COM Port, you can ignore this step.

Example 1

1. Function code 01 (16#01): the PLC reads the data from several bit devices that are not discrete input devices (16 pieces of data is read in this example). For function code 02, the operation is the same as for function code 01.



ES3 Series CPU module is connected to the DVP-ES2 Series PLC.

When SM96 and M0 are on, the ES3 Series CPU sends and receives the Y0–Y17 commands from the DVP-ES2.

When the address of Y0 is 16#0500, the states of Y0–Y17 in DVP-ES2 are listed in the following table.

Device	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

The following table lists the operands for the MODRW instruction.

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0001
S_3	Device address	16#0500

S	Register for reading and writing the data	D10.0
n	Data length	16

ASCII mode

You do not need to convert the ASCII codes and they are all expressed in 16# values.

- ES3 sends the communication command: " : 01 01 05 00 00 10 E9 CR LF".
- ES3 receives the communication command: " : 01 01 02 D2 04 26 CR LF".

RTU mode

- ES3 sends the communication command: "01 01 05 00 00 10 3D 0A".
- ES3 receives the communication command: "01 01 02 D2 04 E4 9F".

If the format is correct, SM100 is ON.

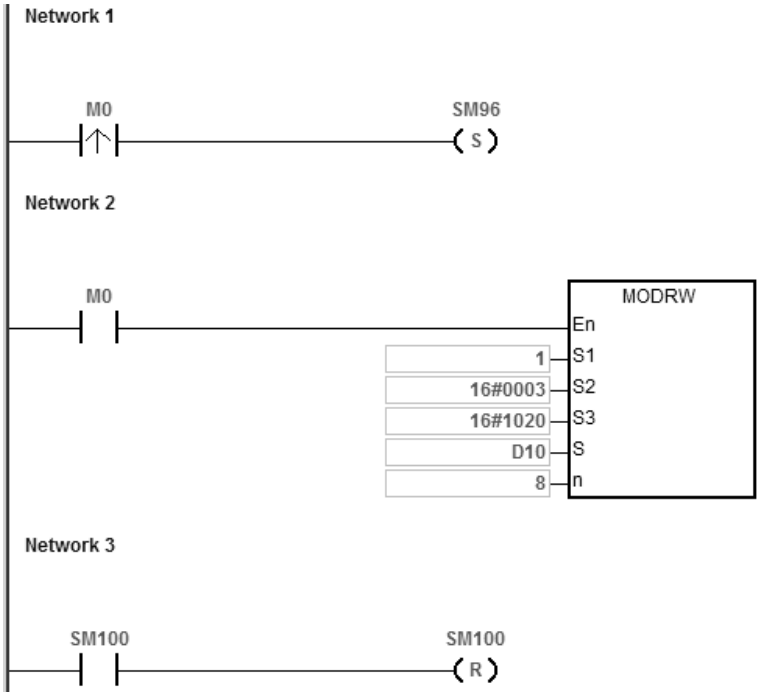
2. The response messages from the DVP-ES2 are stored in registers D10.0 to 10.15 (the data read is D10.15–D10.0=16#04D2).

Device	D10.7	D10.6	D10.5	D10.4	D10.3	D10.2	D10.1	D10.0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	D10.15	D10.14	D10.13	D10.12	D10.11	D10.10	D10.9	D10.8
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

3. After the receiving the data sent back from the DVP-ES2, the PLC confirms the data format sent back from DVP-ES2 and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON.

Example 2

1. Function code 03 (16#03): the PLC reads the data from several bit devices that are not discrete input devices (eight pieces of data is read in this example). For function code 04, the operation is the same as for function code 03.



2. ES3 Series CPU module is connected to the DVP-ES2 Series PLC.
   When SM96 and M0 are on, the ES3 Series CPU module sends and receives D32–D39 from the DVP-ES2.

3. When the address of D32 is 16#1020, the values of D32–D39 in DVP-ES2 are listed in the following table.

Device	D32	D33	D34	D35	D36	D37	D38	D39
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

The following table lists the operands of the MODRW instruction.

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0003
S_3	Device address (D32)	16#1020
S	Register involved for reading and writing the data	D10
n	Data length	8

ASCII mode

You do not need to convert the ASCII codes, and they are all expressed in 16# values.

- ES3 sends the communication command: “：01 03 10 20 00 08 C4 CR LF”.
- ES3 receives the communication command: “：01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”.

RTU mode

- ES3 sends the communication command: “01 03 10 20 00 08 41 06”.
- ES3 receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”.

If the format is correct, SM100 is ON.

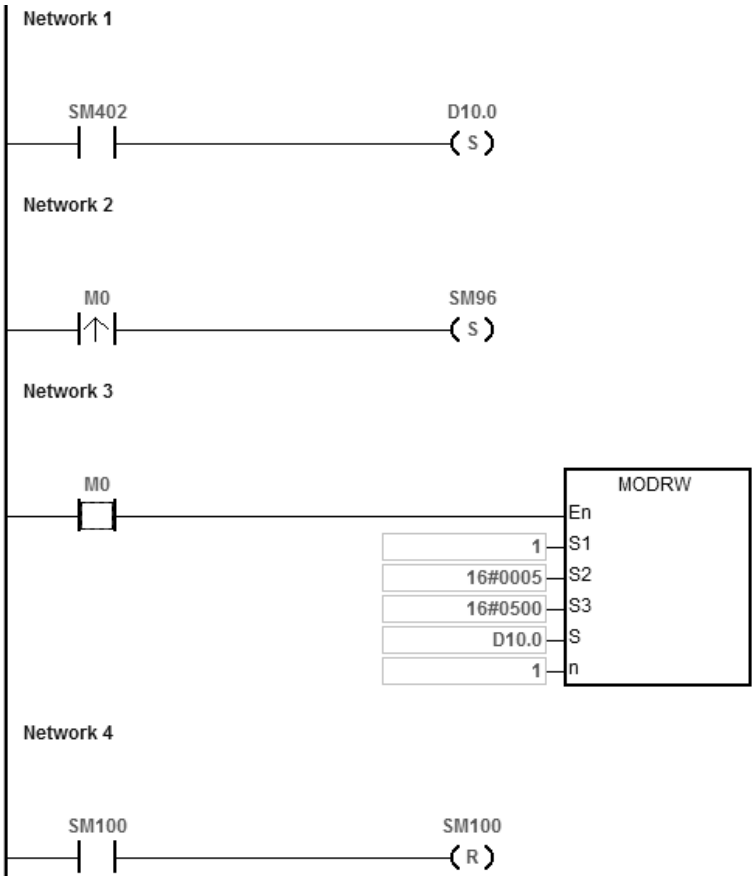
- The response messages from the DVP-ES2 is stored in registers D10 to D17.
- The following table lists the values in D10–D17.

Device	D10	D11	D12	D13	D14	D15	D16	D17
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

4. After the receiving the data sent back from the DVP-ES2, the PLC confirms the data format sent back from DVP-ES2 and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON.

Example 3

1. Function code 05 (16#05): the PLC writes the state into a bit device. The device is set to ON in this example.



2. The ES3 Series CPU module is connected to the DVP-ES2 series PLC. D10.0 is ON and Y0 in the DVP-ES2 Series PLC is also ON. When SM96 and M0 are ON, the PLC can set the state of Y0.

   The following table lists the operands for the MODRW instruction.

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0005
S_3	Device address	16#0500
S	Register for reading and writing the data	D10.0
n	Data length (not used with this function code)	1

ASCII mode

The numbers below are only for reference. Instead of showing the values in the ASCII codes, here the expressions are shown in 16# values.

- ES3 sends the communication command: “ : 01 05 05 00 FF 00 F6 CR LF”
- ES3 receives the communication command: “ : 01 05 05 00 FF 00 F6 CR LF”

RTU mode

- ES3 sends the communication command: “01 05 05 00 FF 00 8C F6”
- ES3 receives the communication command: “01 05 05 00 FF 00 8C F6”

If the format is correct, SM100 is ON.

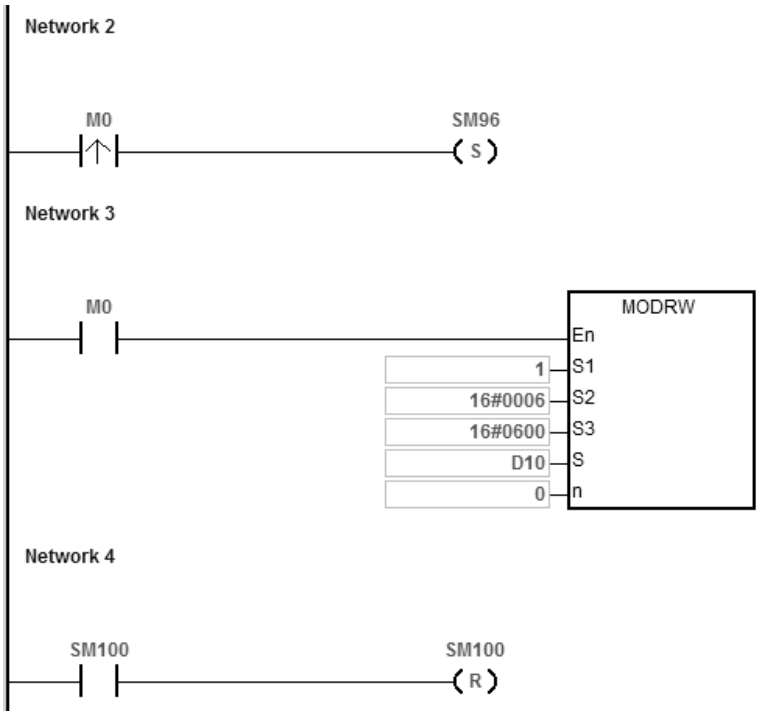
3. After receiving the data from the DVP-ES2, the PLC confirms the data format sent back from the DVP-ES2 and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON.

4. When the DVP-ES2 receives this instruction, the Y0 is ON.

5. Since this function code writes data, the operand **n** is ignored.

Example 4

1. Function code 06 (16#06): the PLC writes the state into a word device.



2. ES3 Series CPU module is connected to the DVP-ES2 series PLC.

3. Suppose D10 is 16#55AA (waiting to write data to the device T0 of the DVP-ES2).

   When SM96 and M0 are ON, the PLC can write data to the T0 of the DVP-ES2 series PLC. The address of T0 is 16#0600.

   The following table lists the operands for the MODRW instruction.

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0006
S_3	Device address of T0	16#0600
S	Register T0 for reading and writing the data	D10
n	Data length (not used with this function code)	0

ASCII mode

You do not need to convert the ASCII codes, and they are all expressed in 16# values.

- ES3 sends the communication command: " : 01 06 06 00 55 AA F4 CR LF"
- ES3 receives the communication command: " : 01 06 06 00 55 AA F4 CR LF"

RTU mode

- ES3 sends the communication command: "01 06 06 00 55 AA 36 6D"
- ES3 receives the communication command: "01 06 06 00 55 AA 36 6D"

If the format is correct, SM100 is ON.

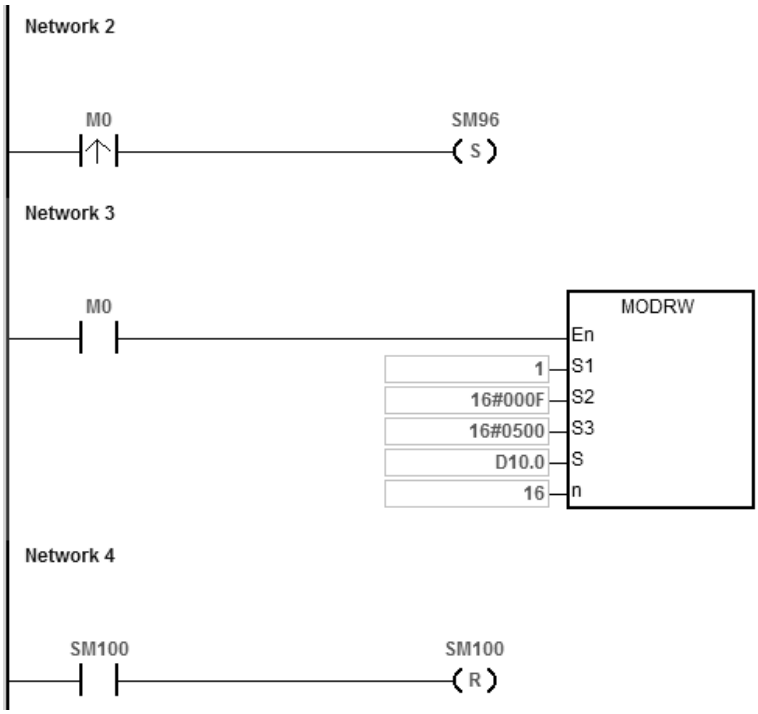
4. After receiving the data from the DVP-ES2, the PLC confirms the data format sent back from the DVP-ES2 and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON.

5. When the DVP-ES2 receives this instruction, it writes the data stored in the device D10 to the device T0 in the DVP-ES2.

6. Since this function code writes data, the operand **n** is ignored.

Example 5

1. Function code 0F (16#0F): the PLC writes the states into several bit devices.



2. ES3 Series CPU module is connected to the DVP-ES2 series PLC.

- Suppose D10.15-D10.0=16#04D2 (waiting to write the state of Y0-Y17 of the DVP-ES2)

Device	D10.7	D10.6	D10.5	D10.4	D10.3	D10.2	D10.1	D10.0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	D10.15	D10.14	D10.13	D10.12	D10.11	D10.10	D10.9	D10.8
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

When SM96 and M0 are ON, the PLC can set the state of Y0-Y17 in the DVP-ES2. The address of Y0 is 16#0500.

The following table lists the operands for the MODRW instruction.

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#000F
S_3	Device address of Y0	16#0500
S	Registers Y0–Y17 for reading and writing the data	D10.0
n	Data length	16

ASCII mode

You do not need to convert the ASCII codes, and they are all expressed in 16# values.

- ES3 sends the communication command: " : 01 0F 0500 0010 02 D2 04 03 CR LF"
- ES3 receives the communication command: " : 01 0F A0 00 00 10 40 CR LF"

RTU mode

- ES3 sends the communication command: "01 0F 05 00 00 10 02 D2 04 EA 43"
- ES3 receives the communication command: "01 0F A0 00 00 10 76 07"

If the format is correct, SM100 is ON.

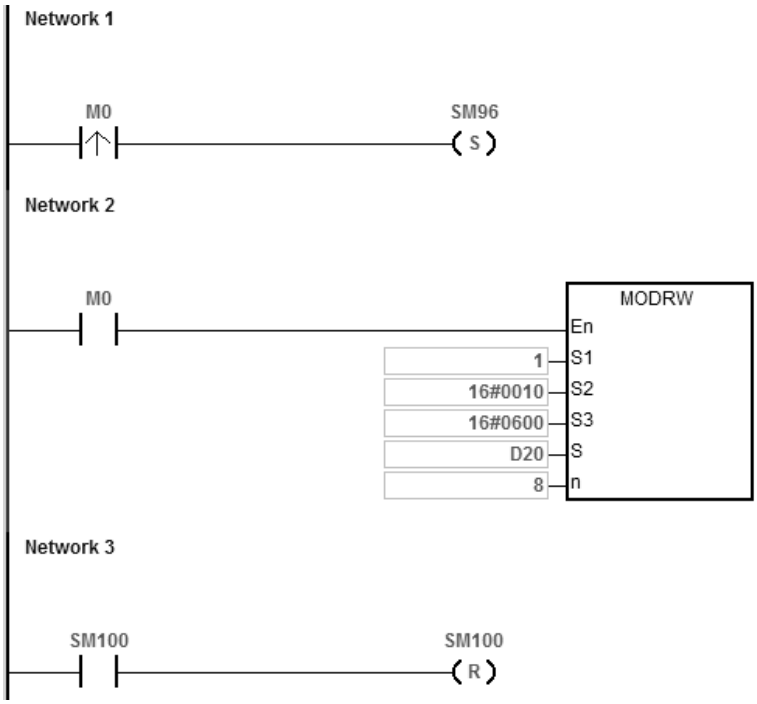
3. After receiving the data sent back from the DVP-ES2, the PLC confirms the data format sent back from the DVP-ES2 and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON.

Device	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
State	ON	ON	OFF	ON	OFF	OFF	ON	OFF
Value	D				2			
Device	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
State	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
Value	0				4			

4. Since this function code writes data, the operand **n** is ignored.

Example 6

1. Function code 10 (16#10): the PLC writes the states into several word devices.



2. ES3 Series CPU module is connected to the DVP-ES2 series PLC.

3. Suppose the values for D20–27 are listed in the following table (waiting to write data to the devices T0–7 of the DVP-ES2).

Device	D20	D21	D22	D23	D24	D25	D26	D27
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

When SM96 and M0 are ON, the PLC can write data to the T0–7 in the DVP-ES2 series PLC. The address of T0 is 16#0600.

The following table lists the operands of the MODRW instruction.

Operand	Description	Device
S_1	Unit address	1
S_2	Function code	16#0010
S_3	Device address of T0	16#0600
S	Register T0–7 for reading and writing the data	D20
n	Data length (not used with this function code)	8

ASCII mode

You do not need to convert the ASCII codes, and they are all expressed in 16# values.

- ES3 sends the communication command: " : 01 10 0600 00 08 10 1234 5678 1122 3344 5566 7788 99AA BBCC 8F CR LF"
- ES3 receives the communication command: " : 01 10 06 00 00 08 E1 CR LF"

RTU mode

- ES3 sends the communication command: "01 10 06 00 00 08 10 1234 5678 1122 3344 5566 7788 99AA BBCC 0B 0C"
- ES3 receives the communication command: "01 10 06 00 00 08 C1 47"

If the format is correct, SM100 is ON.

5. After receiving the data sent back from the DVP-ES2, the PLC confirms the data format sent back from the DVP-ES2, and determines if it is correct. If no error occurs in the format, the corresponding special flags SM100 are ON; if not SM102 is ON. When the DVP-ES2 receives this instruction, it writes data stored in the devices D20–27 to the device T0–7 in the DVP-ES2.

Device	T0	T1	T2	T3	T4	T5	T6	T7
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

6. Since this function code writes data, the operand **n** is ignored.

Additional remarks

1. If the value in $\mathbf{S_1}$ or $\mathbf{S_2}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the device specified by **S** is not sufficient to contain the **n** pieces of data, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **n** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. If the function code specified by $\mathbf{S_2}$ is related to bit devices, the device specified by **S** must be a bit device; otherwise, an operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.
5. If the function code specified by $\mathbf{S_2}$ is related to word devices, the device specified by **S** must be a word device; otherwise, an operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.
6. If the communication command is 0x05 or 0x06, the value in **n** can be ignored. The length of the data is only one bit

or one word.

7. The MODRW instruction is not executed if the sending flags SM96 and SM97 are not ON.

8. If a communication timeout occurs, the timeout flags SM104 and SM105 are ON, and the receiving flags SM98 and SM99 are OFF.

9. If an error occurs while receiving data, the error flags SM102 and SM103 are ON, and the receiving flags SM98 and SM99 are OFF.

10. If the function code specified by **S_2** is related to word devices, the device in the external equipment with which the PLC communicates must be a word device. If the function code specified by **S_2** is related to bit devices, the device in the external equipment with which the PLC communicates must be a bit device.

11. Please refer to section 6.19.3 for more details on communication register setups (SM, SR).

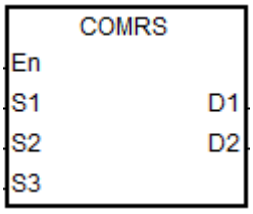
API	Instruction code		Operand	Function
1812	COMRS		**S_1, S_2, S_3, D_1, D_2**	Sending and receiving communication data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●							
S_3								●	●				○	○		
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D_1		●			●	●							
D_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Communication port number (1–2)

S_2 : Source of the data to be sent

S_3 : Length of the data to be sent

D_1 : First device where communication data received is stored

D_2 Condition for ending receiving data

Explanation

1. **S_1** is a communication port number: COM1 is number 1 and COM 2 is number 2. If the data is out of the communication port range, the instruction does not execute any sending or receiving.

2. If you use a specific character or characters to end receiving data, it is suggested that you apply the instruction to ASCII data. If you do not apply the instruction to ASCII data, it is suggested that you use a timeout period to end receiving data.

3. **S_2** is the source of the data to be sent.

   S_3 is the length of the data to be sent.

If **S_2** is D100 and **S_3** is 10, the instruction sends the values in the low bytes in D100–D109 through the communication port specified by **S_1**.

4. No strings are sent if the setting value in **S_3** is 0. The maximum number of characters that can be sent is 256 words.

5. **D_1** is the length of the data that is received.

   D_1+1–**D_1**+n are the devices to store the data that is received

   If **D_1** is D200, the value in **D_2** is 3, and the value in **D_2**+1 is 16#0D0A, the instruction stores data received in the low bytes in the devices starting from D201 (the high bytes is unchanged) The instruction continues to receive data until it receives the consecutive stop characters 16#0D and 16#0A. The instruction writes the length of the data received to D200 after receiving 16#0D and 16#0A, and sets a completion flag to ON after the receiving data stops.

6. **D_2** is the mode for receiving data

   D_2+1 is the condition that ends receiving of data

   D_2 and **D_2**+1 are described in the following table.

D_2	Mode for receiving data	Setting value in D_2+1	Remark
0	Not receiving communication data	Unused	After the sending of data is complete, set a completion flag to ON.
1	When the time after the last piece of data received exceeds the time set in **D_2**+1, the receiving of data is complete.	The setting value in **D_2**+1 is time. The unit of measurement is 1 millisecond. The setting value in **D_2**+1 is between 5–3000.	If the time that you set is greater than 3000 milliseconds, the value in **D_2**+1 is 3000. If the time that you set is less than 5 milliseconds, the value in **D_2**+1 is 5.
2	The data received ends with a specific character.	The setting value in **D_2**+1 is a specific character.	If a specific character is 16#0A, the value in **D_2**+1 is 16#000A.
3	The data received ends with two consecutive specific characters.	The setting value in **D_2**+1 is two specific characters.	If two specific characters are 16#0D and 16#0A, the value in **D_2**+1 is 16#0D0A.
4	The data received starts with a specific character. When the time after the last piece of data is received exceeds the time set in **D_2**+1, the receiving of data is	A specific character is stored in the high byte in **D_2**+1, and the time is stored in the low byte in **D_2**+1. The time set in the low byte in **D_2**+1 is in the range of	If a start character is 16#3A, and time is 15 milliseconds, the value in **D_2**+1 is 16#3A0F.

	complete.	5–255 milliseconds.	
5	The data received starts with a specific character, and ends with a specific character.	The setting value in **D_2**+1 is a specific start character, and a specific end character.	If a start character is 16#3A, and a stop character is 16#0A, the value in **D_2**+1 is 16#3A0A.
6	A specific quantity of data is received.	The setting value in **D_2**+1 is the length of the data to receive. The setting value is between 1–256.	If you want to receive 10 characters, set the value in **D_2**+1 to 10.
7	The data received ends with a specific character and generates communication interrupts.	The setting value in **D_2**+1 is a specific end character.	If an end character is 16#0A, the value in **D_2**+1 is 16#000A.
8	Set the quantity of data received and then generate communication interrupts.	The setting value in **D_2**+1 is the length of the data received. The setting value is between 1–256.	If you want to receive 10 characters, set the value in **D_2**+1 to 10.
9	The data received ends with a specific character or a specific quantity of data received; when either condition is met, the transmission is complete.	A specific end character is stored in the high byte in **D_2**+1, and the time is stored in the low byte in **D_2**+1. The time set in the low byte in **D_2**+1 must be between 1–255 milliseconds.	If an end character is 16#0A, and time is 15 milliseconds, the data length is 15 words, the value in **D_2**+1 is 16#0A0F.
Others	If the mode used is not supported, the instruction is not executed.		

7. Except for mode 6 and 8, when data received in **D_2** exceeds the maximum range of the received data length (256 words) and no ending character is received, the instruction stops executing and treats this operation as a receiving error. **D_1**+0 is 0 and **D_1**+1–**D_1**+n do not store the received data.

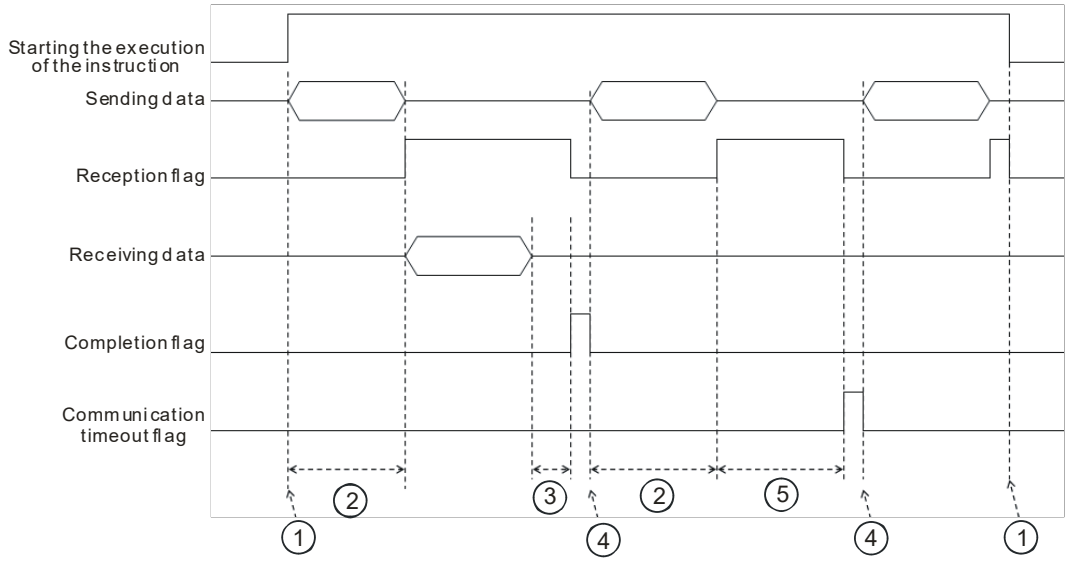
8. The interactions among the communication port, the related special auxiliary relays, and the related special data register are described in Section 6.19.3.

9. Timing diagrams

   - Mode for receiving data: 0

     When data is sent, you cannot cancel the sending of data. If the conditional contact preceding the instruction is not enabled, the data will still be sent, but the completion flag will not be set to ON after sending of the data is complete.

   - Mode for receiving data: 1 or 4



Description:

①→ Start/stop the execution of the instruction.

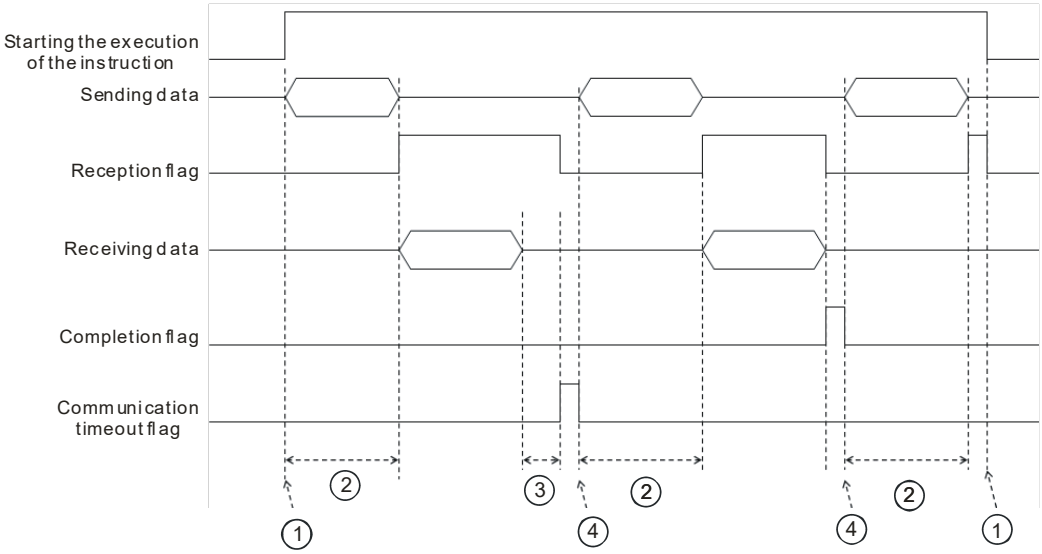
②→ Time during which data is sent. The period of time in which data is sent is not measured.

③→ After the first character is received, the time that passes before the next character is received is measured. Whenever a character is received, the instruction clears the time measured. The completion flag is not be set to ON until the time measured is greater than the setting value in **D_2**+1.

④→ If the instruction is still enabled after you reset the completion flag or the communication flag, the next communication data is sent automatically when the instruction is scanned in the next cycle.

⑤→ When the PLC begins to receive data, it begins to measure the time that passes. It does not set the communication timeout to ON until the time measured exceeds the timeout period. It is suggested that you set the timeout period to be longer than the time set in **D_2**+1.

- Mode for receiving data: 2, 3, 5, 6, or 9.



Description:

① → Start/stop the execution of the instruction.

② → Time during which data is sent. The period of time in which data is sent is not measured.

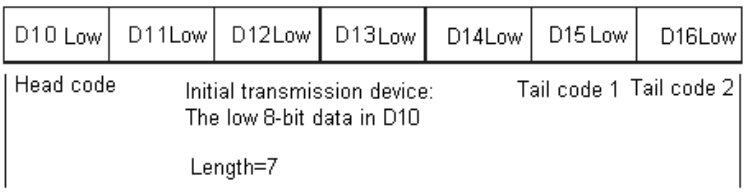
③ → After the first character is received, the time that passes before the next character is received is measured. Whenever a character is received, the instruction clears the time measured. The communication timeout flag is not set to ON until the time measured exceeds the timeout period.

④ → If the instruction is still enabled after you reset a completion flag or a communication flag, the next communication data is sent automatically when the instruction is scanned in the next cycle.

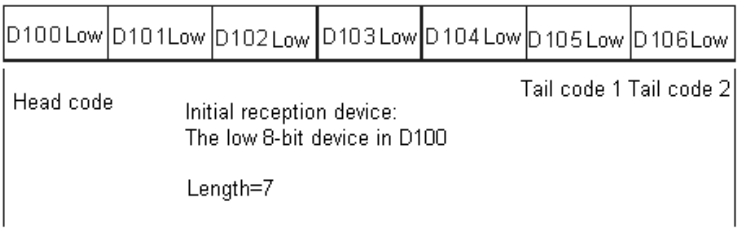
10. Mode for sending data / Mode for receiving data

8-bit mode: The command that is edited is stored in the initial transmission device, and the command to be sent includes the head code and the tail code. The instruction divides the 16-bit data into the high 8-bit data and the low 8-bit data. The instruction ignores the high 8-bit data, and can send or receive the low 8-bit data can be sent or received. Take standard Modbus for example.

Sending the data: (PLC→external equipment)

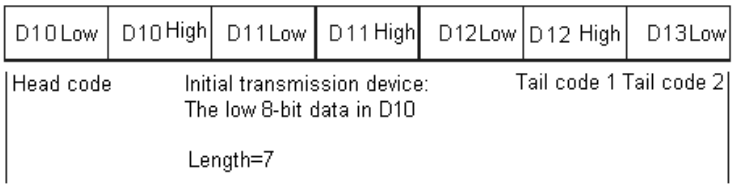


Receiving the data: (External equipment→PLC)

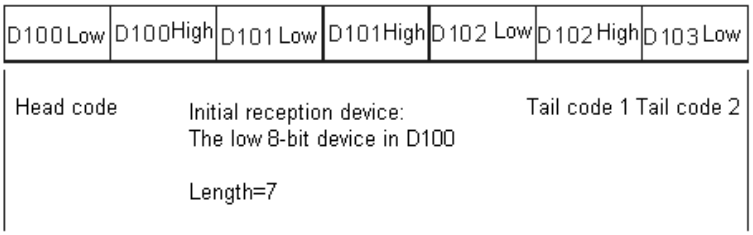


16-bit mode: The command that is edited is stored in the initial transmission device, and the command to be sent includes the head code and the tail code. When SM106/SM107 is OFF, the instruction divides the 16-bit data into the high 8-bit data and the low 8-bit data.

Sending the data: (PLC→external equipment)



Receiving the data: (External equipment→PLC)



The data that the PLC receives from the external equipment includes the head and the tail code; therefore, you have to be aware of the setting for the length.

11. When the mode is 7 or 8, the corresponding communication port and the interrupt number are listed in the following table.

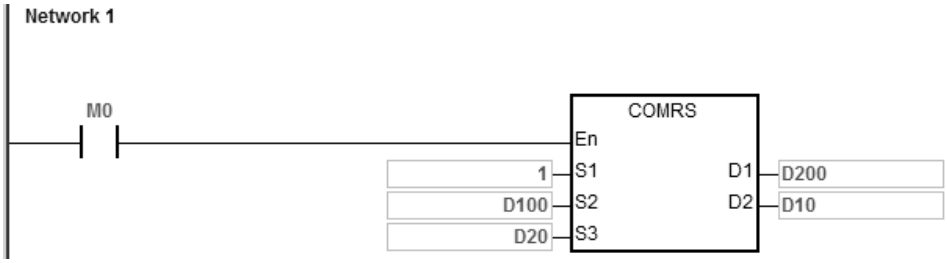
Communication port number	COM1	COM2	Card1	Card2
Interruption number	I300	I302	I304	I306

12. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

The following examples use COM1 (RS485).

Example 1

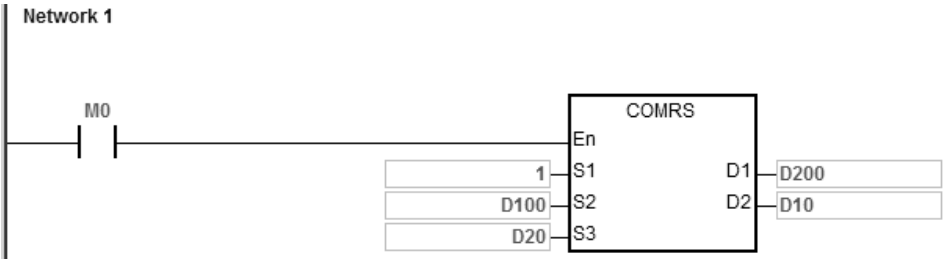
The mode in **D_2** is 0 (not receiving communication data) and you set the mode for sending and receiving data to 8-bit mode (SM106=ON).



1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
3. Set D10=16#0000 (sending data only, not receiving data).
4. Enable the contact M0.
5. The PLC sends 4 pieces of data.
6. Sending data: PLC→external equipment 31 32 33 34.
7. Since receiving data is not required, after the PLC sends out the data, the operation ends, and SM100=0.
8. To send more data, set the flag SM100 to OFF to start the operation again.

Example 2

The mode in **D_2** is 1 (setting the timeout value to 5–3000 ms) and you set the mode for sending and receiving data to 16-bit mode (SM106=OFF).



1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#3231, D101=16#3433.

3. Set D10=16#0001 (mode: 1), D11=300 (set the timeout value to 300 ms).

4. Enable the contact M0.

5. PLC sends 4 pieces of data.

6. Sending data: PLC→external equipment 31 32 33 34.

7. After the external equipment receives the data from the PLC, it sends 5 consecutive data to the PLC, each sent in less than 20 ms. External equipment→PLC 35 36 37 38 39.

8. D200=5 (number of data received), and the content of data received: D201=16#3635, D202=16#3837, D203=16#0039.

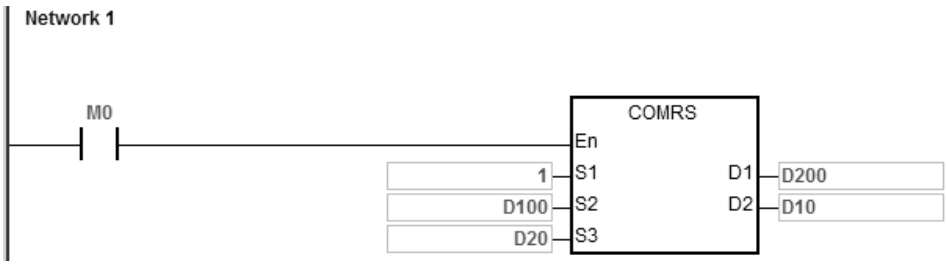
9. SM100=ON: reception of data is complete.

10. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the sending of data is complete, the receiving flag SM98 is ON, and then the PLC starts to receive data. You set the timeout between each data reception in D11. When the interval time exceeds the set timeout and no data is received, SM100 is ON.

Example 3

The mode in **D_2** is 2 (the data received ends with a specific character.) and you set the mode of sending and receiving data to 8-bit mode (SM106=ON).



1. Set the length of the data to be sent: D20=0, meaning the PLC will not send data but only receives data.

2. Set D10=16#0002 (mode: 2), D11=16#000A (the ending character is 16#0A).

3. Enable the contact M0.

4. The PLC waits to receive data from the external equipment. D20=0 means that the PLC does not send data to the external equipment.

5. The external equipment sends data to the PLC.

   External equipment → PLC 31 32 33 34 35 0A.

6. D200=6 (number the data received), the content of data received: D201=16#0031, D202=16#0032, D203=16#0033, D201=16#0034, D202=16#0035, D203=16#000A

7. SM100=ON: reception of data is complete.

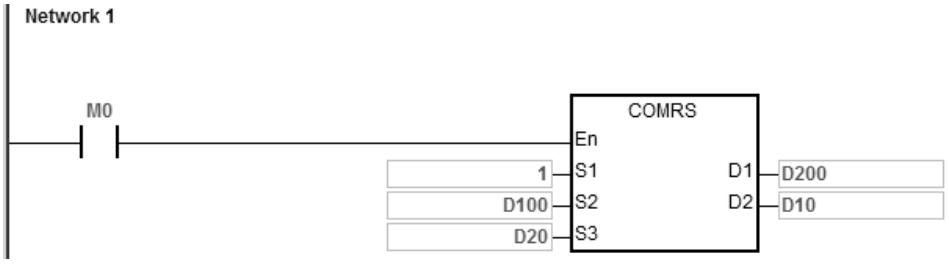
8. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the sending of data is complete, the receiving flag SM98 is ON and then the PLC starts receiving data until receiving the ending character (16#0A). When the reception of data is complete, SM100 is ON. If the communication timeout occurs but the ending character (16#0A) is still not received, the communication timeout flag SM104 is ON.

Example 4

The mode in **D_2** is 3 (the data received ends with two specific characters.) and you set the mode for sending and receiving data to 16-bit mode (SM106=OFF).

This example uses a DVP-ES2 as the external equipment and writes H1234 to D100 in the DVP-ES2.



1. The length for the data to be sent: D20=17.

2. The contents for the data to be sent: D100=16#303A, D101=16#3031, D102=16#3136, D103=16#3630, D104=16#3134, D105=16#3332, D106=16#3334, D107=16#0D46, D108=16#000A.

3. Set D10=16#0003 (mode: 3), D11=16#0D0A (the ending characters are 16#0D and 16#0A).

4. Enable the contact M0.

5. The PLC sends 17 pieces of data.

   Sending data: PLC→external equipment 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

6. The external equipment receives the data from the PLC and the last 2 data are 16#0D and 16#0A.

   External equipment → PLC 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A

   (ASCII code: 0106106412343FCRLF)

7. D200=17 (number of the data received), and the content of the received data: D201=16#303A, D202=16#3031, D203=16#3136, D204=16#3630, D205=16#3134, D206=16#3332, D207=16#3334, D208=16#0D46, D209=16#000A.

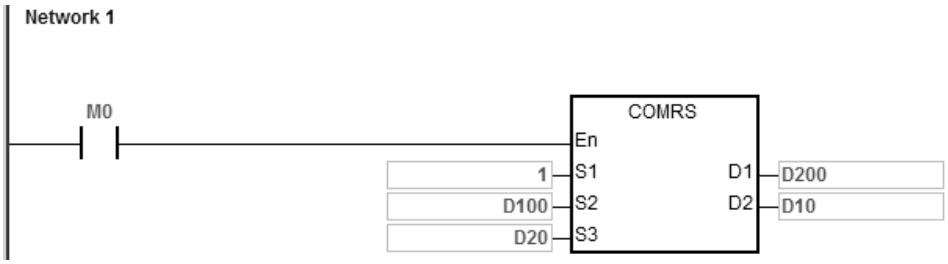
8. SM100=ON: reception of data is complete.

9. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When sending of data is complete, the receiving flag SM98 is ON and then the PLC starts receiving data until it receives the ending character (16#0D0A). When the reception of data is complete, SM100 is ON. If the communication timeout occurs but the ending character (16#0D0A) is still not received, the communication timeout flag SM104 is ON.

Example 5

The mode in **D_2** is 4 (the data received starts with a specific character and you set the timeout value to 5–255 ms.) and you set the mode of sending and receiving data to 8-bit mode (SM106=ON).



1. The length for the data to be sent: D20=4.

2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.

3. Set D10=16#0004 (mode: 4), D11=16#3A0F (the starting character is 16#3A and set the time value to 16#0F, meaning 15ms).

4. Enable the contact M0.

5. The PLC sends 4 pieces of data.

   Sending data: PLC→external equipment 31 32 33 34

6. The external equipment receives data from the PLC and then sends 7 consecutive words to the PLC with an interval of 1 ms between each sending.

   External equipment → PLC 30 3A 35 36 37 38 39

7. D200=6 (number of the data received), and the content of the received data: D201=16#003A, D202=16#0035, D203=16#0036, D204=16#0037, D205=16#0038, D206=16#0039.

8. SM100=ON: reception of data is complete.

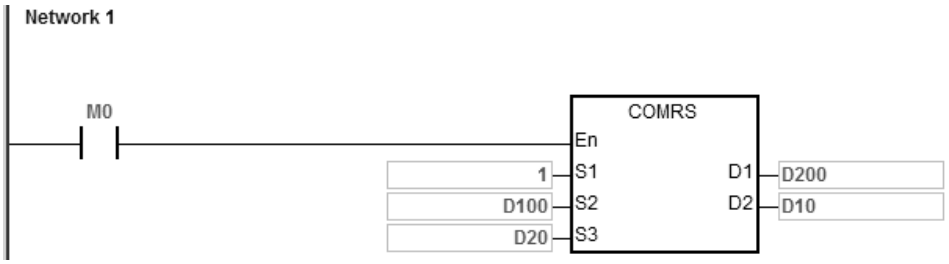
9. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the sending of data is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When the PLC receives the starting character 16#3A, it starts receiving data. The interval timeout between receiving each piece of data is set in D11. When the interval time exceeds the set timeout 16#0F (15 ms) and no data is coming in, SM100 is ON.

Example 6

The mode in $\mathbf{D_2}$ is 5 (the data received starts and ends with a specific character) and you set the mode of sending and receiving data to 16-bit mode (SM106=OFF).

The example uses a DVP-ES2 as the external equipment and reads data from D100 in the DVP-ES2.



1. The length for the data to be sent: D20=17.

2. The contents for the data to be sent: D100=16#303A, D101=16#3031, D102=16#3133, D103=16#3630, D104=16#3034, D105=16#3030, D106=16#3831, D107=16#0D37, D108=16#000A

3. Set D10=16#0005 (mode: 5), D11=16#3A0A (the starting character is 16#3A and the ending character is 16#0A).

4. Enable the contact M0.

5. The PLC sends 17 pieces of data.
   Sending data: PLC→external equipment 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A
   (ASCII code: 0106106412343FCRLF)

6. The external equipment receives data from the PLC and the last 2 data are 16#0D and 16#0A.
   External equipment → PLC 3A 30 31 30 36 31 30 36 34 31 32 33 34 33 46 0D 0A
   (ASCII code: 0106106412343FCRLF)

7. D200=15 (number of the data received), and the content of the received data: D201=16#303A, D202=16#3031, D203=16#3033, D204=16#3132, D205=16#3332, D206=16#4234, D207=16#0D34, D208=16#000A.

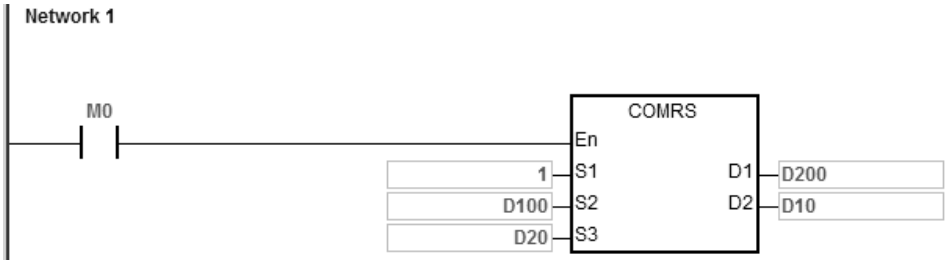
8. SM100=ON: reception of data is complete.

9. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When the PLC receives the starting character 16#3A, it starts receiving data until receiving the ending character 16#0A, and SM100 is ON. If the communication timeout occurs but the starting character 16#3A or the ending character 16#0A is still not received, the communication timeout flag SM104 is ON.

Example 7

The mode in $\mathbf{D_2}$ is 6 (the received data length) and you set the mode of sending and receiving data to 8-bit mode (SM106=ON).



1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
3. Set D10=16#0006 (mode: 6), D11=16#0008 (8 pieces of data to be received).
4. Enable the contact M0.
5. The PLC sends out 4 pieces of data.

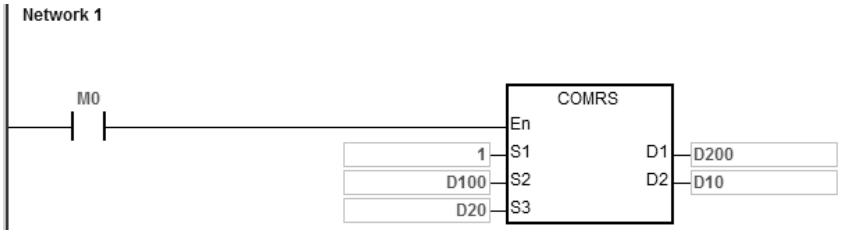
   Sending data: PLC→external equipment 31 32 33 34
6. The external equipment receives data from the PLC and then sends 8 consecutive data to the PLC.

   External equipment → PLC 32 33 34 35 36 37 38 39
7. D200=8 (number of the data received), and the content of the received data: D201=16#0032, D202=16#0033, D203=16#0034, D204=16#0035, D205=16#0036, D206=16#0037, D207=16#0038, D208=16#0039.
8. SM100=ON: reception of data is complete.
9. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When receiving a set quantity of data, the SM100 is ON. If the communication timeout occurs but the set quantity of data is still not received, the communication timeout flag SM104 is ON.

Example 8

The mode in D_2 is 7 (the data received ends with a specific character and generates communication interrupts) and you set the mode of sending and receiving data to 8-bit mode (SM106=ON).



Communication interrupt programs:

1. Clear the interrupt: D30=0.
2. The length for the data to be sent: D20=4.
3. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
4. Set D10=16#0007 (mode: 7), D11=16#000A (16#0A is the ending character).
5. Enable the contact M0.
6. The PLC sends out 4 pieces of data.

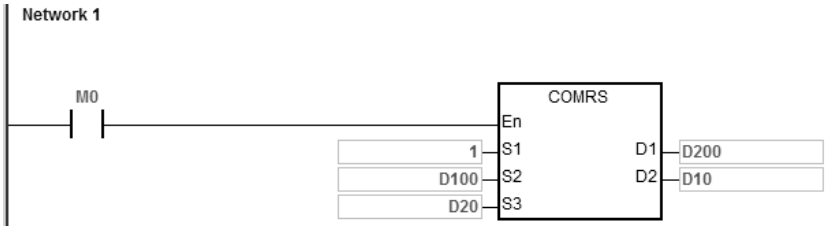
   Sending data: PLC→external equipment 31 32 33 34
7. D30=0 (the programs in the interrupt are not executed).
8. The external equipment sends data to the PLC.

   External equipment → PLC 31 32 33 34 35 0A
9. D200=6 (number of the data received), and the content of the received data: D201=16#0031, D202=16#0032, D203=16#0033, D201=16#0034, D202=16#0035, D203=16#000A.
10. SM100=ON: reception of data is complete.
11. D30=1 (the interrupt is triggered and then INC D30 is executed).
12. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When receiving the set ending character (16#06), SM100 is ON. If the communication timeout occurs but the set ending character is still not received, the communication timeout flag SM104 is ON.

Example 9

The mode in **D_2** is 8 (the set quantity of data is received and generates communication interrupts) and you set the mode of sending and receiving data to 8-bit mode (SM106=ON).



Communication interrupt programs:

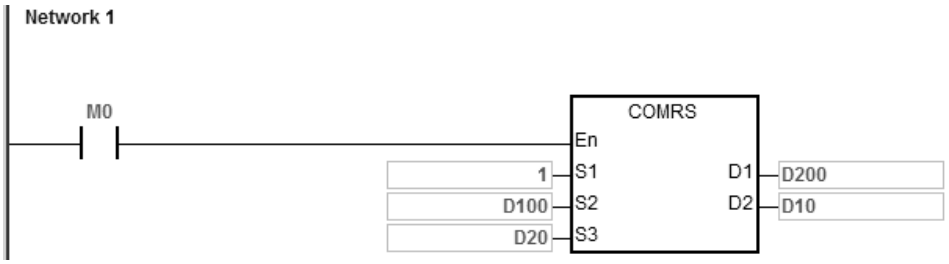
1. Clear the interrupt: D30=0
2. The length for the data to be sent: D20=4.
3. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
4. Set D10=16#0008 (mode: 8), D11=16#0008 (8 pieces of data to be received).
5. Enable the contact M0.
6. The PLC sends out 4 pieces of data.

   Sending data: PLC→external equipment 31 32 33 34
7. D30=0 (the programs in the interrupt are not executed).
8. The external equipment receives data from the PLC and then sends 8 consecutive data to the PLC.
9. External equipment → PLC 32 33 34 35 36 37 38 39
10. D200=8 (number of the data received), and the content of the received data: D201=16#0032, D202=16#0033, D203=16#0034, D204=16#0035, D205=16#0036, D206=16#0037, D207=16#0038, D208=16#0039.
11. SM100=ON: reception of data is complete.
12. D30=1 (the interrupt is triggered and then the INC D30 is executed).
13. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When receiving the set quantity of data, SM100 is ON. If the communication timeout occurs but the set quantity of data is still not received, the communication timeout flag SM104 is ON.

Example 10

The mode in **D_2** is 9 (the set ending character or the set quantity of data is received) and set the mode of sending data/mode of receiving data to 8-bit mode (SM106=ON).



1. The length for the data to be sent: D20=4.
2. The contents for the data to be sent: D100=16#0031, D101=16#0032, D102=16#0033, D103=16#0034.
3. Set D10=16#0009 (mode: 9), D11=16#0A0F (the ending character is 16#0A and the set data length is 16#0F).
4. Enable the contact M0.
5. The PLC sends out 4 pieces of data.
   Sending data: PLC→external equipment 31 32 33 34
6. The external equipment receives data from the PLC and then sends 15 pieces of data to the PLC.
   External equipment → PLC 31 32 33 34 35 0A 41 42 43 44 45 46 47 48 49
7. D200=6 (number of the data received), and the content of the received data: D201=16#0031, D202=16#0032, D203=16#0033, D204=16#0034, D205=16#0035, D206=16#000A.
   The PLC stops receiving data after the 6th piece of data is received.
8. SM100=ON: reception of data is complete.
9. To send more data, set the flag SM100 to OFF to start the operation again.

NOTE: When the data sending is complete, the receiving flag SM98 is ON and then the PLC is ready to receive data. When receiving the set ending character or the set quantity of data, the SM100 is ON. If the communication timeout occurs but the set ending character or the set quantity of data is still not received, the communication timeout flag SM104 is ON.

Additional remarks

1. There is no limit on the number of times you can execute the COMRS communication instruction. However, each communication port can only be enabled by one communication instruction, and the later communication instructions that follow are not executed.
2. The instruction does not use checksum when you execute this instruction. If you need a checksum, use COMRS and another available instruction.
3. If the value in **D_2** is 2, 3, 5, 6 or 9, it is suggested that you set a timeout period. After you set a timeout period, the instruction tries to send the data again if a stop character is not received.
4. The instruction does not automatically clear the value in **D_1**–**D_1**+n whenever the instruction is executed or the PLC begins to receive new communication data. You can know whether and how much data the PLC receives only after a completion flag switches from OFF to ON . If the you want to clear the values in **D_1**–**D_1**+n, use the ZRST instruction (API 1206).
5. If the value in **S_1** is out of range, the instruction is not executed.
6. If the number of devices starting from **S_2** is not equal to the value in **S_3**, the instruction is executed, SM0 is ON, and the error code in SR0 is 16#2003.
7. If the value in **D_2** is not between 0–9, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
8. If the value in **D_2** is 6, 8 or 9, and the number of devices starting from **D_1** is not equal to the value in **D_2**+1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
9. If the quantity of data received is greater than the number of devices starting from **D_1**, the data that cannot be stored is ignored.
10. If a completion flag is ON, the PLC stops receiving data. If a communication port receives data when a completion flag is ON, the data is not received.
11. If the setting value in **S_3** is not between 0–256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
12. When the mode of **D_2** is 6 or 8, the length of **D_2**+1 not between 1–256, the instruction is not executed. SM0 is ON, and the error code in SR0 is 16#200B.

API	Instruction code			Operand	Function
1813		COMDF	P	**S_1, S_2, S_3, S_4, S_5, D**	Setting the communication format for a serial communication port

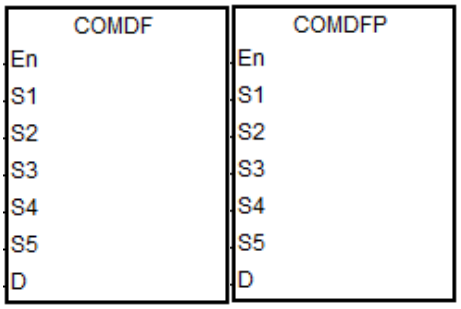
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1													○	○		
S_2													○	○		
S_3													○	○		
S_4													○	○		
S_5													○	○		
D													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2													
S_3													
S_4													
S_5													
D													

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

6

Symbol



S_1 : Baud Rate (Unit:100 bps)

S_2 : Number of data bits

S_3 : Parity bit

S_4 : Number of end bits

S_5 : Modbus format selection

D : Communication port number

Explanation

1. This instruction provides a way to directly set the parameter values, instead of declaring variables.

2. **S_1** sets the baud rate with the units 100 bps. For example, a value 96 indicates 9600 bps.

3. **S_2** sets the number of data bits. The value 7 indicates 7 data bits and 8 indicates 8 data bits. If the value is not 7 or 8 in **S_2**, the instruction uses the default value.

4. **S_3** sets the parity bit. The value 0 indicates None (no parity bit). The value 1 indicates Odd bit checking. The value 2 corresponds to Even bit checking. If the value in **S3** is not 0, 1 or 2, the instruction uses the default value.

5. **S_4** sets the number of end bits. The value 1 (preset) indicates 1 bit. The value 2 indicates 2 bits. If the value in **S_4** is not 1 or 2, the instruction uses the default value.

6. **S_5** sets the communication mode for Modbus communication. The value 0 indicates ASCII (default value). The value of 1 indicates RTU. If the value in **S_5** is not 0 or 1, the instruction uses the default value.

7. **D** sets communication port number. The number for COM1 is 1 and COM2 is 2. If the setting value is out of the valid range, the instruction does not set the communication port format.

8. You can also directly set the communication port in HWCONFIG in ISPSoft (COM Port settings) or with the special registers. For more on in HWCONFIG, see the ISPSoft user manual. Refer to Section 6.19.3 for setting the communication-related SR and SM registers.

9. The communication at the actual communication port changes immediately after you change the setting of the instruction. If some communication is being carried out at the moment, it is forced to cancel. Additionally, the corresponding setting value in SM/SR changes accordingly. For details on SM/SR, refer to Section 6.19.3.

10. This instruction does not change any setting for the actual communication port when the communication format setting is the same as the previous setting.

Example

1. This example uses the PLC COM1 port. Other PLC communication ports are similar.

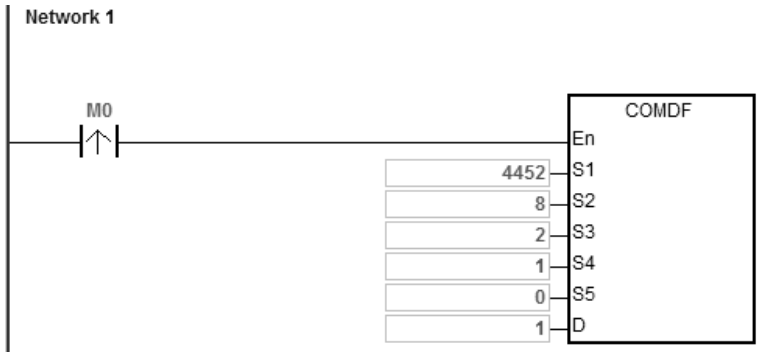
2. The contact for the start condition is M0.

3. Set the (RS485) communication format of PLC COM1 to 115200, 8, E and 1.

4. Set the (RS485) communication mode of PLC COM1 to ASCII .

5. The following table explains the COMDF operands for the example.

Operand	Description		Content value
S_1	Baud Rate	115200 bps	1152
S_2	Number of data bits	8	8
S_3	Parity bit	E	2
S_4	Number of end bits	1	1
S_5	Mobdus format selection	ASCII	0
D	Communication port number	PLC COM1	1


Network 1
M0
COMDF
En
4452
S1
8
S2
2
S3
1
S4
0
S5
1
D

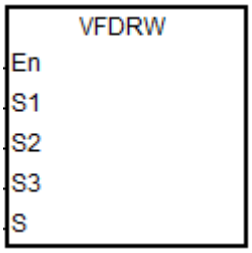
API	Instruction code	Operand	Function
1814	VFDRW	**S_1, S_2, S_3, S**	Serial communication instruction exclusive for Delta AC motor drive

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●		○	○	○	○		
S_2								●	●		○	○	○	○		
S_3								●	●		○	○	○	○		
S								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Communication port number

S_2 : VFD station address

S_3 : Function code

S : Source and received data

Explanation

1. **S_1** sets the communication port number. The number for COM1 is 1 and COM2 is 2. If the value exceeds the valid range, the instruction does not receive any communication data.

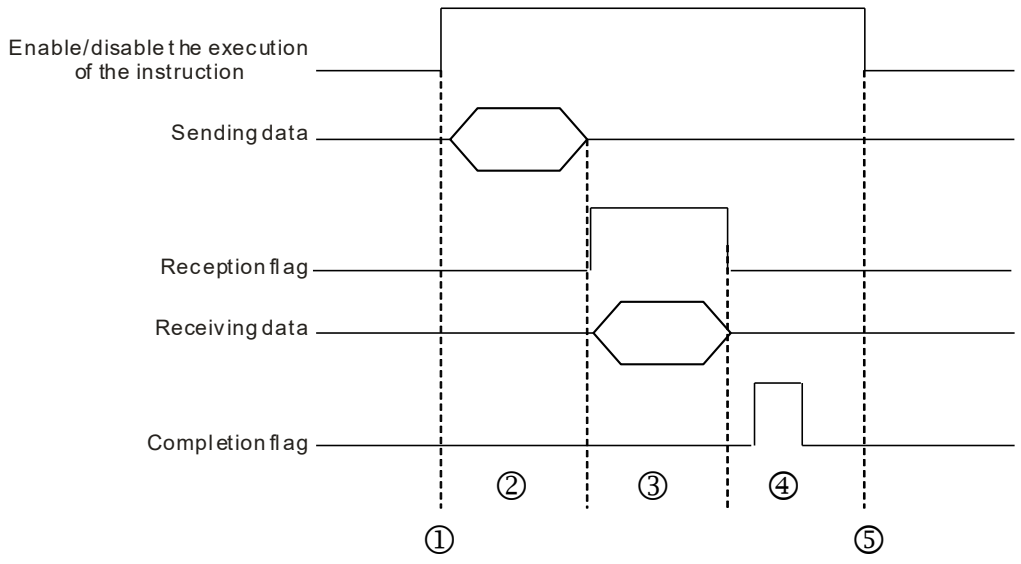
2. **S_2** sets the station address for the VFD AC motor drive. 0 indicates that the instruction uses the broadcast mode. The range is between 0–254, and the instruction is not executed if this value is out of the valid range.

3. **S_3** is the communication function code, and **S** is the source or received data as explained in the following table.

S_3 function code	**S_3** function name	**S** source and received data	Remark
0	Reset due to abnormality	Unused	Any value can be stored in **S**.

S_3 function code	**S_3** function name	**S** source and received data	Remark
1	Clockwise running command	Velocity value	Refer to AC motor drive user manual for the setting value and the unit.
2	Counterclockwise running command	Velocity value	
3	Stop	Unused	Any value can be stored in **S**.
4	Jog clockwise running command	Unused	Refer to AC motor drive user manual for setting the jog velocity.
5	Jog counterclockwise running command	Unused	
6	Reading the state	Received state values	Refer to AC motor drive user manual for the meaning of the state values of the 5 bit addresses H2100–H2104 for VFD.

4. The following chart shows the timing for sending and receiving data.



Description:

① → Start or stop the execution of the instruction.

② → Transmitting data begins. During this time, the communication timeout time is not measured.

③ → The reception flag is set. From the moment when the first character is received to the moment when the next

character is received, the period of time is measured. Whenever a character is received, the measured time is cleared to 0. The instruction generates the communication timeout flag if the time measured is greater than the communication timeout setting value.

④ → When receiving of data is complete, the instruction sets the completion flag. You must clear the flag before receiving more data.

⑤ → The instruction is stopped for one cycle after the completion flag is set. Then the instruction can be started in the next cycle.

5. There is no limit to the number of times the instruction can be executed. The instruction can use only one communication port for the output and execution of one communication instruction each time. If receiving and sending data is complete, you must disable the instruction to correctly release the communication control.

6. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example of setting the communication protocol

1. Set the PLC COM1 (RS485) port with station address 2 and the communication format in HWCONFIG with these values: ASCII , 115200, 7, N, 2.

2. Set the motor drive parameters using the panel on the Delta C2000 AC motor drive according to the following steps.

   A. Set 09-00 to 1: the station address of the AC motor drive is set to 1.

   B. Set 09-01 to 115.2: RS485 baud rate of the AC motor drive is 115200.

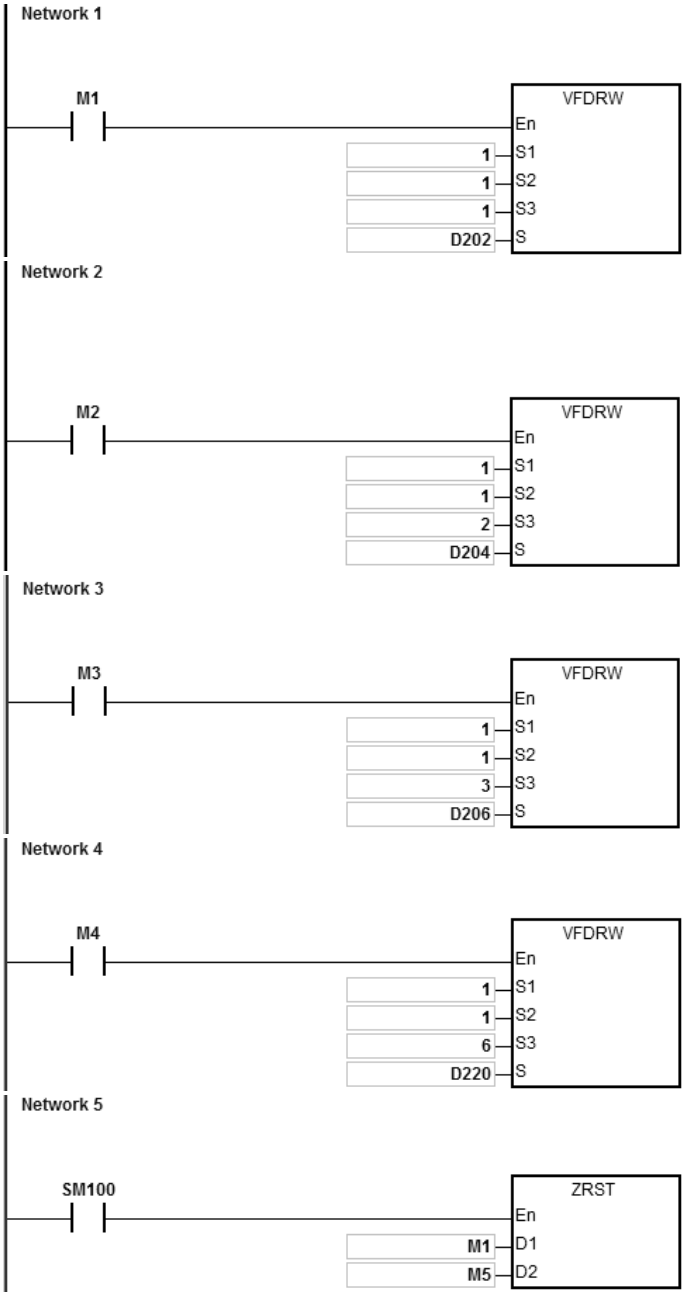
   C. Set 09-04 to 1: RS485 communication format of the AC motor drive is 7, N, 2.

   D. Set 09-20 to 1: the frequency instruction is input through RS485.

   E. Set 09-21 to 2: the running instruction is input through RS485.

Example

Use the VFDRW instruction to control the velocity: make the VFD run forward at the frequency of 120Hz, then run in reverse at the frequency of 180Hz, and then stop running.



1. Connect ES3 COU to VFD.

   Set D202=12000 initially. When M1 is ON, VFD starts to accelerate after receiving the clockwise running command, and runs clockwise at 120Hz.

2. Set D204=18000 initially. When M2 is ON, VFD starts to decelerate until it stops after receiving the counterclockwise running command, and runs counterclockwise at 180Hz.

3. When M3 is ON (at this time, the value in D206 is ignored), VFD decelerates to stop after receiving the stop command.

4. When M4 is ON, the instruction reads the values of H2100–H2104 of VFD and stores them in D220–224.

Device	D220	D221	D222	D223	D224
Content	Error code	VFD state	Frequency command	Output frequency	Output current

The state of the VFD:

Bit2=1 causes the VFD to execute the Jog command. Bit4–3=11B causes the VFD to run counterclockwise. The frequency command is 18000, and causes the VFD to run at 180Hz. For the definitions of the parameter addresses in the communication protocol, refer to the Delta AC Motor Drive user manual.

5. The reception completion flag SM100 is ON, and the values of M1–M5 are cleared to avoid interfering with the next communication command.

After receiving the data that the VFD sends back, the instruction checks the format of the data sent back from VFD. If the data format is correct, SM100 is ON; otherwise, SM102 is ON.

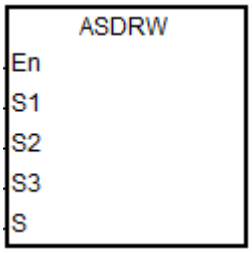
API	Instruction code			Operand	Function
1815		ASDRW		S_1, S_2, S_3, S	Serial communication instruction exclusive for Delta servo drive

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●		○	○	○	○		
S_2								●	●		○	○	○	○		
S_3								●	●		○	○	○	○		
S								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Communication port number

S_2 : Station address of the servo

S_3 : Function code

S : Source and received data

Explanation

1. S_1 sets the communication port number. The number for COM1 is 1 and COM2 is 2. If the value exceeds the valid range, the instruction does not receive any communication data.

2. S_2 sets the station address for the servo. 0 indicates that the instruction uses the broadcast mode. The range of the value is between 0–254. The instruction is not executed if the value is out of the valid range.

3. Refer to Delta Servo Operation manual for details on servo parameters.

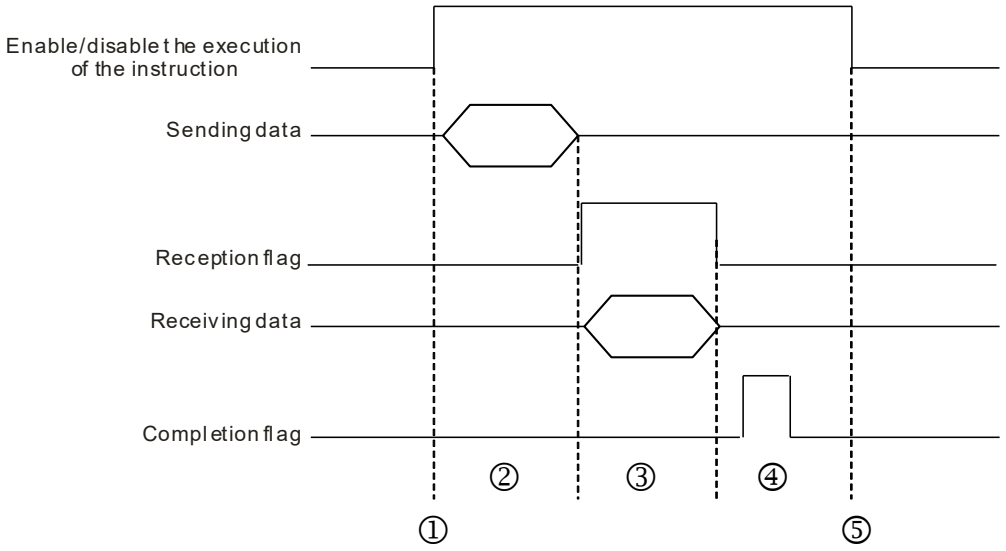
4. S_3 is the communication function code, and **S** is the source or received data as explained in the following table.

S_3 and **S** operands for the A, AB, A+, B Series			
S_3 function code	**S_3** function name	**S** source or received data	Remark
0	Reading the servo state value	Occupies 5 consecutive devices **S**–**S**+4	Reading the state value from P0-04–P0-08
1	Reading the servo register value	Occupies 8 consecutive devices **S**–**S**+7	Reading the value from the registers P0-09–P0-16
2	Writing the servo register value	Occupies 8 consecutive devices **S**–**S**+7	Writing the data in the registers P0-09–P0-16
3	Jog velocity input, clockwise running, counterclockwise running and stop	The range of velocity: 1–3000; 4999 (run clockwise) 4998 (run counterclockwise) 5000 (stop)	Writing the data into the registers P4-05
4	Servo ON/OFF	1: Servo On Any other value: Servo OFF	Writing the data into the registers P2-30
5	Velocity command	Valid range: -5000 to 5000	Writing the data into the registers P1-09–P1-11
6	Torque command	Valid range: -300 to 300	Writing the data into the registers P1-12–P1-14

S_3 and **S** operands for the A2, B2 Series			
S_3 function code	**S_3** function name	**S** source or received data	Remark
16	Reading the servo state value	Occupies 10 consecutive devices **S**–**S**+9	Reading the state value from the registers P0-09–P0-13 (32-bit value)
17	Writing data into the servo register	Occupies 8 consecutive devices **S**–**S**+7	Writing the data into the registers P0-17–P0-20 (32-bit value)
18	Writing the mapping parameter value [*1]	Occupies 8 consecutive devices **S**–**S**+7	Writing the data into the registers P0-25–P0-28 (32-bit value)
19	Jog velocity input, run clockwise, run counterclockwise,	Valid range of velocity: Valid range 1–5000;	Writing data into the registers P4-05

	stop	4999 (run clockwise) 4998 (run counterclockwise) 0 (stop)	
20	Servo ON/OFF	1: Servo On Any other value: Servo OFF	Writing the data into the registers P2-30
21	Velocity command (3 sets)	Occupies 6 consecutive devices with the valid setting value range: -60000 to 60000	Writing data into the registers P1-09–P1-11 (32-bit value)
22	Torque command (3 sets)	Occupies 6 consecutive devices with the valid setting value range: -300 to 300	Writing the data into the registers P1-12–P1-14 (32-bit value)
23	Setting the mapping targets of servo parameters	Occupies 8 consecutive devices **S**–**S**+7	Writing the data into the registers P0-35–P0-38 (32-bit value)
*1: B2 series does not support function code 18 and 23.			

5. The following chart shows the timing for sending and receiving data.



Description:

①→ Start or stop the execution of the instruction.

②→Transmitting data begins. During this period, the communication timeout time is not measured.

③→ The reception flag is set. From the moment when the first character is received to the moment when the next character is received, the period of time is measured. Whenever a character is received, the measured is cleared to 0. The instruction generates the communication timeout flag if the time measured is greater than

the communication timeout setting value.

④ → When receiving of data is complete, the instruction sets the completion flag. You must clear the flag before receiving more data.

⑤ → The instruction is disabled for one cycle after the completion flag is set. Then the instruction can be started in the next cycle.

6. There is no limit to the number of times the instruction can be executed. The instruction can use only one communication port for the output and execution of one communication instruction each time. If receiving and sending data is complete, you must disable the instruction to correctly release the communication control.

7. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example of setting the communication protocol:

1. Set the PLC COM1 (RS485) port in HWCONFIG with these values: ASCII , 115200, 8, E, 1

2. Set the servo parameters using the panel on the Delta ASDA-A2 servo according to the following steps.

   A. Set P2-08 to 10 to restore the factory settings.

   B. Cycle the power on the servo off and then back on.

   C. Set P1-01 to 0001 (PR mode).

   D. Set P3-00 to 2: the station address of the servo is set to 2.

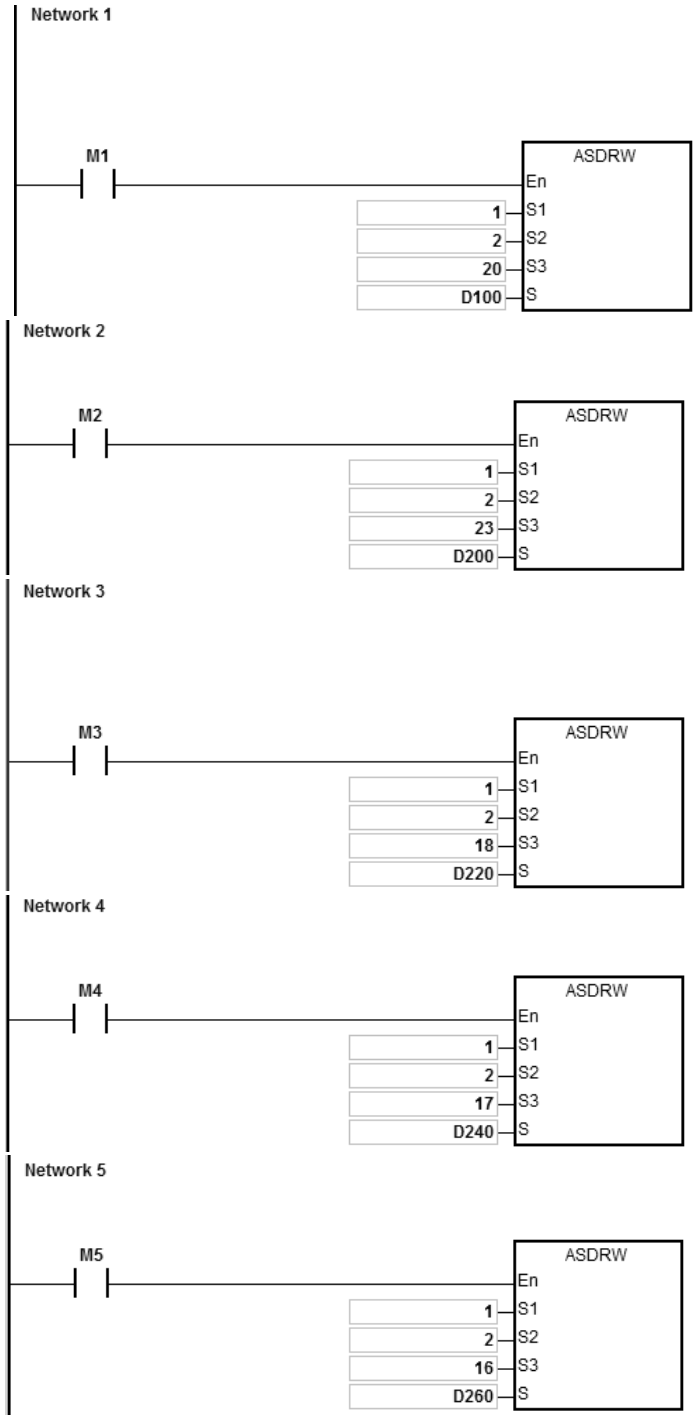
   E. Set P3-01 to 0205: the RS485 baud rate of the servo is set to 115200.

   F. Set P3-02 to 0004: the RS485 communication format of the servo is set to 8, E, 1.

   G. Cycle the power to the servo again after completing these settings.

Example

Use the ASDRW instruction to control the velocity: make the servo run to the relative position 5000000 PUU by accelerating for 400 ms to the speed 3000.0 r/min and then decelerating for 200 ms.



1. Connect the ES3 CPU to ASDA-A2.

   Set D100=1 initially. When M1 is ON, ASDA-A2 is Servo ON.

2. Set the values in D200–D207 as in the following table, which are written to P0-35–P0-38 in the ASDA-A2. The values occupies 8 consecutive devices.

Device	D200	D201	D202	D203	D204	D205	D206	D207
Setting value (16#)	05140515		06020602		06030603		053C0507	

When M2 is ON, the values in D200–D207 are written to P0-35–P0-38 in the ASDA-A2.

The setting values of P0-35–P0-38 are set the mapping target for P0-25–P0-28. You can set the mapping target; refer to the Delta servo operation manual.

ASDA-A2	P0-35	P0-36	P0-37	P0-38
Setting value (16#)	0514 0515	0602 0602	0603 0603	053C 0507

Setting the value of P0-38 to 053C 0507indicates that the mapping parameter target for P0-28 is P5-60 (16 bits) and P5-07 (16 bits).

ASDA-A2	P0-25	P0-26	P0-27	P0-28
Mapping parameter target	P5-20, P5-21	P6-02	P6-03	P5-60, P5-07
Parameter name	Acceleration/deceleration time1 Acceleration/deceleration time2	Path type1	Path1 data	Target velocity and PR command trigger

3. The following table shows the setting values in D220–D227.

Device	D220	D221	D222	D223	D224	D225	D226	D227
Setting value (16#)	0190 00C8		0000 1083		4C4B40		7530 0001	

When M3 is ON, the values in D220-D227 are written into P0-25–28 in the ASDA-A2.

ASDA-A2	P0-25	P0-26	P0-27	P0-28
Setting value (16#)	0190 00C8	0000 1083	4C4B40	7530 0001

The servo starts running with the parameters: acceleration time=0190 (400 ms), deceleration time=00C8 (200 ms), path type=1083, position command=4C4B40 (5000000 PUU), target velocity=7530 (3000.0 rpm) and PR command trigger =1.

4. Set the values in D240-D247 as shown in the following table, to be written into P0-17–P0-20 in the ASDA-A2. The values occupy 8 consecutive devices.

Device	D240	D241	D242	D243	D244	D245	D246	D247
Setting value (10#)	41		0		0		0	

When M4 is ON, the values in D240–D247 are written into P0-17–20 in the ASDA-A2.

The setting values of P0-17–20 set the contents of P0-09–12.

You can set the contents to be displayed; refer to the Delta Servo Operation manual.

ASDA-A2	P0-17	P0-18	P0-19	P0-20
Setting value (10#)	41	0	0	0

Setting P0-17 to 41 indicates that the contents of P0-09 is the drive state.

ASDA-A2	P0-09	P0-10	P0-11	P0-12
Content	Drive state	Number of motor feedback pulses	Number of motor feedback pulses	Number of motor feedback pulses

5. When M5 is set to ON, the values of P0-17- P0-20 of ASDA-A2 are read to D260-D267.

Device	D260	D262	D264	D266
Content	Drive state	Number of motor feedback pulses	Number of motor feedback pulses	Number of motor feedback pulses

When the drive state bit (Bit 4) is one, it indicates that the target position is reached.

Refer to Delta servo operation manual for the explanation for P0-46.

6. When the reception completion flag SM100 is set to ON, the instruction clears the values of M1–M5 to avoid interfering with the next communication command.

After receiving the data that the ASDA-A2 sends back, the PLC checks the format of the data sent back from the ASDA-A2. If the data format is correct, SM100 is ON; otherwise, SM102 is ON if the data format is incorrect.

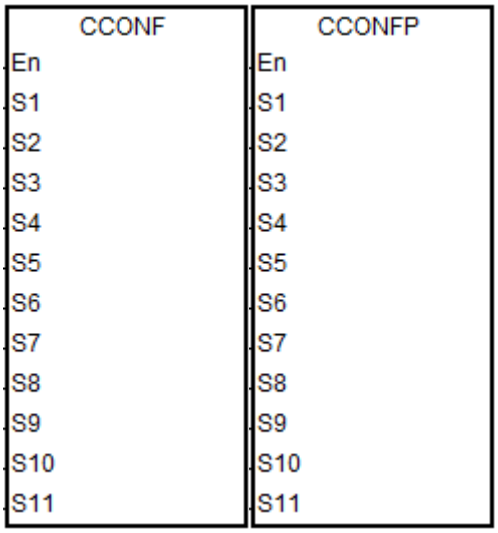
API	Instruction code			Operand	Function
1816		CCONF	P	S_1-S_{11}	Setting the parameters in the data exchange table for a communication port

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7			●					●								
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		
S_{11}			●					●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1-S_{11}	Refer to the explanation for the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1-S_{11} : Refer to the explanation for the instruction.

Explanation

1. The following table lists the names and descriptions of **S_1–S_{11}**.

Device	Name	Description	Data type	Remark
S_1	Communication port number	1=COM1, 2=COM2. Any other value, the instruction does not modify the operand.	WORD	
S_2	Item number in the data exchange table	Valid range: 1–32. If the value is out the range, the instruction does not modify the operand.	WORD	
S_3	The station address of the remote equipment	Valid range: 0 and 1–240. If the value is greater than 240, the instruction uses 240 as the value. If the value is less than or equal to 0, the original station address is not changed.	WORD	A negative number is processed as 0.
S_4	Function code for reading data	Supports 16#01, 02, 03, 04 and 17. If the value exceeds the valid range, the instruction does not modify the operand. If **S_4**=16#17, it is processed as 16#17.	WORD	
S_5	Reading the remote communication address	16#0000–16#FFFF	WORD	
S_6	Reading data length	Bit function code, 16#01, 02: Supports 1–256 bits; Word function code, 16#03, 04, 17: Supports 1–100 words. If the value is 0 or smaller than 0, it indicates do not read data. If the value is greater than the maximum, the instruction uses the maximum value.	WORD	
S_7	The local register for storing the data received	Bit function code: only M device can be selected. Word function code: only D device can be selected.	BOOL WORD	
S_8	Function code for writing data	Supports 16#05, 06, 0F, 10. If the value is out of the range, the	WORD	

Device	Name	Description	Data type	Remark
		instruction does not modify the operand.		
S_9	Writing the remote communication address	16#0000–16#FFFF	WORD	
S_{10}	Writing data length	Bit function code, 16#0F: Supports 1–256 bits; Word function code, 16#10: Supports 1–100 words. For function code 16#06, the value can only be 1. If the value is 0 or smaller than 0, it indicates do not read data. If the value is greater than the maximum, the instruction uses the maximum value.	WORD	
S_{11}	The local register into which the remote data are written	Bit function code: only M can be selected. Word function code: only D can be selected.	BOOL WORD	

2. It is suggested that you use the pulse instruction.

3. See the details about the Modbus function codes in S_4 and S_8 below.

   The command for the ES3 reads the data from several bit devices (which are not discrete input devices) is 1 (16#01).

   The command for the ES3 reads the data from several bit devices (which are only discrete input devices) is 2 (16#02).

   The command for the ES3 to read the data from several word devices (which are not input registers) is 3 (16#03).

   The command for the ES3 to read the data from several word devices (which are only input registers) is 4 (16#04).

   The command for the ES3 to write the state into a bit device is 5 (16#05).

   The command for the ES3 to write the state into a word device is 6 (16#06).

   The command for the ES3 to write the state into several bit devices is 15 (16#0F).

   The command for the ES3 to write the data into several word devices is 16 (16#10).

   The command for the ES3 to synchronously read from and to write the data into several word devices is 23 (16#17).

Only the function codes listed above are supported. Other function code setting values, such as 0, are invalid (including communication address, length and the start register), and the instruction executes the data exchange function based on the original communication parameter settings.

4. When you select 16#17 (for reading and writing synchronously) in **S_4** (the function code for reading), the operand **S_8** (the function code for writing) is processed as invalid and 16#17 is automatically processed for writing data.

5. The parameter values specified by the instruction are valid only while PLC is running. If you cycle the power on the PLC, the data in the data exchange table set in HWCONFIG are taken as default values. If you want to modify the values for parameters, use this instruction.

6. You can use the instruction to immediately set the parameters for the communication connection when the data exchange function is not running. The new communication parameters are not used until the next cycle if the data exchange function is running; however, the instruction changes the communication parameters for the connection number.

   For example, while the data exchange function is running with the parameters set for connection 3, you can specify new the parameters for connection 3. The new parameters are not used until the next cycle when connection 3 starts.

7. The instruction only allows you to change the communication parameters. Refer to the following table for the flags to start and close the communication connection function if you use the PLC program to perform those functions.

   When you set the automatic scan function through ISPSoft, the start/stop flag of the connection number automatically updates the start/stop state once after the data exchange function finishes executing the scan.

   The following table describes the SM flags.

SM No.	Attribute	Explanation for COM1 data exchange parameters
SM750	R/W	Enables data exchange
SM752–SM783	R/W	Enables data exchange connections 1–32
SM784–SM815	R	Reading success for data exchange connections 1–32
SM816–SM847	R	Error flags for data exchange connections 1–32
SM No.	Attribute	Explanation for COM2 data exchange parameters
SM862	R/W	Enables data exchange
SM864–SM895	R/W	Enables data exchange connections 1–32
SM896–SM927	R	Reading success for data exchange connections 1–32
SM928–SM959	R	Error flags for data exchange connections 1–32

The PLC sets the flag to ON when the reading success flag indicates that receiving data is complete and the data checked are correct. If an error occurs in receiving data or a communication timeout occurs, the error flag is set to ON refer to the error codes). The PLC does not reset any of the reading success or error flags to OFF in the data exchange since the reading success flag and error flag of every connection number are not ON simultaneously.

8. The following table describes the SR (only-read registers) in the data exchange function.

SR No.	Description
SR1335	Cycle of the actual connections 1–32 in the COM1 data exchange
SR1336	Cyclic connection number in the current COM1 data exchange
SR1340–SR1371	Error codes of the connections 1–32 in the COM1 data exchange
SR1375	Cycle of the actual connections 1–32 in the COM2 data exchange
SR1376	Cyclic connection number in the current COM2 data exchange
SR1380–SR1411	Error codes for connections 1–32 in the COM2 data exchange

9. The data exchange function does not provide a writing success flag. It is suggested that you refer to the connection number in the execution to judge whether the writing of data succeeds or not.

   For example, when the executed connection number is 3 in SR1336, the successive communication actions are to read the communication data first, and then to write the communication data after reading is completed. The connection number is incremented to 4 after writing is completed.

10. If the reading data length value in **S_6** and the initial device in **S_7** are out of the range of the D or M device, the length value in **S_6** is automatically set to be in the valid range. For example, if the length value in **S_6** is100 and the initial device of **S_7** is M8182, the value in **S_6** is set to 10 automatically.

11. In the following cases, the instruction is not executed and the parameter settings in HWCONFIG are not changed for the communication port, and SM0 is set to ON, and the error code in SR0 is 16#200B.

    - An input error occurs when the setting values in **S_1, S_2, S_4** and **S_8** are out of the specified range.
    - When **S_4** or **S_8** function code selects the bit type for reading or writing data, the local device for storing data **S_7** or **S_{11}** must select the M device. An input error occurs if the selection is not M.
    - When the **S_4** or **S_8** function code selects the word type for reading or writing data, the local device for storing data **S_7** or **S_{11}** must select the D device. An input error occurs if the selection is not D.

Example: ES3 COM1 (RS485)

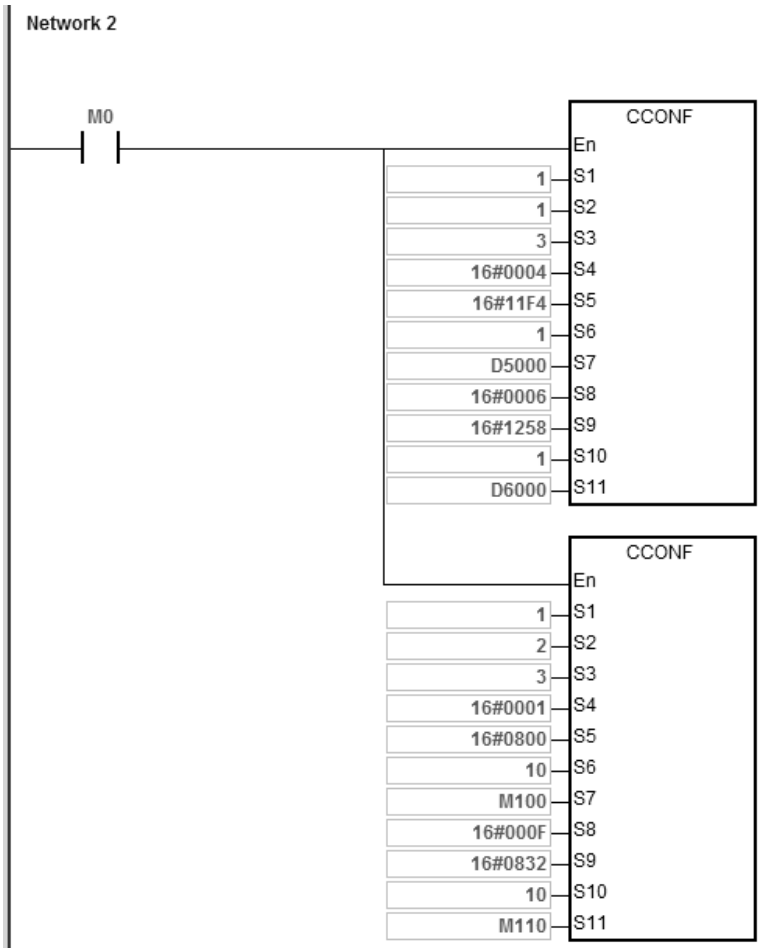
1. For the data exchange between the ES3 Series CPU and the DVP-ES2 CPU, the following table shows the COM1 data exchange table in HWConfig in ISPSoft.

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
1	Specified connection	2	D50	←	D50	50	H03	Program control
			D100	→	D100	50	H10	
2	Specified connection	2	D150	←	D150	50	H03	Program control
			D200	→	D200	50	H10	

2. Before starting data exchange, suppose that the corresponding data between the ES3 Series CPU and the DVP-ES2 CPU are listed in the following table.

ES3 PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D50-D99	0	D50-D99	1-50
D100-D149	100-149	D100-D149	0
D150-D199	0	D150-D199	150-199
D200-D249	200-249	D200-D249	0

3. The data exchange between the ES3 Series CPU and the DVP-ES2 CPU starts when X0 is ON.



4. After data exchange starts, the corresponding data between the ES3 Series CPU and the DVP-ES2 CPU change as shown in the following table.

ES3 PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D50-D99	1-50	D50-D99	1-50
D100-D149	100-149	D100-D149	100-149
D150-D199	150-199	D150-D199	150-199
D200-D249	200-249	D200-D249	200-249

5. When M0 is ON, the COM1 data exchange table parameters in the ES3 Series CPU are modified as shown in the following table.

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
1	Specified connection	3	D5000	←	D500	1	H04	Program control
			D6000	→	D600	1	H06	
2	Specified	3	M100	←	M0	10	H01	Program

Item No.	Detection method	Remote station address	Local device		Remote device	Length	Function code	How to start
	connection		M110	➔	M50	10	H0F	control

6. Due to the change of COM1 data exchange table parameters, the corresponding data between the ES3 Series CPU and the DVP-ES2 CPU are modified as shown in the following table.

ES3 PLC (Master)	Content value	ES2 PLC (Slave)	Content value
D5000	3000	D500	3000
D6000	4000	D600	4000
M100-M109	ON	M0-M9	ON
M110-M119	OFF	M50-M59	OFF

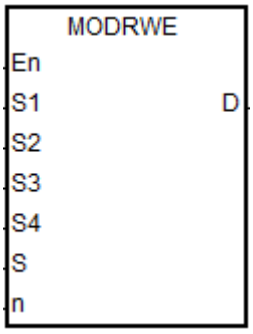
API	Instruction code	Operand	Function
1817	MODRWE	**S_1, S_2, S_3, S_4, S, n, D**	Reading and writing Modbus data without using any flags

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S								●								
n								●	●				○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S	●	●			●	●							
n		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Communication port number

S_2 : Unit address

S_3 : Function code

S_4 : Device address

S : Register for reading and writing the data

n : Data length

D : Flag for completion of the reading and writing data

Explanation

1. **S_1** sets the serial number of a communication port. The number of COM1 is 1 and COM2 is 2. If the value exceeds the valid range, the instruction does not receive any communication data.

2. Refer to the explanation of the MODRW instruction (API 1808) for the meaning of the operands **S_2**, **S_3**, **S_4**, **S** and **n**.

3. **D** sets the communication state flags when the instruction completes the communication, and the flags occupy 3 consecutive devices. You must reset the flags to OFF. The following table gives explanations of the flag states.

Operand	Description
D	Receiving data successful.
D +1	Error occurs in receiving data.
D +2	Reception timeout flag

NOTE: Only one flag is set to ON among the three state flags, and the corresponding special flags (SM) are also set to ON every time the communication is completed. Refer to Section 6.19.3 for the use of the special flags.

4. The timing for sending the instruction begins when the instruction starts. You must disable the instruction for a scan cycle after communication is completed. The next communication instruction can be sent as soon as the instruction is restarted.

5. The communication action and control sequence of the instruction are similar to MODRW instruction (API 1808). The only difference between the two instructions is that you can send the communication command without control over the flag for sending data.

6. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example

You can compare the MODRW program and MODRWE program, for example, by using the PLC COM1 and function code 03 to read eight pieces of data from D20 in the DVP-ES2. The same is true for other PLC communication ports. Refer to the MODRW instruction (API 1808) and the following example for the use of other function codes.

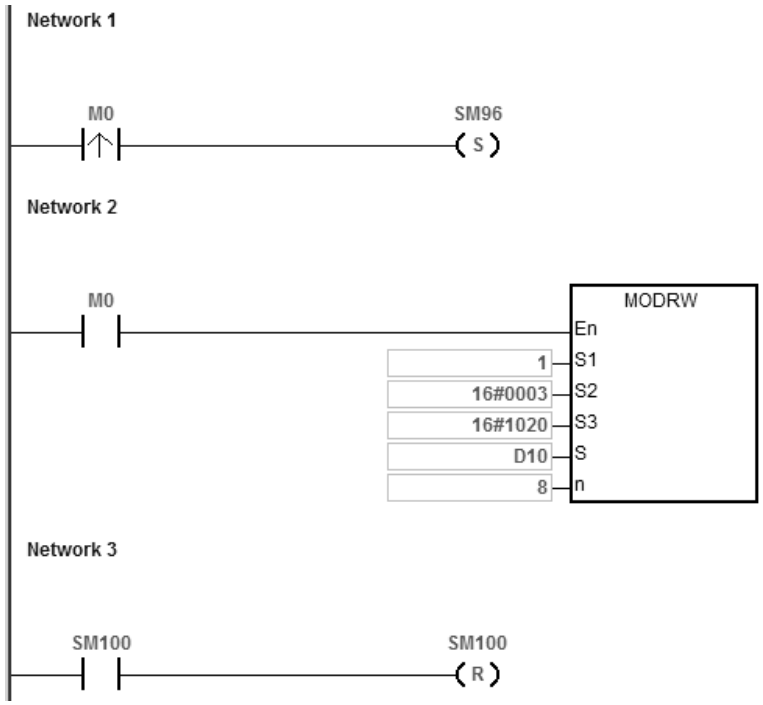
1. The device address of D20 in the DVP-ES2 CPU (16#1020) and the content values in D20–D27 are shown in the following table.

Device	D20	D21	D22	D23	D24	D25	D26	D27
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

2. The ES3 Series PLC reads the content values from D20–D27 in the DVP-ES2 CPU through communication.

Method 1: Using MODRW instruction

The data in D20-D27 of the DVP-ES2 are read when SM96 is ON and M0 is ON.



The following table explains the MODRW operands.

Operand	Description	Device
S_1	Unit address	16#0001
S_2	Function code	16#0003
S_3	Reading the device address of D20	16#1020
S	First register for storing the data read	D10
n	Reading the data length	8

The communication response between the ES3 Series CPU and the DVP-ES2 depends on the mode.

ASCII mode: You do not need to convert the ASCII codes, and they are expressed in 16# values.

- ES3 sends the communication command: “01 03 10 20 00 08 C4 CR LF”
- ES3 receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”

RTU mode

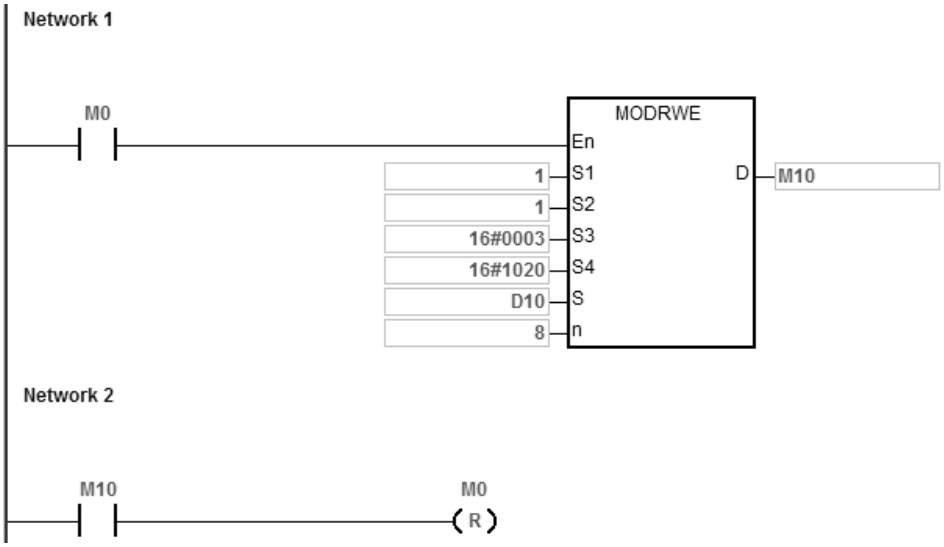
- ES3 sends the communication command: “ 01 03 10 20 00 08 41 06”
- ES3 receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”

SM100 is ON if there is no error in the data.

After the receiving of the data sent from the DVP-ES2, the PLC confirms the data format sent back from the DVP-ES2. If there are no errors in the format, SM100 is ON; otherwise, SM102 is ON. SM104 is ON if there is no response.

Method 2: Using MODRWE instruction

The data in D20-D27 of the DVP-ES2 are read when M0 is ON.



The following table explains the MODRWE operands.

Operand	Description	Device
S1	Communication port number	16#0001
S2	Unit address	16#0001
S3	Function code	16#0003
S4	Reading the device address of D20	16#1020
S	First register for storing the data read	D10
n	Reading the data length	8
D	Flag for completion of reading and writing data	M0

The communication response between the ES3 series CPU and the DVP-ES2 depends on the mode.

ASCII mode: You do not need to convert the ASCII codes, and they are expressed in 16# values.

- ES3 sends the communication command: “01 03 10 20 00 08 C4 CR LF”
- ES3 receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC AA CR LF”

RTU mode

- ES3 sends the communication command: “01 03 10 20 00 08 41 06”
- ES3 receives the communication command: “01 03 10 12 34 56 78 11 22 33 44 55 66 77 88 99 AA BB CC 90 FE”

M0 is ON when there is no error in the data.

After the receiving of the data sent from the DVP-ES2, the PLC confirms the data format sent from DVP-ES2. If there is no error in the format, M0 is ON; otherwise, M1 is ON. M2 is ON if there is no response. The corresponding special flags SM100, SM102 and SM104 are ON as well.

3. The following table lists the content values in D10–D17 of ES3 CPU.

Device	D10	D11	D12	D13	D14	D15	D16	D17
Value (16#)	1234	5678	1122	3344	5566	7788	99AA	BBCC

Additional remarks

1. If you declare the operand in IPSSoft, the data type is ARRAY [3] of BOOL.
2. If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON and the error code in SR0is 16#2003.

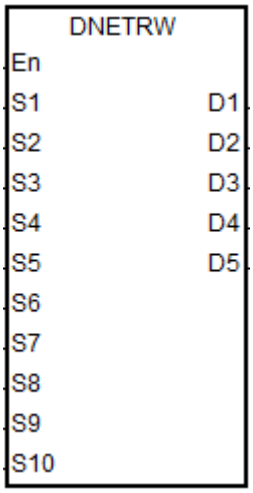
API	Instruction		Operand	Description
1818		DNETRW		S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, D_1, D_2, D_3, D_4, D_5	Reading and writing DeviceNet communication data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7								●	●				○	○		
S_8								●								
S_9								●	●				○	○		
S_{10}								●	●				○	○		
D_1		●	●	●												
D_2		●	●	●												
D_3								●								
D_4								●								
D_5								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S_5		●			●	●							
S_6		●			●	●							
S_7		●			●	●							
S_8		●			●	●							
S_9		●			●	●							
S_{10}		●			●	●							
D_1	●												
D_2	●												
D_3		●			●	●							
D_4		●			●	●							
D_5		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
–	ES3	–

Symbol



S_1 : Number of the module sending the DeviceNet communication

S_2 : DeviceNet MAC ID

S_3 : Service Code

S_4 : Class ID

S_5 : Instance ID

S_6 : Attribute ID

S_7 : Written-data length

S_8 : Device for storing written data

S_9 : Communication timeout time

S_{10} : Number of times of retransmission

D_1 : Completion flag

D_2 : Error flag

D_3 : Error code

D_4 : Read-data length

D_5 : Device for storing read data

Explanation

1. **S_1** is the serial number of the module at the right of the PLC. The first one is number 1, the second one is number 2 and so on. Whatever modules at the right of the PLC must be numbered. The setting range is 1~32. If the setting value (<1 or >32) exceeds the range, the instruction will run at the minimum value or maximum value.

2. **S_2** is the DeviceNet Mac ID within the range: 0~63. It can be the Mac ID of the slave which the master is to read and write as well as the Mac ID of the master which means to read and write the data in the master.

3. **S_3** is the DeviceNet service code.

Service code	Description
0x01	Read all attributes (Get_Attribute_All)
0x02	Set all attributes (Set_Attribute_All)
0x0E	Read one single attribute (Get_Attribute_Single)
0x10	Set one single attribute (Set_Attribute_Single)

4. **S_4**, **S_5** and **S_6** are respectively class ID, instance ID and attribute ID for specifying the parameter path in the EPATH of the DeviceNet protocol.

5. **S_7** is the length of written data with the unit of byte.

6. **S_8** is the starting address of the devices where written data are stored in the order from low bit to high bit.

7. **S_9** is the communication timeout time. The range of the setting value is 1~100 and the unit is 0.1s (second).

8. **S_{10}** is the number of retransmission times and the range is 0~3. When the communication timeout occurs, the communication will be transmitted again.

9. **D_3** displays the read or write error code.

Error code		Description
Code 1 (High Byte)	Code 2 (Low Byte)	
XX	FF	Not conform with DeviceNet standard
20	01	The target slave does not exist.
20	02	Making the connection with the slave failed.
20	03	Sending the explicit message failed.
16	00	Communication timeout

10. **D_4** is the read data length with the unit of byte.

11. **D_5** is the starting address of the devices where read data are stored in the order from low bit to high bit.

12. **D_1** is the communication completion flag and **D_2** is the error flag. See the detailed communication sending procedure and sequence diagram.

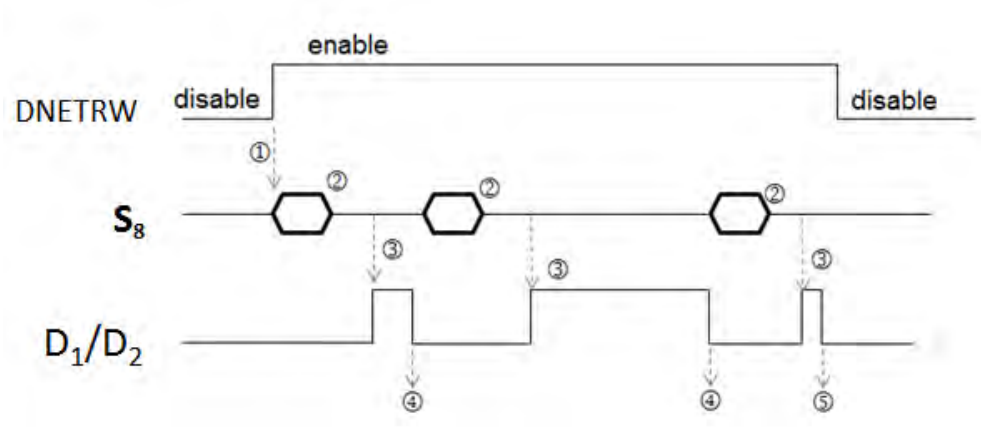
    ① The DNETRW instruction is enabled for the first time and the command will be sent out immediately if the instruction is not be occupied by other DeviceNet communication.

    ② The command is being sent.

    ③ The data sending is completed and the completion flag or error flag changes to On according to the response.

    ④ The next message is ready to be sent out. The next command is sent out right after the completion flag or error flag is cleared.

    ⑤ The command sending is completed and the DNETRW instruction is disabled.



13. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

API	Instruction	Operand	Description
1819	CANRS	**S_1，S_2，S_3，S_4，D_1，D_2，D_3**	User-defined CAN communication sending and receiving

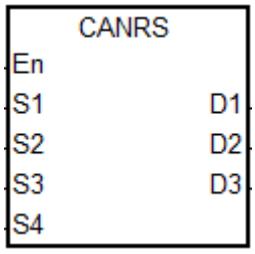
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●							
S_3								●	●				○	○		
S_4								●	●							
D_1								●								
D_2		●	●	●				●								
D_3		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
D_1		●			●	●							
D_2	●												
D_3	●												

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

6

Symbol



S_1 : Communication mode setting

S_2 : Communication ID (MsgID)

S_3 : Sent-data length

S_4 : Starting device where sent source data are stored

D_1 : Starting device where received data are stored

D_2 : Communication completion flag

D_3 : Communication error flag

Explanation

1. Before the CANRS instruction is executed, please ensure that HWCONFIG for the hardware configuration has selected CANopen DS301.

2. **S_1** sets the communication mode. See the following modes that the instruction supports. If the setting value is incorrect, the error flag **D_3** changes to On and the error code SR659 is set to 1.

Communication code		Setting value of S1	Communication format	Description
Higher 8 bits	Lower 8 bits			
0	0	16#0000	2.0A 11-bit ID	Master mode. The master will wait to receive the response message after sending out the communication data.
	1	16#0001		Master mode. After sending the broadcast command, the master will enter the mode of receiving messages from multiple slaves until the receiving timeout occurs or the quantity of messages received is up to 100.
	2	16#0002		Slave mode. The slave receives a message first and then sends back the response message.
	3	16#0003		Slave mode. The slave only receives messages.
1	0	16#0100	2.0B 29-bit ID	Master mode. The master will wait to receive the response message after sending out a message.
	1	16#0101		Master mode. After sending the broadcast command, the master will enter the mode of receiving messages from multiple slaves until the receiving timeout occurs or the quantity of messages received is up to 100.
	2	16#0102		Slave mode. The slave receives a message first and then sends back the response message.
	3	16#0103		Slave mode. The slave only receives messages.

3. **S_2** is the ID of the transmitted message. According to 2.0A or 2.0B protocol, the transmitted data automatically occupies D buffer registers.

   When 2.0A is selected, **S_2** is 11 bits of ID code with the following data transmission format.

S_2 No.	**S_2**
Description	Msg. ID

   When 2.0B is selected, **S_2** (Lo-word) and **S_2**+1 (Hi-word) are both 29 bits of ID code.

S_2 No.	**S_2**	**S_2**+1
Description	Msg. ID（Lo-word）	Msg. ID（Hi-word）

4. **S_3** is the length of the transmitted message. The setting value should be in the range of 0~8 with the unit of byte (8bits). If the setting value (<0 or >8) exceeds the range, the instruction will run at the minimum value 0 or the

maximum 8. If the length of the transmitted message is 0, the communication mode will automatically change into the slave mode to receive messages without sending out any data. The mode can be used to monitor the communication packet.

5. **S_4** is the starting device where transmitted data are stored and only the following 8 bits of data are used.

   For example, 4 messages are transmitted with D10 as the starting device. See the data transmission sequence as below.

S_4 No.	D10	D11	D12	D13
Description	Data1	Data2	Data3	Data4

6. If **S_1** is the master mode in which the master will wait to receive data after sending data or the slave mode, the received data will be directly stored in the device specified by **D_1**. D100 is specified by D_1 Here See the stored content format.

   2.0A mode setting:

D_1 No.	D100	D101	D102 ~ D109 (Lower 8 bits)
Description	Msg. ID	Data Length	Data1 ~ Data8

   2.0B mode setting

D_1 No.	D100	D101	D102	D103 ~ D110 (Lower 8 bits)
Description	Msg. ID（Lo-word）	Msg. ID（Hi-word）	Data Length	Data1 ~ Data8

6

   Note: If the Msg. ID to be received need be specified at the stage of receiving data, set the value of **D_1/ D_1+1** beforehand based on the 2.0A/2.0B mode. If the Msg. ID is not specified, please clear the value of **D_1/ D_1+1** to 0 before receiving data.

7. If **S_1** is the master broadcast mode, the received data will be stored in the device specified by **D_1**. D100 is specified by **D_1** here. See the storage format as below.

   Selecting 2.0A mode: (Here is the introduction of receiving data from 2 slaves. For other data, please increase the Device number specified by **D_1**)

Response sequence		Data from the first slave			Data from the second slave		Data from the third slave
D_1 No.	D100	D101	D102	D103 ~ D110 (Lower 8 bits)	D111	D112~D120	D121~130
Description	Receiving Packet number	Msg. ID	Data Length	Data1 ~ Data8	Msg. ID	Length, Data	ID, Length, Data

Selecting 2.0B mode: (Here is the introduction of receiving data from 2 slaves. For other data, increase the number of **D_1**)

Response sequence		Data from the first slave				Data from the second slave
D_1 No.	D100	D101	D102	D103	D104 ~ D111 (Lower 8 bits)	D112~D122
Description	Receiving Packet number	Msg. ID （Lo-word）	Msg. ID （Hi-word）	Data Length	Data1 ~ Data8	ID, Length, Data

8. **D_2** and **D_3** are the receiving-completed flag and error flag respectively. The two flags will judge if the data receiving is completed or an error occurs based on the current communication state when the instruction is scanned. If the data receiving is completed or an error occurs, the flags will change to On. The two flags will be cleared and change to Off automatically every time the instruction is enabled.

9. When the instruction is set to the master mode, using the receiving timeout setting in HWCONFIG is recommended. So **D_3** will change to On and the error code 2 will be recorded in SR659 if the communication packet has not been received fully within the specified period of time.

   If the receiving timeout time is set to 0, it indicates that the communication timeout is not limited and the status can be applied to the slave mode.

   Note: The communication control right can not be released until the instruction is disabled by manual when the method in which there is no limit to the communication timeout is adopted.

10. There is no limit to the number of times of using the instruction. But only one CAN communication command is allowed to be sent every time. If one command is being sent or received currently, the next CANRS instruction can not be enabled.

11. Here is the CAN BUS format and every bit of content for Msg. ID is explained as below.

    As 2.0A protocol is selected and the value of **S_2** is 16#0123, the Msg. ID content is shown in the following table.

Bit No.	15 ~ 11	10 ~ 8	7 ~ 4	3 ~ 0
S_2 value (16bits)	-	1	2	3

    As 2.0B protocol is selected, the value of **S_2** is set to 16#1234 (Lo-word) and **S_2**+1 is 16#0567 (Hi-word), the Msg. ID content is shown in the following table.

Bit No.	31 ~ 29	28	27 ~ 24	23 ~ 20	19 ~ 16	15 ~ 0
S_2 value (32bits)	-	0	5	6	7	1234

12. If the operand **D_1** is declared in a variable, using the Word-type matrix is recommended.

13. See CANRS communication sequence diagram and explanation.

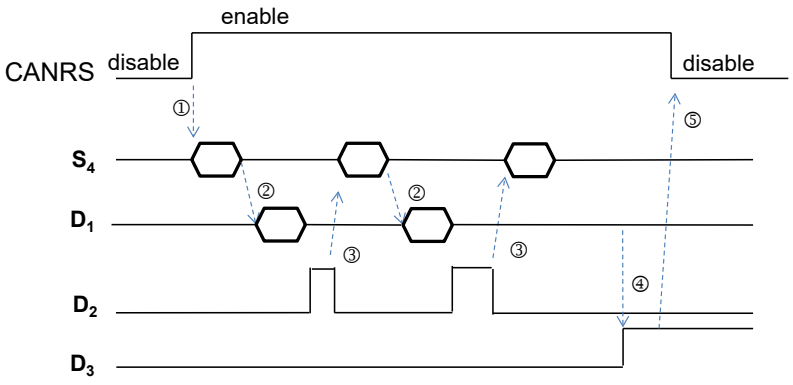
①：The CANRS instruction is enabled. If no other CANRS instruction occupies the control right, the flags **D_2** and **D_3** are cleared and then the data are sent out right away.

②：The data sending is completed and meanwhile the data receiving starts. After the receiving is completed, the completion flag **D_2** changes to On.

③：The instruction sends data again after you clear the flag **D_2**.

④：As the receiving timeout occurs, **D_3** changes to On.

⑤：If an error is found, you can disable the instruction. Alternatively, you can have the instruction resend data after clearing the flag **D_3**.



Example 1

Master Mode

Using the CANRS instruction, the 6-byte data in D10~D15 are transmitted to the slave and the response data from the slave are stored in the devices starting from D100. M100 changes to On when the sending and receiving are over.

See the explanation of relevant parameters.

Master mode_MsgID=2	
Communication mode	0= master mode, waits to receive the response from the slave after sending out data.
Communication ID	The MsgID of the sent message D0=1
Sent-data length	6 BYTE

Starting device where sent data are stored	D10
Starting device where received data are stored	D100 for storing MsgID, D101 for storing the number of received packets D102…. for storing received data
Communication completion flag	M100
Communication error flag	M101

1. As M1 is on, set the MsgID of the sent message to 1 and MsgID of the received message to 2. Then the data sending begins.

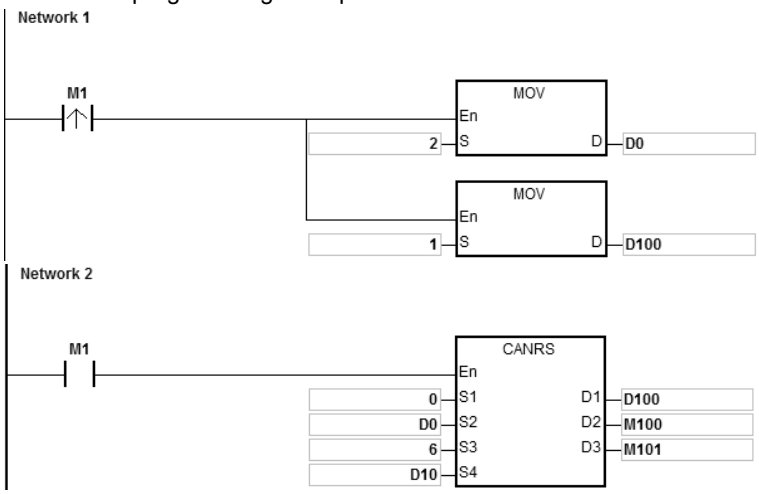
   A. Sending the data in D10~D15 (6 bytes) to the slave is performed.

Device	D10~D15
Content	H0A (Defined by users)

   B. The received response data from the slave are stored in D100~D109. M100 changes to On as the receiving is done.

Device	D100	D101	D102~D109
Content	2	8	H0B

2. Master programming example



Example 2

Slave Mode

Using the CANRS instruction, the received data from the master are stored in the devices starting from D120 and the 8-byte data in D20~D27 are sent back. M110 changes to On when the sending and receiving are over.

See the explanation of relevant parameters.

Slave mode_MsgID=1	
Communication mode	2= Slave mode, sends data after receiving data.
Communication ID	MsgID of the sent message, D0=2
Sent-data length	8 BYTE
Starting device where sent data are stored	D20
Starting device where received data are stored	D120 for storing MsgID D121 for storing the number of received packets D122…. for storing received data
Communication completion flag	M110
Communication error flag	M111

1. As M1 is on, set the MsgID of the sent message to 2 and MsgID of the received message to 1 and then wait to receive data.

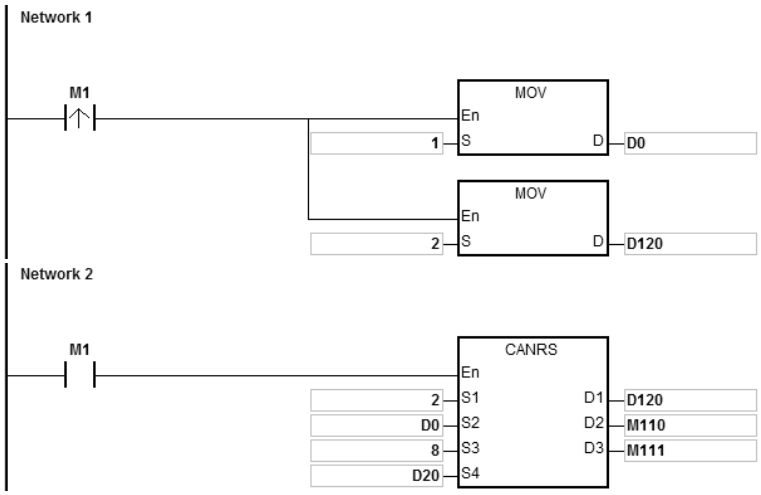
   A. The received 6-byte data from the master are placed in D120~D127.

Device	D120	D121	D122~D127
Content	1	6	H0A

   B. Then the 8-byte data in D20~D27 start to be sent back to the master. M110 changes to On as the data sending is done.

Device	D10~D15
Content	H0B (Defined by users)

2. Slave programming example



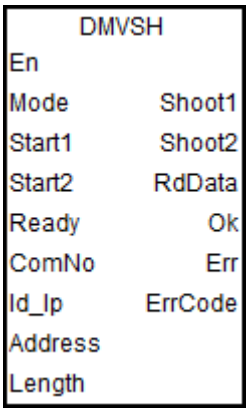
API	Instruction	Operand	Description
1820	DMVSH	**Mode ~ ErrCode**	Enabling Delta DMV detection and communication

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Mode								●					○	○		
Start1		●	●	●				●								
Start2		●	●	●				●								
Ready	●															
ComNo								●					○	○		
Id_Ip								●					○	○		
Address								●					○	○		
Length								●					○	○		
Shoot1		●														
Shoot2		●														
RdData								●								
Ok		●	●	●				●								
Err		●	●	●				●								
ErrCode								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Mode		●											
Start1	●												
Start2	●												
Ready	●												
ComNo		●			●	●							
Id_Ip		●			●	●							
Address		●			●	●							
Length		●			●	●							
Shoot1	●												
Shoot2	●												
RdData		●			●	●							
Ok	●												
Err	●												
ErrCode		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



Mode : The triggering and receiving method

Start1 : Trigger signal for starting set1

Start2 : Trigger signal for starting set2

Ready : Completion signal of when receiving response data from DMV is complete

ComNo : A serial port or network communication port for the PLC to send or receive data

Id_Ip : Serial communication station address or network IP of DMV

Address : Starting device where DMV data to be read are stored

Length : Length of the data to be read

Shoot1 : Output signal when PLC notifies DMV to enable the detection of set1

Shoot2 : Output signal when PLC notifies DMV to enable the detection of set2

RdData : Starting device where the data that the PLC receives from the DMV are stored

Ok : Communication success flag

Err : Communication error flag

ErrCode : Communication error code

Explanation

1. **Mode** sets the method through which the PLC triggers DMV including DMV1000 and DMV2000 and the receiving method. There are 2 modes: 0 and 1 for option. If the setting exceeds the range, PLC will automatically run in mode 0.

2. Please refer to following example explanation for the function of **Start1**, **Start2**, **Ready**, **Shoot1**, **Shoot2**, **RdData** and **Ok** and the process of detection triggering.

3. **ComNo** sets a communication port number of the PLC. The number 1 represents COM1, number 2 represents COM2, and number 4 represents Ethernet port. If the setting is not one of the numbers mentioned above or represents the communication port that the PLC does not support, the error flag **Err** will change to On and the error code 16#0001 will show up.

4. **Id_Ip** sets the communication station address (Mac ID) of the slave DMV or network IP. **Address** is the communication address where DMV detection result is read. **Length** is the length of the read detection data.

5. See the explanation of the values of **ErrCode** in the table below.

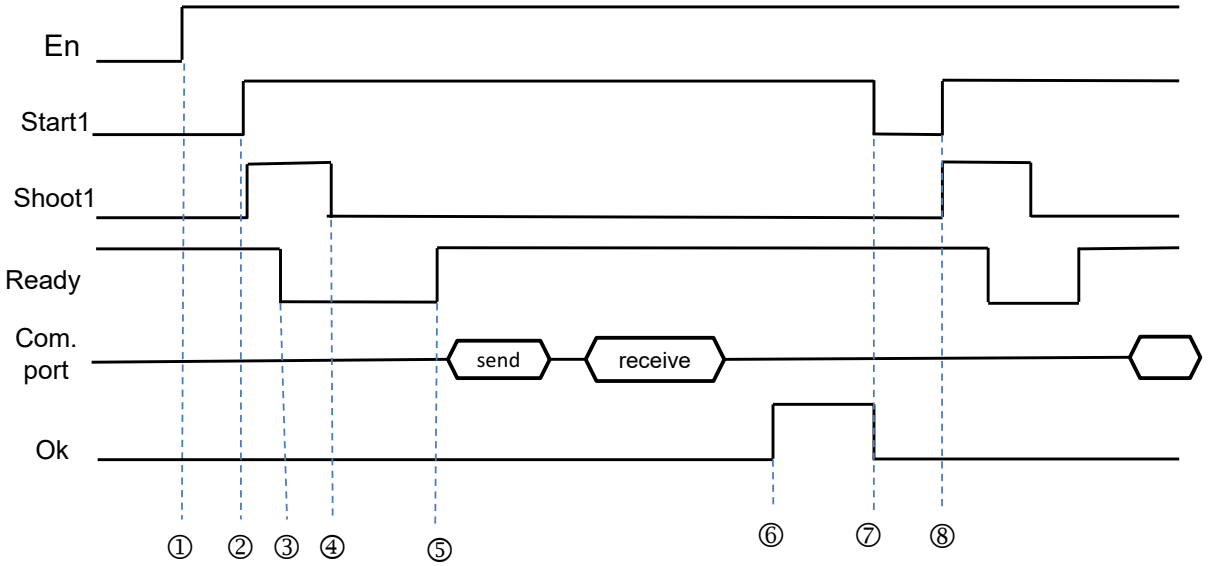
Error code	Description	Correction
16#0001	The specified communication port is incorrect.	Specify a new communication port number.
16#0002	DMV response fault	Check the communication address
16#0003	No response from DMV	Check if the cable is connected properly or DMV has been enabled.
16#0004	Waiting for the response from DMV; DMV detection cannot be enabled repeatedly.	Modify the timing of enabling DMV detection.

Example 1

Mode=0

Only set1 is used to trigger the DMV detection and receiving of DMV response data.

See the sequence control diagram below.



Explanation of the timings:

① The DMVSH instruction is enabled.

② Set Start1 to On to notify the instruction to send the output signal **Shoot1** (ON for about 10ms) and notify DMV to enable the detection function.

③ After DMV receives the trigger message, **Ready** is set from On to Off.

④ **Shoot1** is cleared to Off automatically by the PLC.

⑤ After DMV detection is finished, **Ready** is set from Off to On. The PLC starts to judge if **Ready** is on after waiting for 1.5 times the input filtering time. The PLC receives the message that **Ready** changes from Off to On and sends a read command via Modbus 0x03.

⑥ The **Ok** flag is set to On after the PLC receives the communication data sent back from DMV. At the moment, the already received data can be processed.

⑦ The **Start1** and **Ok** flags can be cleared to Off by manual.

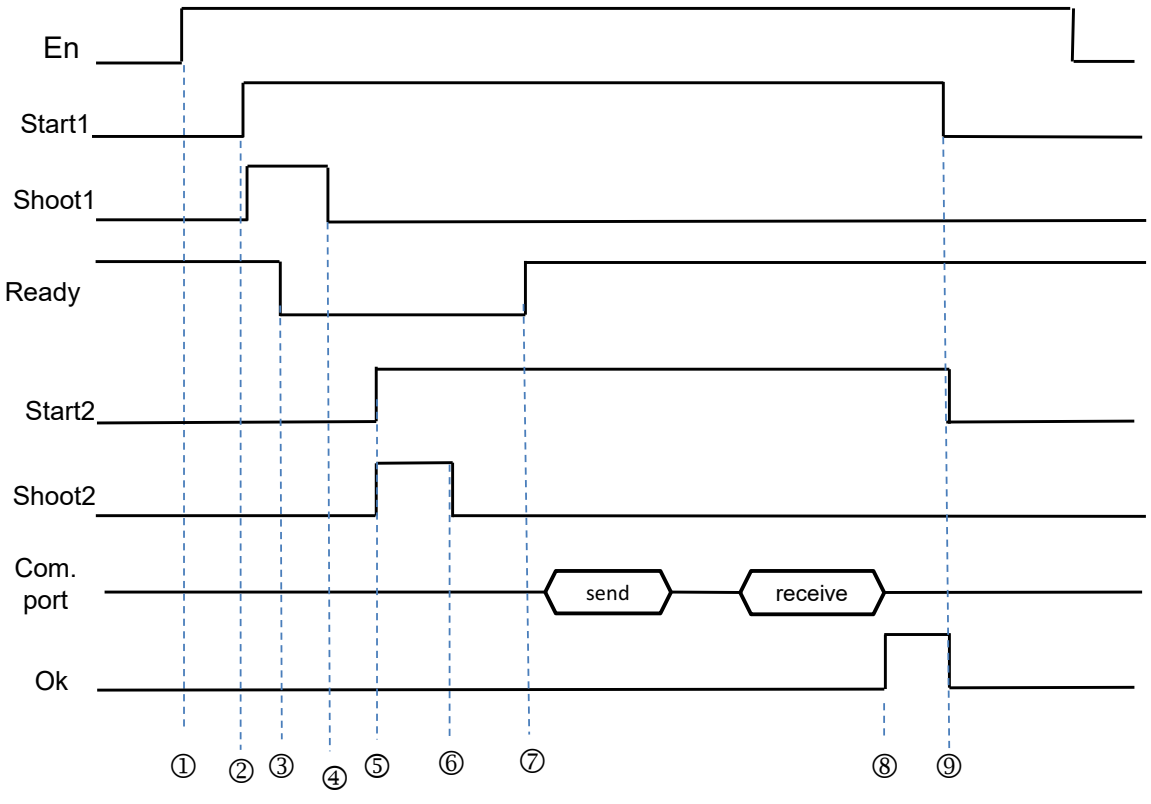
⑧ If the detection need be retriggered to execute, directly set **Start1** to On to start the next-round DMV detection. (Return to step 2 to continue.)

Example 2

Mode = 1

Set1 and set2 are used to trigger the DMV detection separately. But the communication data reading command can not be sent until both **Readys** change to On. (The example is applicable to the two-camera DMV1000 detection.)

See the sequence control diagram below.



① The DMVSH instruction is enabled.

② Set **Start1** to On to notify the instruction to send the output signal **Shoot1** (which is on for about 10ms) and notify DMV to enable the detection function of the first camera.

③ After DMV receives the trigger message, **Ready** is set from On to Off.

④ **Shoot1** is cleared to Off automatically by the PLC.

⑤ Set **Start2** to On to notify the instruction to send the output signal **Shoot2** (which is on for about 10ms) and notify DMV to enable the detection function of the second camera.

⑥ **Shoot2** is cleared to Off automatically by the PLC.

⑦ After DMV detection is finished, **Ready** is set from Off to On and the two detections are complete. Then the PLC sends a read command via Modbus 0x03.

⑧ The **Ok** flag is set to On after the PLC receives the communication data sent back from DMV. At the moment, the already received data can be processed.

⑨ The **Start1**, **Start2** and **Ok** flags can be cleared to Off by manual. If the detection need be retriggered to execute, return to step 2 to proceed.

6.19.3 Descriptions of the Communication-related Flags and Registers

Communication-related flags (SM)

Flag		Description	Action
COM1	COM2		
SM96	SM97	Data sending request flag If you want to use the instruction to send and receive data, you must use the pulse instruction to set the flag to ON. When the instruction is executed, the PLC sends and receives the data. After sending the data, the system automatically resets the flag to OFF.	You set the flag to ON, and the system automatically resets it to OFF.
SM98	SM99	When the flag is ON, the PLC is waiting to receive the data.	The system automatically sets the flag to ON and resets it to OFF.
SM100	SM101	Reception complete flag After receiving the data, the system automatically sets the flag to ON. When the flag is ON, the data received can be processed. After processing the data received, you must to reset flags to OFF.	The system automatically sets the flag to ON, and you reset it to OFF.
SM102	SM103	Data receiving error flag An error occurs during receiving data using the data receiving instruction.	The system automatically sets the flag to ON, and you reset it to OFF.
SM104	SM105	Communication timeout error flag If you set the communication timeout (in SR) and no data is received after the timeout period, the flag is ON. After the problem is solved, you must reset the flag to OFF.	The system automatically sets the flag to ON, and you reset it to OFF.
SM106	SM107	The choice between the 8-bit processing mode and the 16-bit processing mode ON: 8-bit processing mode OFF: 16-bit processing mode	You set the flag to ON and reset it to OFF.
SM210	SM212	Communication mode ON: RTU mode OFF: ASCII mode You can set this value in HWCONFIG in ISPSoft. Not available for Card1 and Card 2.	You set the flag to ON and reset it to OFF.

Flag		Description	Action
COM1	COM2		
SM209	SM211	Communication protocol changed flag The communication protocol changes in accordance with the setting values in SR. If the flag is set to ON, the communication protocol changes in accordance with the setting values in SR and then the system automatically resets the flag to OFF. NOTE: this change does not affect the parameters that you set in HWCONFIG. When the PLC is powered-on again, the PLC operates according to the communication protocol set in HWCONFIG.	You set the flag to ON and reset it to OFF.

NOTE: the above flags are non-latching types.

Communication-related registers (SR)

Special data register		Description
COM1	COM2	
SR201	SR202	Communication port address
SR209	SR212	Communication protocol For details, please refer to the following table to set up the communication format for a serial communication port.
SR210	SR213	Communication timeout, unit: ms Suppose the setting value is larger than 0. When the PLC executes the communication instruction and is in the receiving state, but no data is received after the timeout period or the intervening time between two characters exceeds the setting value, the timeout flag is ON. You can set the register to 0 to disable communication timeout monitoring. See the MODRW instruction set between 100–32767 (ms).

Setting the communication format for a serial communication port

b0	Data length			7 (value=0)		8 (value=1)
b2-b1	Parity bits			00	:	None
				01	:	Odd
				10	:	Even
b3	stop bits			1 bit (value=0)		2 bits (value=1)
b7-b4	0001	(16#1)	:	4800		
	0010	(16#2)	:	9600		
	0011	(16#3)	:	19200		
	0100	(16#4)	:	38400		
	0101	(16#5)	:	57600		
	0110	(16#6)	:	115200		
	0111	(16#7)	:	230400		Not available for RS-232
	1000	(16#8)	:	500000		Not available for RS-232
	1001	(16#9)	:	921000		Not available for RS-232
	1111	(16#F)	:	Self-defined*1		
b8-b15	Undefined (reserved)					

*1: You can set the baud rate in HWCONFIG in ISPSoft.

The data transmission speed is listed in the following table.

Baud rate (bps)	RTU timeout timer (ms)	Baud rate (bps)	RTU timeout timer (ms)
4800	9	115200	1
9600	5	230400	1
19200	3	-	-
38400	2	-	-
57600	1	-	-

6.20 Other Instructions

6.20.1 List of Other Instructions

The following table lists the Other Instructions covered in this section.

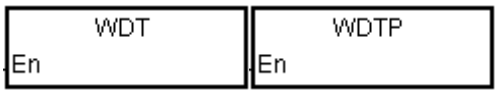
API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
1900	WDT	–	✓	Watchdog timer
1901	DELAY	–	✓	Delaying the execution of the program
1902	GPWM	–	–	General pulse width modulation
1904	EPUSH	–	✓	Storing the contents of the index registers
1905	EPOP	–	✓	Reading data into the index registers
1906	INFO	–	✓	Reading the system data

6.20.2 Explanation of Other Instructions

API	Instruction code		Operand	Function
1900	WDT	P	—	Watchdog timer

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	—

Symbol



Explanation

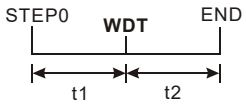
1. This instruction resets the watchdog timer to monitor the operation of the ES3 Series PLC system.

2. If the program scanning time in the PLC exceeds 200 milliseconds, the error LED indicator is ON, and the PLC stops running.

3. The watchdog timer acts in the following cases.

   - The system is behaving abnormally.
   - Program execution takes too much time, and therefore the scan time is longer than the watchdog timer setting value. There are two way you can improve the situation.

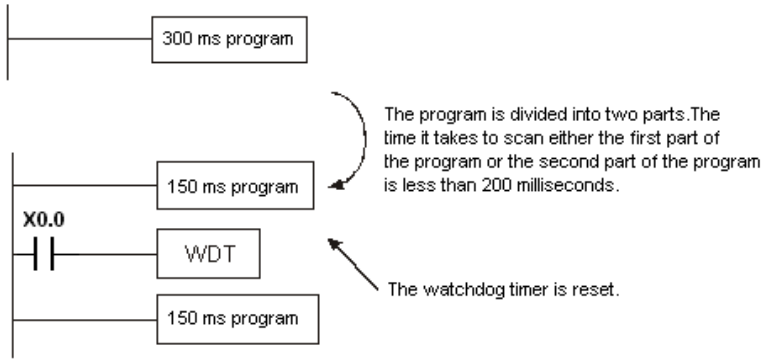
     (a) Use the WDT instruction to reset the watchdog timer.



     (a) Refer to the ISPSoft User Manual for more information about changing the watchdog timer setting value.

Example

Suppose the program scanning time is 300 milliseconds. After the program is divided into two parts, and you insert the instruction WDT between these two parts, the time it takes to scan either the first part of the program or the second part of the program must be less than 200 milliseconds.



API	Instruction code			Operand	Function
1901		DELAY	P	**S**	Delaying the execution of the program

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	–

Symbol

S : Delay

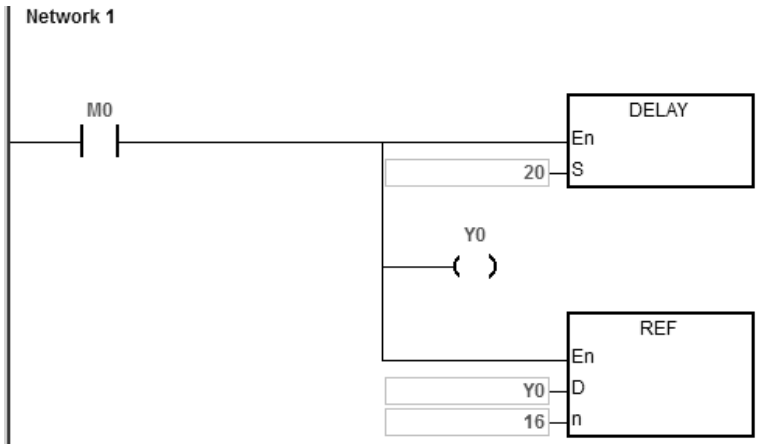
Explanation

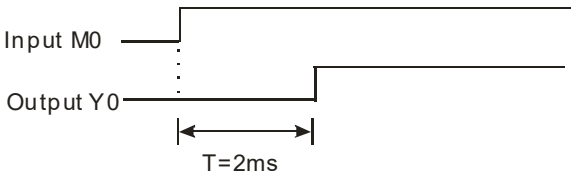
This instruction delays the execution of the program following the DELAY instruction for a period of time specified in **S**.

The unit of **S** is 0.1 milliseconds.

Example

When M0 is ON, the instruction delays the execution of the program following the DELAY instruction for two milliseconds. That is, Y0 is ON and the states of Y0–Y7 and Y10–Y17 are refreshed two milliseconds after the DELAY instruction is executed.





Additional remarks

1. If **S** I less than 0, there is no delay.
2. If **S** is larger than 1000, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. You can adjust the delay according to the actual conditions of your equipment.
4. The delay can be larger than **S** due to communication or other influences.

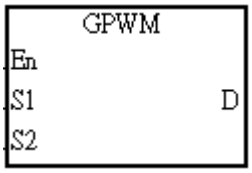
API	Instruction code	Operand	Function
1902	GPWM	**S_1, S_2, D**	General pulse width modulation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●			○	○				
S_2					●	●		●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
–	ES3	–

Symbol



S_1 : Pulse width

S_2 : Pulse cycle

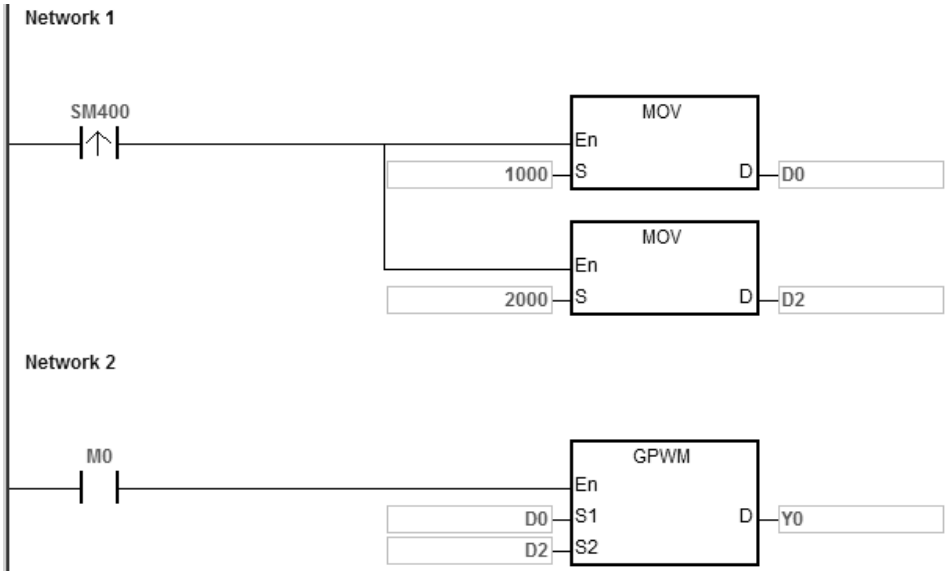
D : Output device

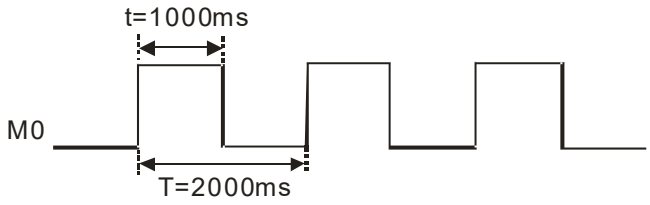
Explanation

1. This instruction outputs every pulse with a width specified by **S_1** and with a cycle specified by **S_2** from the device specified by **D**.
2. The pulse width specified by **S_1** is t. t must be between 0–3276 milliseconds.
3. The pulse cycle specified by **S_2** is T. T must be between 1–32767 milliseconds, and **S_1** should be less than **S_2**.
4. The **S_2**+1 and **S_2**+2 parameters are for system use. Please do not change them.
5. If **S_1** is less than 0, there is no pulse output. If **S_1** is larger than **S_2**, the output device stays ON.
6. You can change **S_1** and **S_2** during the execution of this instruction.
7. If the conditional contact is not enabled, there is no pulse output.
8. When using on-line editing, please reset the conditional contact to initialize the instruction.

Example

When the program is executed, the values in D0 and D2 are 1000 and 2000 respectively. When M0 is ON, the device Y0 outputs the pulses illustrated below. When M0 is OFF, Y0 is OFF.





Additional remarks

1. The instruction counts by the scan cycle. Therefore, the maximum error is one scan cycle. Besides, **S_1**, **S_2**, and (**S_2**-**S_1**) should be larger than the scan cycle; otherwise, an error occurs when the GPWM instruction is executed.
2. If you use the instruction in a function block or an interrupt task, an inaccurate pulse output occurs.
3. If you declare the operand **S_2** in ISPSoft, the data type is ARRAY [3] of WORD/INT.

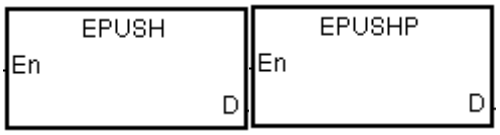
API	Instruction code			Operand	Function
1904		EPUSH	P	**D**	Storing the contents of the index registers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	–

Symbol



D : Device to store the value in the index registers

Explanation

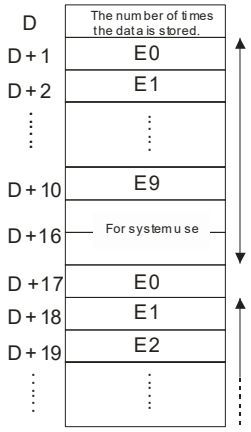
1. This instruction stores the values in E0–E9 in the devices specified by **D**. The valid range for **D** is between 0–99. The instruction is not executed for values that exceed the range.

2. The instruction uses sixteen devices, and the last six devices are for system use. If you execute the instruction and the number of times the data is stored is n (the value in **D**), the instruction stores data in E0–E9 in **D**+(16*n+1)–**D**+(16*n+16), and increments the value in **D** to n+1.

3. The storage for the device D is 1601 (16x100+1).

4. This instruction uses a pulse instruction to interact with the stack, pushing a value onto the stack. Therefore, you must reset the contact before the next operation.

5. When you use this instruction with the EPOP instruction (API 1905), the value stored last in the device specified by **D** is read first, following the LIFO (last in first out) principle.



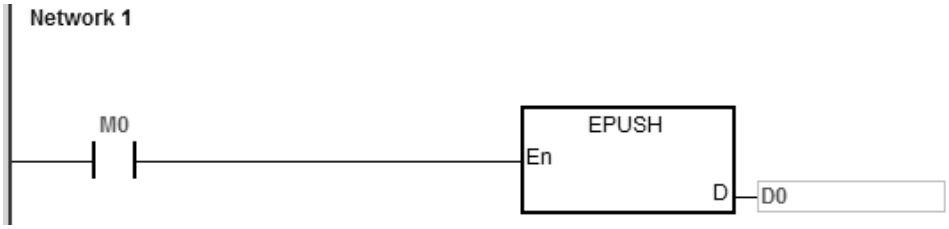
Example

Suppose the value in D0 is 0.

When M0 is ON for the first time, the instruction transmits the data in E0–E9 to D1–D10 and increments the value in D0 to 1.

When M0 switches from OFF to ON for the second time, the instruction transmits the data in E0–E9 to D17–D26, and increments the value in D0 to 2.

When M0 switches from OFF to ON for the n^{th} time, the instruction transmits the data in E0–E9 to (n*16)+1–(n*16)+10.



Additional remarks

1. If the value in **D** is not between 0–99, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the range of devices is not sufficient for **D**+((the value in **D**)+1)*16-1), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

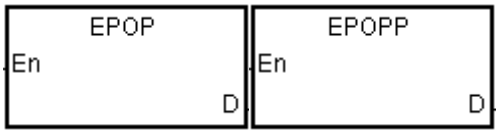
API	Instruction code			Operand	Function
1905		EPOP	P	**D**	Reading the data into the index registers

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	–

Symbol



D : Device from which the value is read

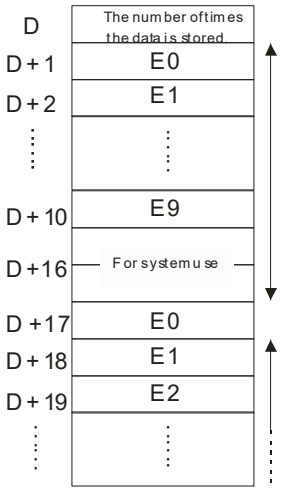
Explanation

1. This instruction reads the values in the devices specified by **D** into E0–E9, and decrements the value in **D** by one. The valid value for **D** is between 1–100. The instruction does not execute values that exceed the range.

2. The instruction involves sixteen devices, and the last six devices are for system use. If you execute the instruction and the number of times the data is stored is n (the value in **D)**, the instruction stores the data in E0–E9 in **D**+16*(n-1)+1–**D**+16*(n-1)+10, and decrements the value in **D** to n-1.

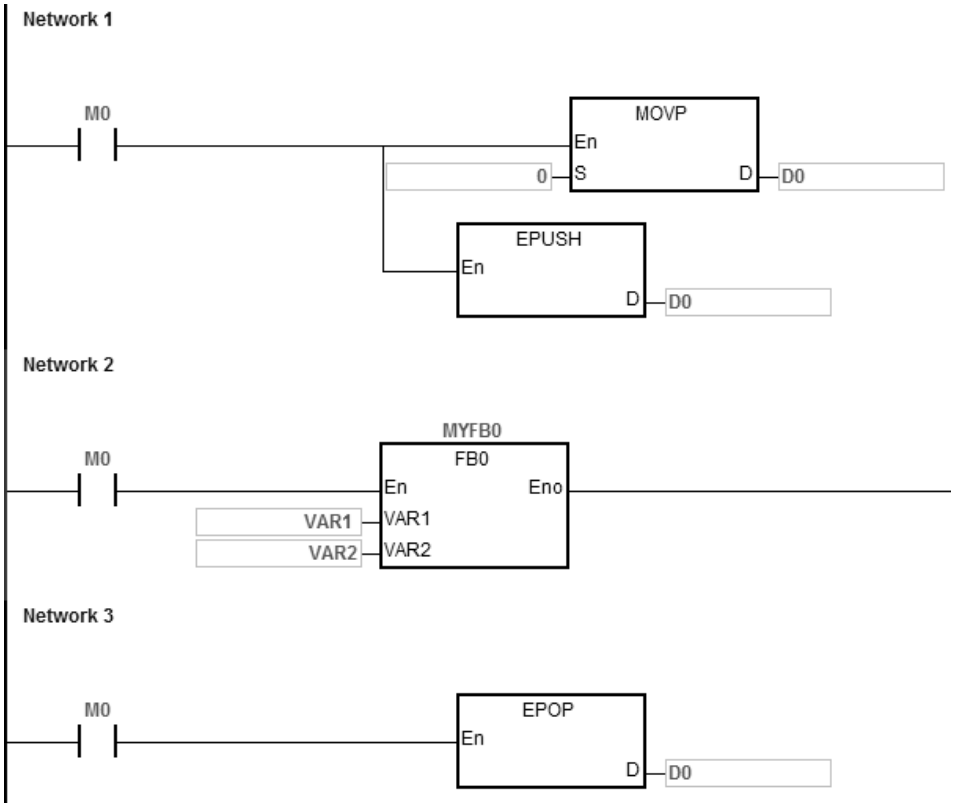
3. This instruction uses a pulse instruction to interact with the stack, taking the TOP VALUE from the stack and assigning it to the specified variable; therefore, you must reset the contact before the next operation.

4. The value that the instruction stores last in the device specified by **D** is read first, following the LIFO (last in first out) principle.



Example

When M0 is ON, the MOV instruction sets the value in D0 is set to 0, and the EPUSH instruction transmits the values in E0–E9 to D1–D10. After the execution of FB0 is complete, the EPOP instruction reads the values in D1–D9 into E0–E9.



Additional remarks

1. If the value in **D** is not in the range 0–100, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the range of device is not sufficient for **D**+((the value in **D**) *16-1), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

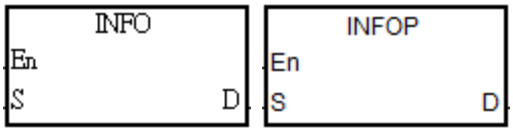
API	Instruction		Operand	Description
1906	INFO	P	**S**，**D**	Reading the system data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S													○			
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●				●							
D		●			●	●							

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	—

Symbol



S : System-data reading code

D : Starting device where the read system data are stored

Explanation

1. **S** is the system information reading code and the codes are listed in the following table.

S code No.	Description	Number of consecutive **D** devices (Number of Words)	Remark
0	Reads the serial number of the product	9	ASCII code
1	Reads Ethernet Mac ID	3	Hex value
2	Reads already-power-on time	2	32-bit value, unit: minutes
others	Reserved	0	The instruction is not executed.

2. If the product serial no. is 32ES311TW19450012, the reading code of the instruction is 0 and the read data are stored in the devices starting from D0, the product serial no. is stored in the order as below.

D0	D1	D2	D3	D4	D5	D6	D7	D8
16#3233	16#5345	16#3133	16#5431	16#3157	16#3439	16#3035	16#3130	16#0032
‘32’	‘ES‘	‘31’	‘1T’	‘W1’	‘94’	‘50’	‘01’	‘2’

3. If Mac ID is 16#123456789ABC, the reading code of the instruction is 1 and the read data are stored in the devices starting from D10, the Mac ID data are stored in the order as below.

D10	D11	D12
16#1234	16#5678	16#9ABC

4. If the already-power-on time is 70000 minutes (16#00011170), the reading code of the instruction is 2 and the read data are stored in the devices starting from D20, the already-power-on time data are stored in the order as below.

D20	D21
16#1170	16#0001

5. Only the first-time execution of the INFO instruction is effective when the conditional contact is met. Therefore, it is recommended to use the pulse instruction INFOP. If the INFO instruction is used with the normally-open contact together, the instruction will be automatically executed only once.

Additional remark:

If **D** is declared in a variable and the code is 0, 1 and 2, they can be declared respectively in ARRAY [9], [3], [2] of WORD.

6.21 String Processing Instructions

6.21.1 List of String Processing Instructions

The following table lists the String Processing instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2100	BINDA	DBINDA	✓	Converting a signed decimal number into an ASCII code
2101	BINHA	DBINHA	✓	Converting a binary hexadecimal number into an hexadecimal ASCII code
2102	BCDDA	DBCDDA	✓	Converting a binary-coded decimal number into an ASCII code
2103	DABIN	DDABIN	✓	Converting a signed decimal ASCII code into a signed decimal binary number
2104	HABIN	DHABIN	✓	Converting a hexadecimal ASCII code into a hexadecimal binary number
2105	DABCD	DDABCD	✓	Converting an ASCII code into a binary-coded decimal number
2106	$LEN	–	✓	Calculating the length of a string
2109	$FSTR	–	✓	Converting a floating-point number into a string
2110	$FVAL	–	✓	Converting a string into a floating-point number
2111	$RIGHT	–	✓	Retrieving characters from a string starting from the right
2112	$LEFT	–	✓	Retrieving characters from a string starting from the left
2113	$MIDR	–	✓	Retrieving a part of a string
2115	$SER	–	✓	Searching a string
2116	$RPLC	–	✓	Replacing the characters in a string
2117	$DEL	–	✓	Deleting the characters in a string
2118	$CLR	–	✓	Clearing a string
2119	$INS	–	✓	Inserting a string
2122	SPLIT	–	✓	Splitting a string
2123	MERGE	–	✓	Merging strings

6.21.2 Explanation of String Processing Instructions

API	Instruction code			Operand	Function
2100	D	BINDA	P	**S, D**	Converting a signed decimal number into an ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

S : Source value

D : Device where the conversion result is stored

Explanation

1. This instruction converts the signed decimal binary number in **S** into an ASCII code, and stores the conversion result in **D**.

2. The instruction supports SM690 to control the ending character.

3. For 16-bit instructions, the value in **S** must be a binary number and between -32768 to 32767. The operand **D** occupies four word devices. The instruction converts the data is converted as follows.

b15 b0		b15 b8	b7 b0	
1234	➔	16#32	16#31	**D** + 0
		16#34	16#33	**D** + 1
		16#00	16#00	**D** + 2

If SM690 is OFF, the instruction stores the ending character 16#0000 in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged without the ending character.

If the value in **S** is a positive value, the instruction stores only the value but not the sign character in **D**. If the value in **S** is a negative value, the instruction stores the "-" sign character in **D** (16#2D). For example, if the value in **S** is -12345 and SM690 is OFF, the conversion result is as follows.

b15 b0		b15 b8	b7 b0	
-1234	➔	16#31	16#2D (-)	**D** + 0
		16#33	16#32	**D** + 1
		16#00	16#34	**D** + 2

4. For 32-bit instructions, the value in **S** must be a binary number between -2147483648 to 2147483647. The operand **D** occupies six word devices. The data is converted as follows.

b31 b0		b15 b8	b7 b0	
12345678	➔	16#32	16#31	**D** + 0
		16#34	16#33	**D** + 1
		16#36	16#35	**D** + 2
		16#38	16#37	**D** + 3
		16#00	16#00	**D** + 4

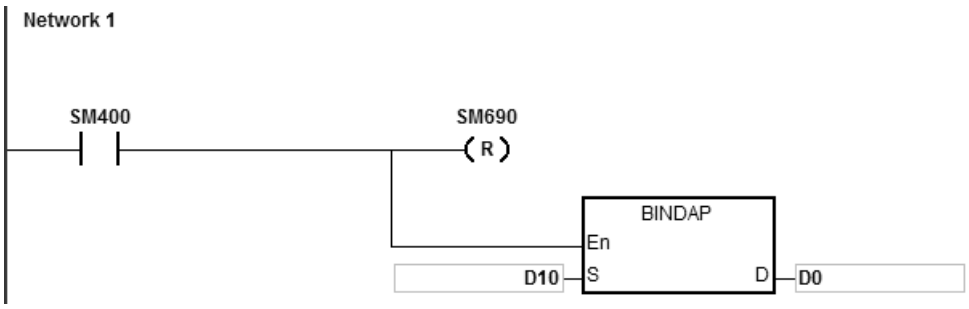
If SM690 is OFF, the instruction stores the ending character 16#0000 in **D**+4. If SM690 is ON, the value in **D**+4 is unchanged without the ending character.

In addition, if the value in **S** is a positive value, the instruction stores only the value but not the sign character in **D**. If the value in **S** is a negative value, the instruction stores the "-" sign character in **D** (16#2D). For example, if the value in **S** is -12345678, and SM690 is OFF, the conversion result is as follows.

b31 b0		b15 b8	b7 b0	
-12345678	➔	16#31	16#2D	**D** + 0
		16#33	16#32	**D** + 1
		16#35	16#34	**D** + 2
		16#37	16#36	**D** + 3
		16#00	16#38	**D** + 4

Example 1

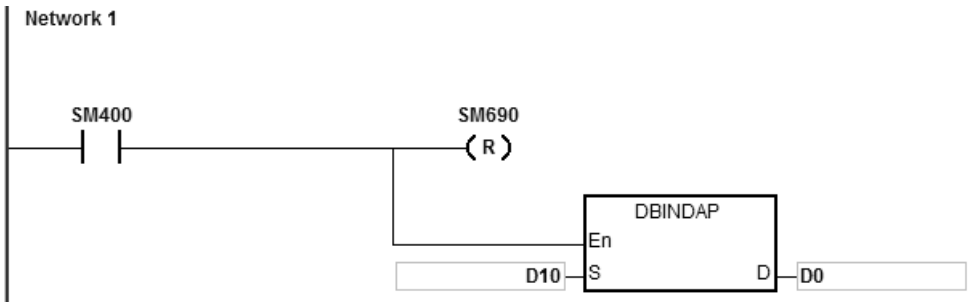
Suppose the value in L0 is 5126 and SM690 is OFF. When the instruction executes, the values are D0=16#3135, D1=16#3632, D2=16#0000.



D10	b15 b8	b7 b0	
5126 ➔	16#31	16#35	D0
	16#36	16#32	D1
	16#00	16#00	D2

Example 2

Suppose the value in D10 is -3842563 and SM690 is OFF. When the instruction is executed, the values are D0=16#332D, D1=16#3438, D2=16#3532, D3=16#3336, D4=16#0000.



D10	b15 b8	b7 b0	
-3842563 ➔	16#33	16#2D	D0
	16#34	16#38	D1
	16#35	16#32	D2
	16#33	16#36	D3
	16#00	16#00	D4

Additional remarks

1. If value in the device **D** is not sufficient for the conversion, SM0 is ON, and the error code in SR0 is 16#2003.
2. For 16-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [4] of WORD/INT.
3. For 32-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [6] of WORD/INT.

API	Instruction code			Operand	Function
2101	D	BINHA	P	**S, D**	Converting a binary hexadecimal number into a hexadecimal ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol

S : Source value

D : Device where the conversion result is stored

6

Explanation

1. This instruction converts the hexadecimal binary number in **S** an ASCII code, and stores the conversion result in **D**.

2. The instruction supports SM690 to control the ending character.

3. For 16-bit instructions, the value in **S** must be between 16#0000–16#FFFF, and should be a four-digit binary number. The operand **D** occupies three word devices.
   If SM690 is OFF, 16#0000 is stored in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged. For example, if the value in **S** is 16#02A6 and SM690 is OFF, the conversion result is as follows.

b15 b0		b15 b8	b7 b0	
02A6	➔	16#32	16#30	**D** + 0
		16#36	16#41	**D** + 1
		16#00	16#00	**D** + 2

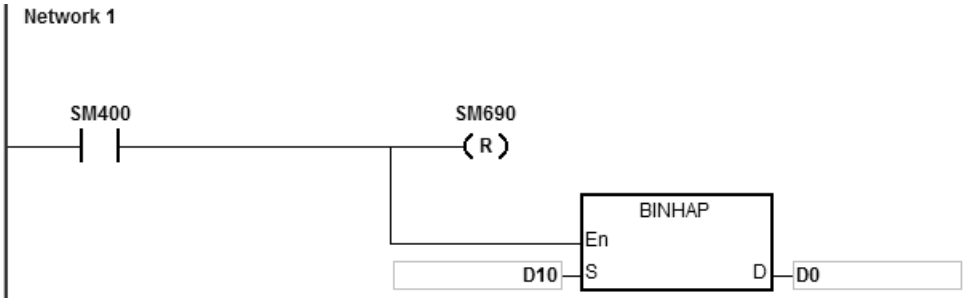
4. For 32-bit instructions, the value in **S** must be between 16#00000000–16#FFFFFFFF, and should be an eight-digit binary number. The operand **D** occupies five word devices.

   If SM690 is OFF, 16#0000 is stored in **D**+4. If SM690 is ON, the value in **D**+4 is unchanged. For example, if the value in **S** is 16#03AC625E and SM690 is OFF, the conversion result is as follows.

b31 b0		b15 b8	b7 b0	
03AC 6251	➔	16#33	16#30	**D** + 0
		16#43	16#41	**D** + 1
		16#32	16#36	**D** + 2
		16#31	16#35	**D** + 3
		16#00	16#00	**D** + 4

Example 1

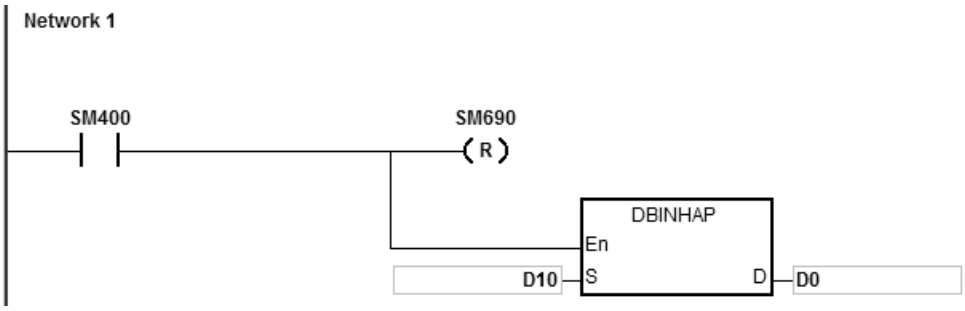
Suppose the value in D10 is 16#9C06 and SM690 is OFF. When the instruction executes, the values are D0=16#4339, D1=16#3630, D2=16#0000.



D10		b15 b8	b7 b0	
16#9C06	➔	16#43	16#39	D0
		16#36	16#30	D1
		16#00	16#00	D2

Example 2

Suppose the value in D10 is 16#7B3C581F and SM690 is OFF. When the instruction executes, the values are D0=16#4237, D1=16#4333, D2=16#3835, D3=16#4631, D4=16#0000.



D10		b15 b8	b7 b0	
16#7B3C 581F	➔	16#42	16#37	D0
		16#43	16#33	D1
		16#38	16#35	D2
		16#46	16#31	D3
		16#00	16#00	D4

Additional remarks

1. For 16-bit instructions, if **D**+2 exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
2. For 32-bit instructions, if **D**+4 exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
3. For 16-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of WORD/INT.
4. For 32-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [5] of WORD/INT.

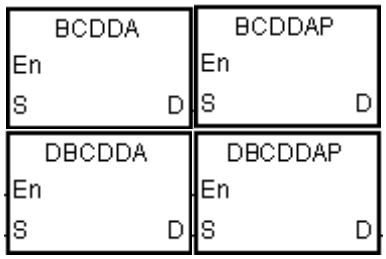
API	Instruction code			Operand	Function
2102	D	BCDDA	P	**S, D**	Converting a binary-coded decimal number into an ASCII code

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●	●	●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●		●	●	●						
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source value

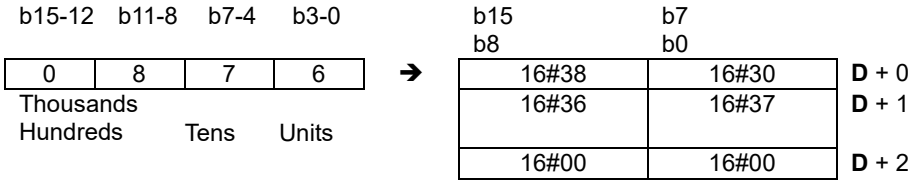
D : Device where the conversion result is stored

Explanation

1. This instruction converts a binary-coded decimal number in **S** into an ASCII code, and stores the conversion result in **D**.

2. The instruction supports SM690 to control the ending character.

3. The binary-coded decimal value in **S** used in the 16-bit instruction must be between 0–9999, and should be a four-digit binary-coded decimal value. The operand **D** occupies three word devices. The data is converted as follows.



If SM690 is OFF, the instruction stores the ending character 16#0000 in **D**+2. If SM690 is ON, the value in **D**+2 is unchanged without the ending character.

4. For 32-bit instructions, the binary-coded decimal value in **S** must be between 0–99999999, and should be an eight-digit binary-coded decimal value. The operand **D** occupies five word devices. The data is converted as follows.

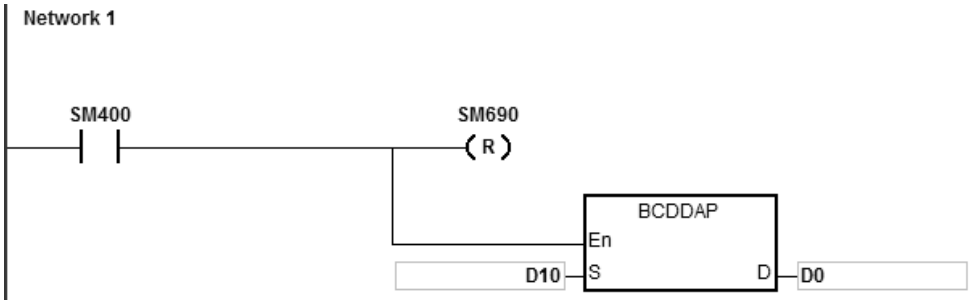
b31	b0		b15 b8	b7 b0	
0876 5432		➔	16#38	16#30	**D** + 0
			16#36	16#37	**D** + 1
			16#34	16#35	**D** + 2
			16#32	16#33	**D** + 3
			16#00	16#00	**D** + 4

5. If SM690 is OFF, the instruction stores the ending character 16#0000 in **D**+5. If SM690 is ON, the value in **D**+5 is unchanged without the ending character.

6. Even if the first digit of binary-coded decimal value in **S** is 0, the instruction converts it into an ASCII code 0 (16#30).

Example 1

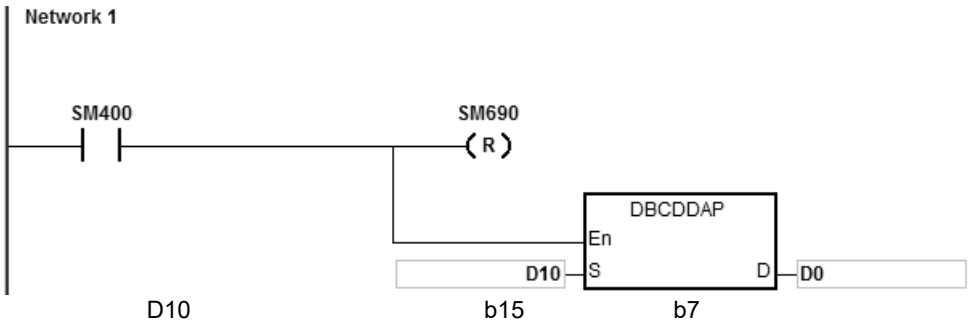
Suppose the binary-coded decimal value in D10 is 1295 and SM690 is OFF. When the instruction executes, the values are D0=16#3231, D1=16#3539, D2=16#0000.



D10	b15 b8	b7 b0	
1295	➔	16#32	16#31	D0
		16#35	16#39	D1
		16#00	16#00	D2

Example 2

Suppose the binary-coded decimal value in D10 is 00578352 and SM690 is OFF. When the instruction executes, the values are D0=16#3030, D1=16#3735, D2=16#3338, D3=16#3235, D4=16#0000.



D10	b15 b8	b7 b0	
0057 8352	➔	16#30	16#30	D0
		16#37	16#35	D1
		16#33	16#38	D2
		16#32	16#35	D3
		16#00	16#00	D4

Additional remarks

1. For 16-bit instructions, if the value in **S** is not between 0–9999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.

2. For 32-bit instructions, if the value in **S** is not between 0–99999999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D. The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.

3. For 16-bit instructions, if **D**+2 exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.

4. For 32-bit instructions, if **D**+4 exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.

5. For 16-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [3] of WORD/INT.

6. For 32-bit instructions, if you declare the operand **D** in ISPSoft, the data type is ARRAY [5] of WORD/INT.

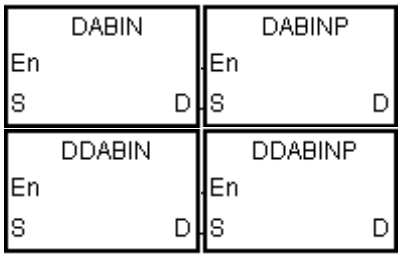
API	Instruction code			Operand	Function
2103	D	DABIN	P	**S, D**	Converting a signed decimal ASCII code into a signed decimal binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



S : Source value

D : Device where the conversion result is stored

Explanation

6

1. This instruction converts the signed decimal ASCII code in **S** into a signed decimal binary number, and stores the conversion result in **D**.

2. For 16-bit instructions, the operand **S** occupies three word devices, and the decimal ASCII code in **S** must be between -32768 to 32767. If the value in **S** is a string and does not include the ending character 16#00, the conversion can be up to 5 digits (excluding the sign).

	b15 b8	b7 b0
S + 0	" 2 "	" 1 "
S + 1	" 4 "	" 3 "
S + 2	16#00	16#00

➔ b15 b0: 1234 **D**

3. If the first character is " " (a space), the sign is a positive sign. If the first character is "-", the sign is a negative sign. Take the string "2345" for example.

	b15 b8	b7 b0
S + 0	" 2 "	" - "
S + 1	" 4 "	" 3 "
S + 2	16#00	" 5 "

➔ b15 b0: -2345 **D**

4. For 32-bit instructions, the operand **S** occupies six word devices, and the decimal ASCII code in **S** must be between -2147483648 to 2147483647. If the value in **S** is a string and does not include the ending character 16#00, the conversion can be up to 10 digits (sign excluded).

	b15 b8	b7 b0		b31 b0	
S + 0	" 2 "	" 1 "	➔	1234567890	**D**
S + 1	" 4 "	" 3 "			
S + 2	" 6 "	" 5 "			
S + 3	" 8 "	" 7 "			
S + 4	" 0 "	" 9 "			
S + 5	16#00	16#00			

5. If the first character is " " (a space), the sign is a positive sign. If the first character is "-", the sign is a negative sign. Take the string "-234567890" for example.

	b15 b8	b7 b0		b31 b0	
S + 0	' 2 '	' - '	➔	-234567890	**D**
S + 1	' 4 '	' 3 '			
S + 2	' 6 '	' 5 '			
S + 3	' 8 '	' 7 '			
S + 4	' 0 '	' 9 '			
S + 5	16#00	16#00			

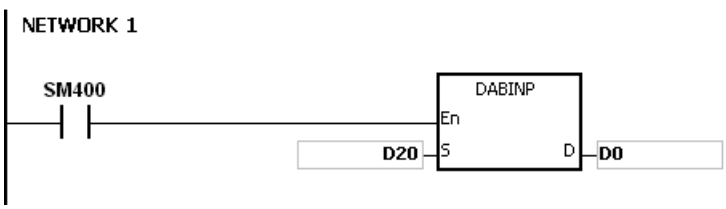
6. If the first digit of the string in the device **S** is blank (16#20) + sign (16#2B), the value in **S** is processed as 0. For the second digit, if the number is not 0–9, the instruction treats it as the end of a string and no error message will be shown. For example if the word order is 16#20→16#31→16#32→16#2B, the conversion result is 12.

7. For 16-bit instructions, the string range in the device **S** is 1–6 (positive or negative signs included) and for 32-bit instruction the range is 1–11 (positive negative signs included).

8. Only the 32-bit instructions can use the 32-bit counter, but not the device E.

Example 1

Suppose the values are D20=16#302D, D21=16#3433, D22=16#0035. When the instruction executes, the value is D0=-345.



	b15 b8	b7 b0
D20	16#30	16#2D
D21	16#34	16#33
D22	16#00	16#35

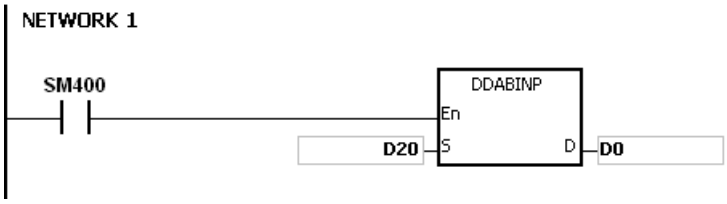
➔

b15 b0	
-345	D0

Example 2

Suppose the values are D20=16#322D, D21=16#3433, D22=16#3635, D23=16#2037, D24=16#0000, D25=16#0000.

When the instruction executes, the value is D0=-234567.



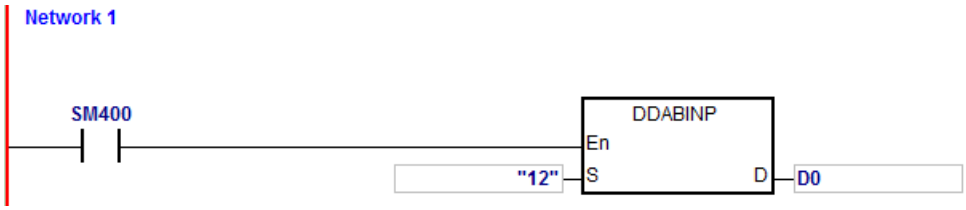
	b15 b8	b7 b0
D20	16#32	16#2D
D21	16#34	16#33
D22	16#36	16#35
D23	16#20	16#37
D24	16#00	16#00
D25	16#00	16#00

➔

b31 b0	
-234567	D0

Example 3

Suppose the string value in **S** is 12. When the instruction executes, the value is D0=12.



Additional remarks

1. If the value of the first word in **S** is an ending character (16#00), the value is processed as 0 (16#30).
2. If the value of the first digit in **S** is 16#20 (blank) or 16#2B (+) or 16#2D (-) and the second digit is 16#00, the value in **S** is processed as 0 (16#30).
3. Even if the first digit of binary-coded decimal value in **S** is 0, it is converted into the ASCII code 0 (16#30).
4. The value of the first digit in **S** only supports ASCII codes, 16#30–16#39 (0–9), 16#200 (blank), 16#2D (negative sign), 16#2B (positive sign), 16#00 (ending character). If the value of the first digit in **S** is not ASCII code, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. Except the first digit, if the value of other digits in **S** are not ASCII codes, 16#30–16#39 or 16#00, the value in **S** is processed as 16#00.
6. If the value in **S** exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003. The instruction is not executed.
7. For 16-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [3] of WORD/INT.
8. For 32-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [6] of WORD/INT.

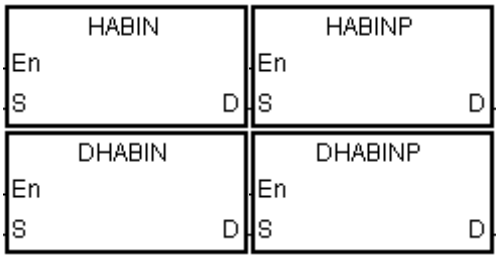
API	Instruction code			Operand	Function
2104	D	HABIN	P	**S, D**	Converting a hexadecimal ASCII code into a hexadecimal binary number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S					●	●		●	●						○	
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



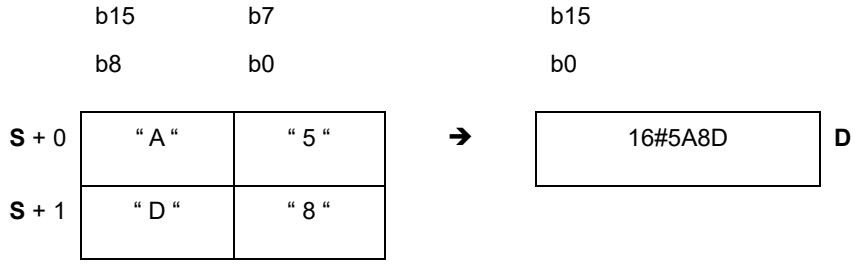
S : Source value

D : Device where the conversion result is stored

Explanation

1. This instruction converts a hexadecimal ASCII code in **S** into a hexadecimal binary number, and stores the conversion result in **D**.

2. For 16-bit instructions, the operand **S** occupies two word devices. If the value in **S** is a string and does not include the ending character 16#00, the conversion can be up to 4 digits (sign excluded). The hexadecimal ASCII code in **S** must be between 0000–FFFF. If **S** is a string, the string must be between “0”–“FFFF”.



3. For 32-bit instructions, the operand **S** occupies four word devices. If the value in **S** is a string and does not include the ending character 16#00, the conversion can be up to 8 digits (sign excluded). The hexadecimal ASCII code in **S** must be between 00000000–FFFFFFFF. If **S** is a string, the string must be between “0”–“FFFFFFFF”.

	b15 ~ b8	b7 ~ b0
S + 0	" C "	" 5 "
S + 1	" 8 "	" B "
S + 2	" 7 "	" 0 "
S + 3	" 1 "	" E "

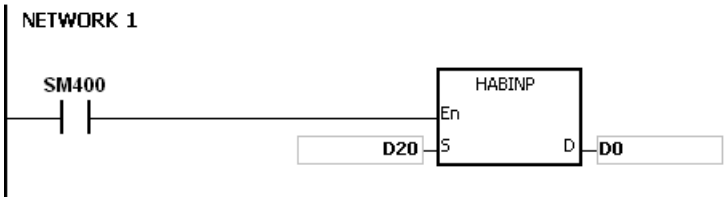
➔

b31 ~ b0	
16#5CB8 07E1	**D**

4. The valid string range in the device **S** for 16-bit instruction is 1–4 and for 32-bit instruction it is 1–8.

Example 1

Suppose the values are D20=16#3641, D21=16#4633 (ASCII 16#A63F). When the instruction executes, the value is D0=-22977.



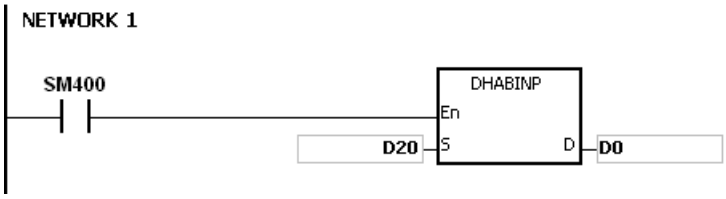
	b15 ~ b8	b7 ~ b0
D20	16#36	16#41
D21	16#46	16#33

➔

b15 ~ b0	
16#A63F	D0

Example 2

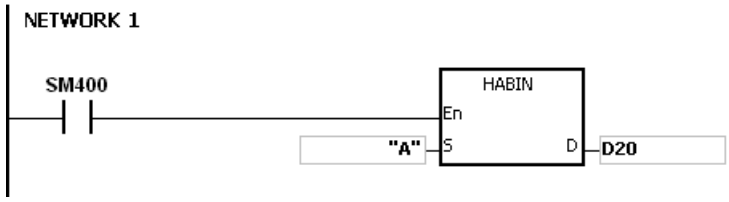
Suppose the values are D20=16#4634, D21=16#3244, D22=16#3738, D23=16#3035 (ASCII 16#4FD28750). When the instruction executes, the value is (D1, D0)=16#4FD28750.



	b15 b8	b7 b0		b31 b0	
D20	16#46	16#34	➔	16#4FD2 8750	D0
D21	16#32	16#44			
D22	16#37	16#38			
D23	16#30	16#35			

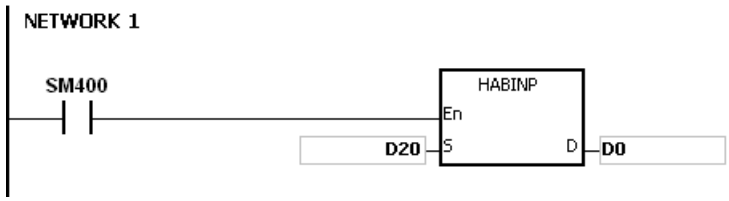
Example 3

Suppose the string value in **S** is A. When the instruction executes, the value is D20=16#A=10.



Example 4

Suppose the values are D20=16#3641, D21=16#4600 (ASCII 16#00A6). When the instruction executes, the value is D0=166.



	b15 b8	b7 b0		b15 b0	
D20	16#36	16#41	➔	16#00A6	D0
D21	16#46	16#00			

Additional remarks

1. If the ASCII code in **S** is not between 16#30–16#39 (“0”–”9”), or between 16#41–16#46 (“A”–”F”), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. For 16-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

3. For 32-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [4] of WORD/INT.

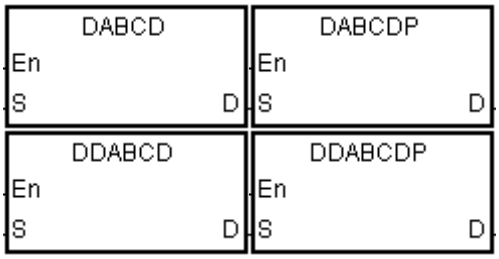
API	Instruction code			Operand	Function
2105	D	DABCD	P	**S, D**	Converting an ASCII code into a binary-coded decimal number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
D					●	●	●	●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●	●		●	●	●						

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	ES3

Symbol



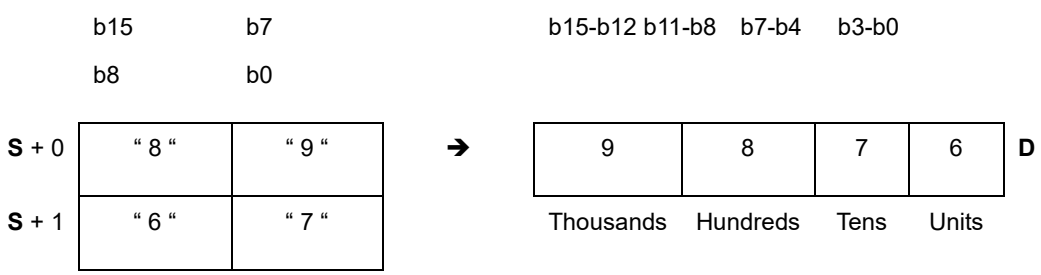
S : Source value

D : Device where the conversion result is stored

Explanation

1. This instruction converts the ASCII code in **S** into a binary-coded decimal number, and stores the conversion result in **D**.

2. For 16-bit instructions, the operand **S** occupies two word devices, and the ASCII code in **S** must be between 0000–9999. If **S** is a string, the string must be between "0"–"9999".



3. For 32-bit instructions, the operand **S** occupies four word devices, and the ASCII code in **S** must be between 0000000–99999999. If **S** is a string, the string must be between "0"–"99999999".

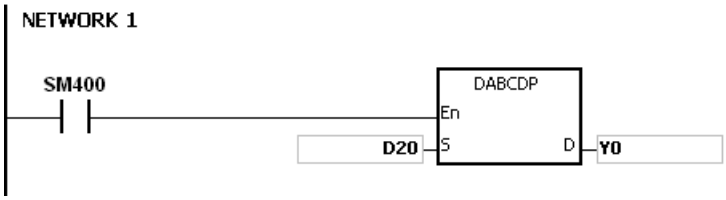
	b15 b8	b7 b0		b15-b12 b11-b8 b3-b0			b7-b4	
S + 0	" 8 "	" 9 "	➔	5	4	3	2	**D**
S + 1	" 6 "	" 7 "		9	8	7	6	**D**+1
S + 2	" 4 "	" 5 "						
S + 3	" 2 "	" 3 "						

4. If the value in **S** is 16#20 the value is processed as 16#30. If the value in **S** is 16#00, the value is processed as an ending character.

5. For 16-bit instructions, if **S** is a string, the number of characters contained in the string must be between 1–4. For 32-bit instructions, if **S** is a string, the number of characters contained in the string must be between 1–8.

Example 1

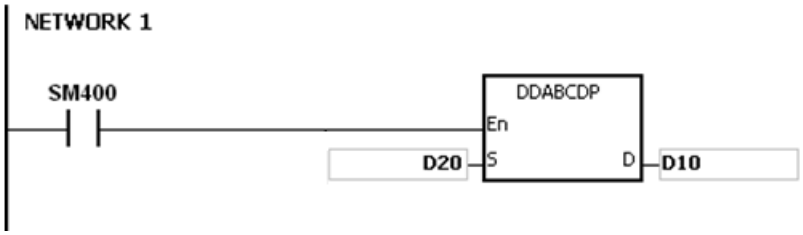
Suppose the values are D20=16#3820, D21=16#3637 (ASCII 876). When the instruction executes, the instruction converts the value into Y0=16#876.



	b15 b8	b7 b0		b15-b12 b11-b8 b3-b0			b7-b4	
D20	16#38	16#20	➔	0	8	7	6	Y0
D21	16#36	16#37						

Example 2

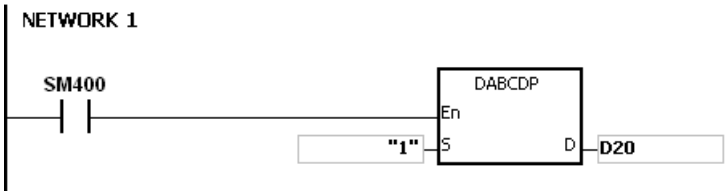
Suppose the values are D20=16#3738, D21=16#3536, D22=16#3334, D23=16#3132 (ASCII 87654321). When the instruction executes, the value is (D11, D10)= 16#87654321.



	b15 b8	b7 b0		b31 b0	
D20	16#37	16#38	➔	8765 4321	D0
D21	16#35	16#36			
D22	16#33	16#34			
D23	16#31	16#32			

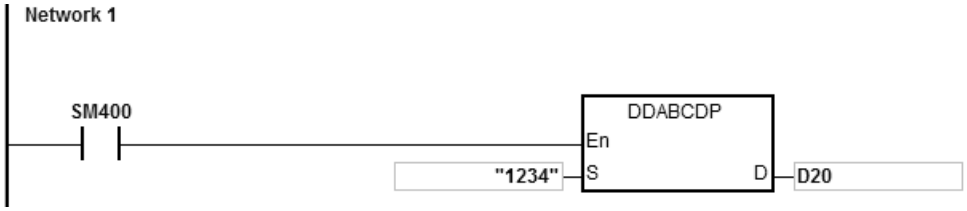
Example 3

Suppose the string value in **S** is 1. When the instruction executes, the value is D20=16#0001.



Example 4

Suppose the string value in **S** is 1234. When the instruction executes, the value is (D21, D20)= 16#00001234.



Additional remarks

1. If the ASCII code in **S** is not ASCII codes 16#30–16#39, 16#20, or 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If **S** is a string and the number of characters contained in the string exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. For 16-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

4. For 32-bit instructions, if you declare the operand **S** in ISPSoft, the data type is ARRAY [4] of WORD/INT.

API	Instruction code			Operand	Function
2106		$LEN	P	**S, D**	Calculating the length of a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

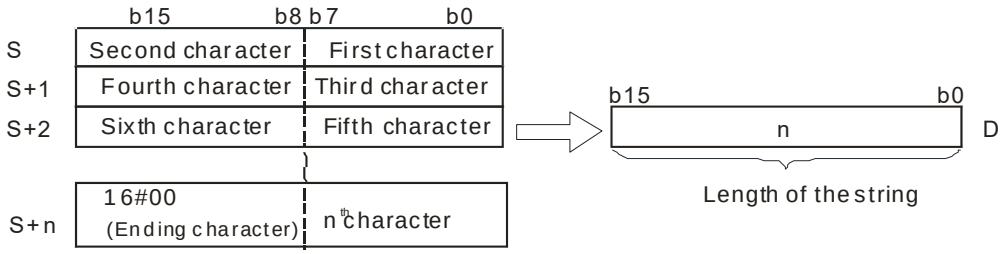
S : String

D : Length of the string

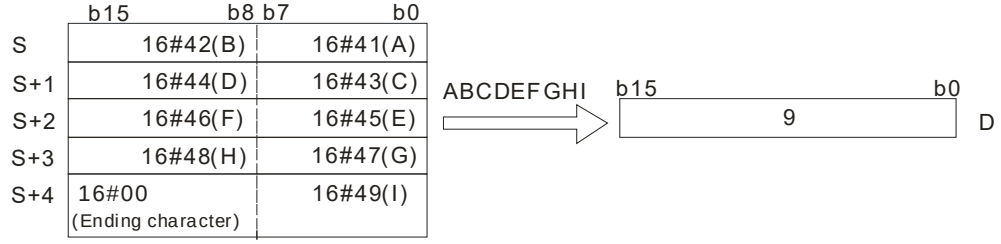
Explanation

1. This instruction calculates the length of the string in **S**, exclusive of the ending character (16#00), and stores the length of the string in **D**.

2. The value stored in **D** must be between 0–32767. If it exceeds this range, the value in **D** is processed as 32767.

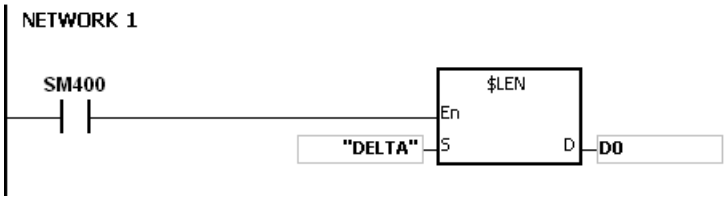


If the data in **S**-**S**+4 is ABCDEFGHI, the calculation result is as follows.



Example 1

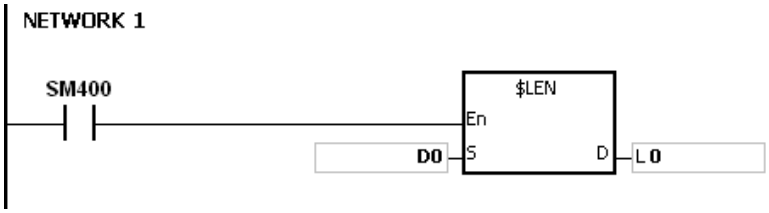
Suppose **S** is the string “DELTA”. When the instruction executes, the value in D0 is 5.



Example 2

Suppose the data in D0–D2 is as follows. When the instruction executes, the value in L0 is 5.

D0	16#45 (E)	16#44 (D)
D1	16#54 (T)	16#4C (L)
D2	16#00 (Ending character)	16#41 (A)



Additional remarks

1. If the string does not end with 16#00, the instruction executes to the maximum length of 32767.

2. If the length of the value exceeds the device range, the last character is processed as the ending character.

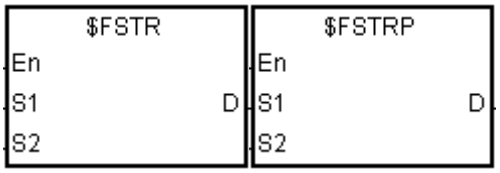
API	Instruction code			Operand	Function
2109		$FSTR	P	**S_1, S_2, D**	Converting a floating-point number into a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●							○
S_2					●	●		●	●							
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1									●				
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Source value

S_2 : First device where the format is stored

D : First device where the conversion result is stored

6

Explanation

1. This instruction converts the floating-point number in **S_1** into a string in accordance with the format setting value in **S_2**, and stores the conversion result in **D**.

2. This instruction converts the floating-point number in **S_1** into a string and appends the ending character 16#00 to the end of the string, and then stores the conversion result in **D**.

(**S_1**+1 · **S_1**)

⇨

Floating-point number

	High byte	Low byte
D	Second ASCII code	First ASCII code
D + 1	Fourth ASCII code	Third ASCII code
D + 2	Sixth ASCII code	Fifth ASCII code
……	……	
D +n-1	m^{th} ASCII code	$m-1^{th}$ ASCII code
D +n		16#00 ending character

Floating-point string

3. The conversion result varies with the setting of **S_2**.

4. The value in **S_2**+1 must be between 2≦**S_2**+1≦20; the sign (+,-) and the digits in an integer or decimal number are included but the decimal point in a decimal number and the exponent number are not included.

Operand	Description
S_2	0: Decimal format 1: Exponential
S_2+1	Number of character

5. If the value in **S_1** is a positive number, the first ASCII code can be stored in **D**; if the value in **S_1** is a negative number, the sign 16#2D (-) is stored first and then the second ASCII is stored.

6. Decimal format (**S_2**=0)

After conversion, the floating-point string in the device D.

- The value in **S_2**+1 must be between 2≦S2+1≦20; the sign (+,-) and the digits in an integer or decimal number are included but the decimal point in a decimal number and the exponent number are not included.

- Example 1

  Suppose the number of characters is eight. Two floating-point numbers examples are -1.2345678 and 123456. The calculation is as follows.

D number	Floating-point number -1.2345678		Floating-point number 123456	
	High byte	Low byte	High byte	Low byte
D	16#31 (1)	16#2D (-)	16#32 (2)	16#31 (1)
D + 1	16#32 (2)	16#2E (.)	16#34 (4)	16#33 (3)
D + 2	16#34 (4)	16#33 (3)	16#36 (6)	16#35 (5)
D + 3	16#36 (6)	16#35 (5)	16#00 ending character	
D + 4	16#00 ending character	16#38 (8)		

After the conversion, if the floating-point number is stored in **S_2**+1 but if the length exceeds the value in **S_2**+1, the instruction rounds off the digits in the decimal number. The floating-point string does not have to fill all of **S_2**+1.

- Example 2

After the conversion, if the floating-point number is stored in **S₂**+1, the instruction uses the exponential format to convert. For example, the number of characters is five digits and the floating-point number is 1234567, the conversion result is1.2346E+06.

D number	Floating-point number 1234567	
D	16#2E (.)	16#31 (1)
D + 1	16#33 (3)	16#32 (2)
D + 2	16#36 (6)	16#34 (4)
D + 3	16#2B (+)	16#45 (E)
D + 4	16#32 (6)	16#30 (0)
D + 5	16#0000 ending character	

- Example 3

After the conversion, if the floating-point number is stored in **S₂**+1, the instruction uses the exponential format to convert. For example, the number of characters is two digits and the floating-point number is 0.00012345, the conversion result is1.2E-04.

D number	Floating-point number 0.00012345	
D	16#2E (.)	16#31 (1)
D + 1	16#45 (E)	16#32 (2)
D + 2	16#30 (0)	16#2D (-)
D + 3	16#00 ending character	16#34 (4)

- Example 4

After the conversion, if the absolute value of the floating-point number is <=10^{-5}, the instruction uses the exponential format to convert. For example, the number of characters is four digits and the floating-point number is 0.00001234, the conversion result is1.234E-05.

D number	Floating-point number 0.00001234	
D	16#2E (.)	16#31 (1)
D + 1	16#33 (3)	16#32 (2)
D + 2	16#45 (E)	16#34 (4)
D + 3	16#30 (0)	16#2D (-)
D + 4	16#00 ending character	16#35 (5)

7. Exponential format (**S_2**=1)

After conversion, the floating-point string in the device D.

- The value in **S_2**+1 must be between 2≦S2+1≦20; the sign (+,-) and the digits in an integer and decimal number are included but the decimal point in the decimal number and the exponent number are not included. After calculation, the length adds the exponents (four digits) and the decimal point of the decimal number.
- The number of character in the area for the integer is one digit.
- The number of character in the area for the exponent is four characters.
  If the exponent is a positive number, the instruction adds 16#2B (+) in the area for exponent in **D**. If the exponent is a negative number, the instruction adds 16#2D (-) in the area for exponent in **D**. The number of character in the area for the exponent is two digits. If there is only one digit in the conversion result, the instruction adds 16#30 (0) as the first digit of the area for the exponent.
- Example

  Suppose the number of characters is eight. Two floating-point numbers examples are -123.456789 and 123456. The calculation is as follows.

D number	Floating-point number -123.456789		Floating-point number 12345	
	High byte	Low byte	High byte	Low byte
D	16#31 (1)	16#2D (-)	16#2E (.)	16#31 (1)
D + 1	16#32 (2)	16#2E (.)	16#33 (3)	16#32 (2)
D + 2	16#34 (4)	16#33 (3)	16#35 (5)	16#34 (4)

D + 3	16#36 (6)	16#35 (5)	16#2B (+)	16#45 (E)
D + 4	16#45 (E)	16#38 (8)	16#34 (4)	16#30 (0)
D + 5	16#30 (0)	16#2B (+)	16#00 ending character	
D + 6	16#00 ending character	16#32 (2)		

After the conversion, if the floating-point number is stored in $\mathbf{S_2}$+1, and the instruction rounds off the extra digits.

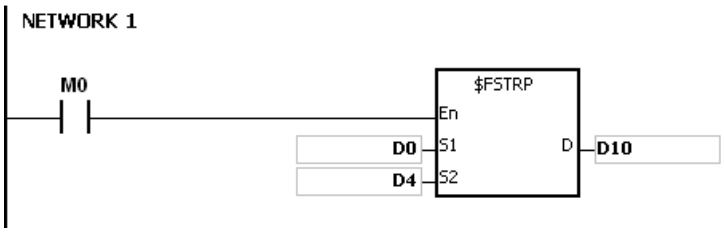
The conversion result of the string length for the floating number -123.456789 in eight characters is 13 (the ending character excluded).

The conversion result of the string length for the floating number 12345 in eight characters is 10 (the ending character excluded).

The floating-point string does not have to fill all the characters.

Example 1

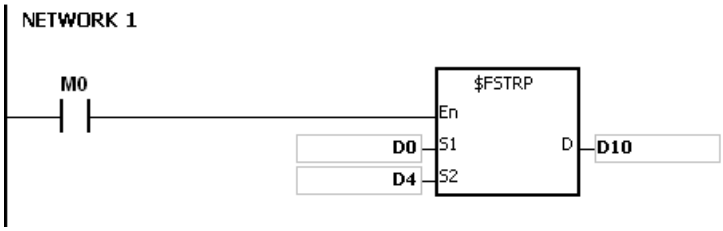
The floating-point number in (D1, D0= 12.3456) is converted into the decimal format in a string (D4=0, D5=8).



D10	16#32 (2)	16#31 (1)
D11	16#33 (3)	16#2E (.)
D12	16#35 (5)	16#34 (4)
D13	16#00 ending character	16#36 (6)

Example 2

The floating-point number in (D1, D0 = 0.0012345678) is converted into the exponential format in a string (D4=1, D5=8).



D10	16#2E (.)	16#31 (1)
D11	16#33 (3)	16#30 (2)
D12	16#35 (5)	16#34 (4)
D13	16#37 (7)	16#36 (6)
D14	16#45 (E)	16#38 (8)
D15	16#30 (0)	16#2D (-)
D16	16#00 ending character	16#33 (3)

Additional remarks

1. If the value in $\mathbf{S_1}$ exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.
2. If the value in $\mathbf{S_2}$ is neither 0 nor 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the value in $\mathbf{S_2}$+1 is not in between $2 \leqq \mathbf{S_2}+1 \leqq 20$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If you declare the operand $\mathbf{S_2}$ in ISPSoft, the data type is ARRAY [2] of WORD/INT.

API	Instruction code			Operand	Function
2110		$FVAL	P	**S, D**	Converting a string into a floating-point number

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D									●				

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : Source value

D : Device where the conversion result is stored

Explanation

1. This instruction converts a string in **S** into a floating-point number, and stores the conversion result in **D**.

	High byte	Low byte
S	Second ASCII code	First ASCII code
S + 1	Fourth ASCII code	Third ASCII code
S + 2	Sixth ASCII code	Fifth ASCII code
……	……	
S +n-1	mth ASCII code	m-1th ASCII code
S +n		16#00 ending character

Floating-point string

⇨ (**D+1** , **D**)

Floating-point number

- Refer to the following sections for the ASCII code usage.
- For decimal or exponential formats, the maximum length for the floating-point string (m) is 24 characters (excluding the ending character 16#00) and the maximum length of **n** is 13 characters.

2. The format of the value in **S** can be decimal or exponential. The instruction automatically determines the format according to the value in **S**.
   - Decimal format: the length for the floating-point string is nine; the ending character 16#00 is not included.

	High byte	Low byte		
S	16#31 (1)	16#32 (2)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#34 (4)	16#33 (3)	⇨	21.234567
S + 3	16#36 (6)	16#35 (5)		
S + 4	16#00 ending character	16#37 (7)		

OR

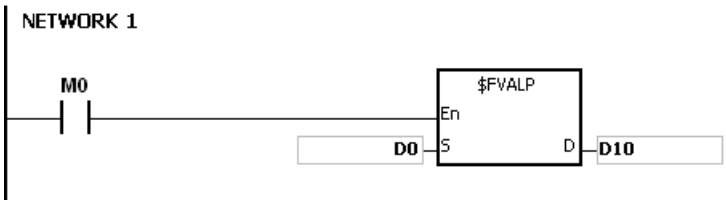
	High byte	Low byte		
S	16#31 (1)	16#2D (-)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#34 (4)	16#33 (3)	⇨	-1.234568
S + 3	16#36 (6)	16#35 (5)		
S + 4	16#00 ending character	16#38 (8)		

- Exponential format: the length for the floating-point string is 9; the ending character 16#00 is not included.

	High byte	Low byte		
S	16#31 (1)	16#2D (-)		
S + 1	16#32 (2)	16#2E (.)		**D**
S + 2	16#45 (E)	16#33 (3)	⇨	-1230
S + 3	16#30 (0)	16#2B (+)		
S + 4	16#00 ending character	16#32 (3)		

3. If the sign code in **S** is 16#20, 16#30, or 16#2B, then the conversion result is a positive value. If the sign code in $\mathbf{S_1}$ is 16#2D, then the conversion result is a negative value.

Example 1

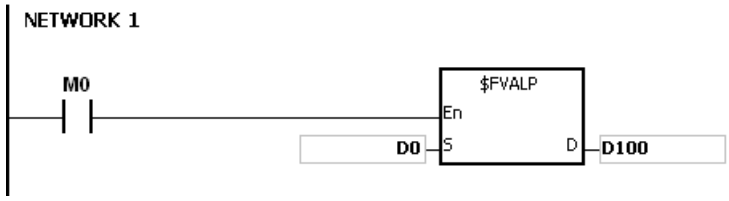


D0	16#31 (1)	16#32 (2)
D1	16#32 (2)	16#2E (.)
D2	16#34 (4)	16#33 (3)
D3	16#36 (6)	16#35 (5)
D4	16#00 ending character	16#37 (7)

(D11 · D10)

⇨ 21.234567

Example 2



D0	16#31 (1)	16#2D (-)
D1	16#32 (2)	16#2E (.)
D2	16#45 (E)	16#33 (3)
D3	16#30 (0)	16#2B (+)
D4	16#00 ending character	16#32 (3)

(D101 · D100)

⇨ -1230

Additional remarks

1. If the length of the string in **S** exceeds 25 bytes and does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#2003.

2. There are some rules for the value in **S**. if the value in **S** does not follow the rules, the instruction is not executed, SM0 is ON and the error code in SR0 is 16#2003.
   The first ASCII code: signs16#2B(+), 16#2D(-), blank16#20, and numbers 16#30(0)–16#39(9) are allowed.
   If the first ASCII code is a sign or a blank, the second ASCII code must be a number.
   The second ASCII code can be in either decimal or exponential format.

- Decimal format:

The decimal point "," (16#2E), can only occur once and there must be numbers before and after the decimal point.

- Exponential format:

The decimal point "," (16#2E), can only occur once and there must be numbers before and after the decimal point.

There must be a number before the exponent.

Integers: only numbers "0" (16#30)–"9" (16#39) are allowed.

Decimals: only numbers "0" (16#30)–"9" (16#39) are allowed.

Exponents: The format for a four digit ASCII code is as below.

There must be an "E" (16#45) or "e" (16#65) and it can only occur once.

There must be a sign "+"(16#2B) or "-" (16#2D) and it can only occur once.

There must be two digits; numbers "0" (16#30)–"9"(16#39).

3. If the string in **S** is out of range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.

- If the first character in the string is a number "0"–"9" (16#30–16#39), the valid range for a floating-point string is 1–24. The minimum length for the string is one.
- If the first character in the string is a blank (16#20) or a sign ("+"(16#2B) or "-" (16#2D)), the valid range for a floating-point string is 2–24. The minimum length for the string is two ("+1").

4. If the conversion result exceeds the range of values that can be represented by floating-point numbers, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

API	Instruction code			Operand	Function
2111		$RIGHT	P	**S, n, D**	Retrieving characters from a string beginning from the right.

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
n					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : String

n : Number of characters to be retrieved

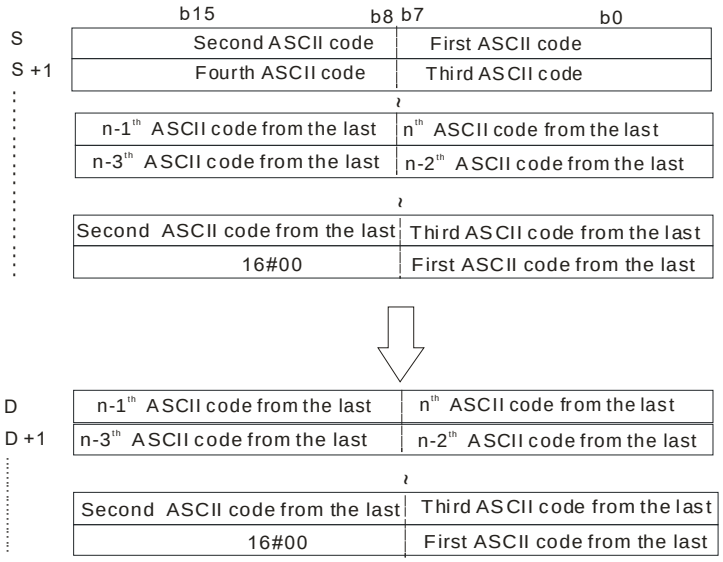
D : Device where the characters retrieved are stored

Explanation

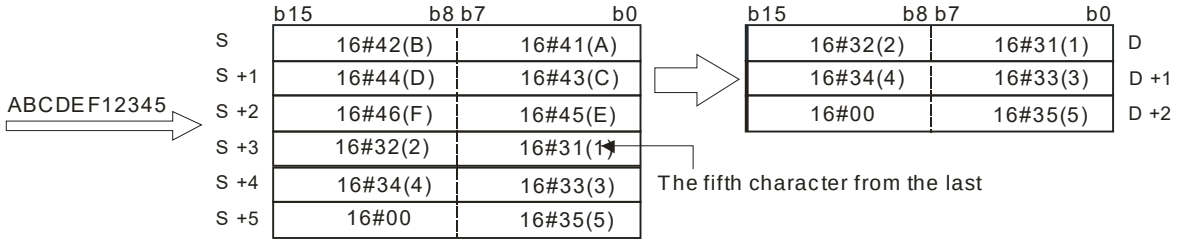
6

1. This instruction retrieves **n** characters from the string in **S** from the right, and stores the retrieved characters in **D**. When the data type of **S** (source value) is string, the maximum length for the value in **S** is 31 characters. When the data type of **S** (source value) is not string, the maximum length for the value in **S** is 255 characters.

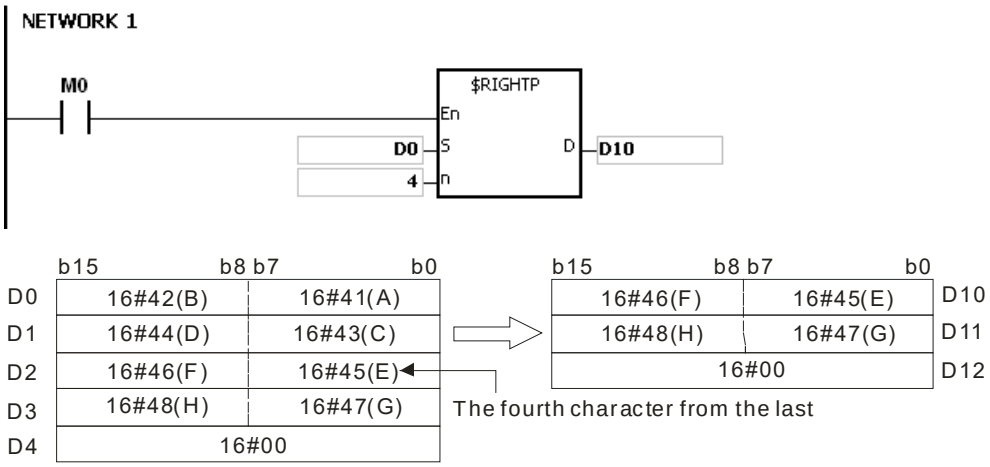
2. If **n** is 0, the value in **D** is 0. The maximum length for n is 255 characters. .



If the data in **S** is ABCDEF12345 and **n** is 5, the instruction retrieves five characters in the string in **S** from the right. The conversion result is as follows.



Example



Additional remarks

1. If the operand **S** is not a string ($) but a device with a string, the string in **S** can be up to 256 characters (16#00 ending character included). If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If **n** is less than 0, it is processed as 0. If **n** is greater than the length of the string in **S**, it is processed as the length of the string is **S**.

3. If **D** is not sufficient to contain **n** characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2112		$LEFT	P	**S, n, D**	Retrieving characters from a string beginning from the left.

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●	●						○	
n					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : String

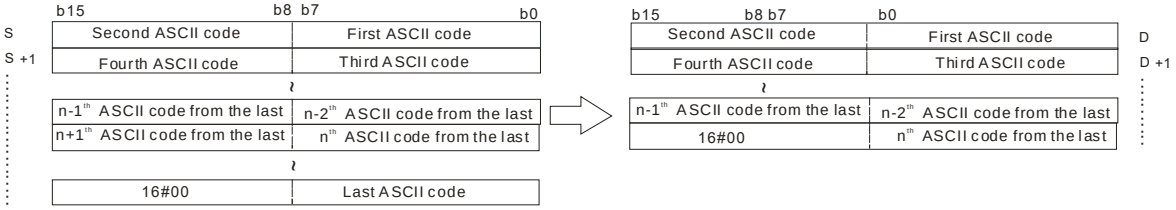
n : Number of characters to retrieve

D : Device where the characters retrieved are stored

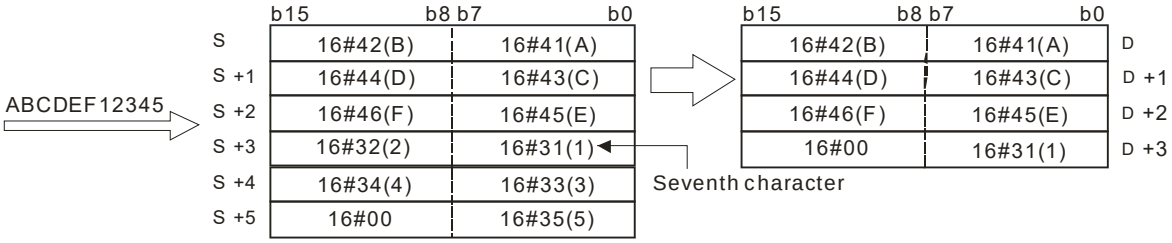
Explanation

1. This instruction retrieves **n** characters from the string in **S** starting from the left, and stores the retrieved characters in **D**. When **S** is a string device, the maximum length for the value in **S** is 31 characters; when the **S** is not a string device, the maximum length for the value in **S** is 255 characters.

2. If **n** is 0, the value in **D** is 0.

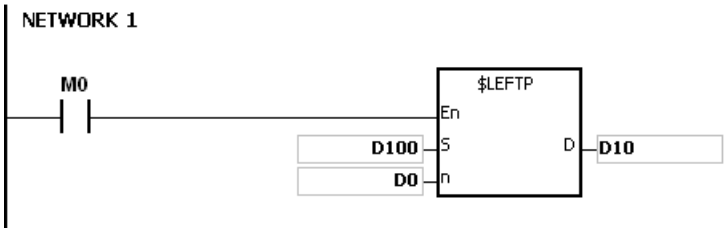


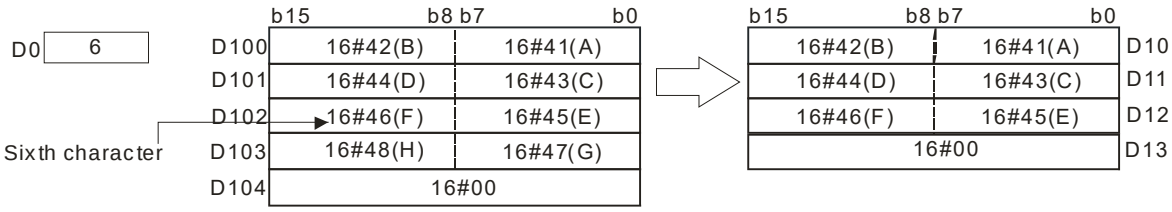
If the data in **S** is ABCDEF12345 and **n** is 7, the instruction retrieves seven characters in the string in **S** from the left. The conversion result is as follows.



Example

When M0 is ON, the instruction retrieves the six characters starting from the character in D100, and stores them in D10.





Additional remarks

1. If the operand **S** is not a string ($) but a device with a string, the string in **S** can be up to 256 characters(16#00 ending character included). If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If **n** is less than 0, it is processed as 0. If **n** is greater than the length of the string in **S**, it is processed as the length of the string is **S**.

3. If **D** is not sufficient to contain **n** characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2113		$MIDR	P	**S_1, S_2, D**	Retrieving a part of a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●		○					
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S_1 : String

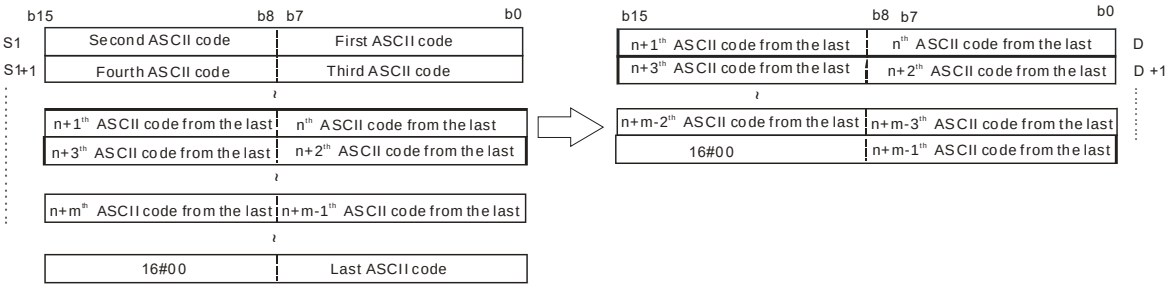
S_2 : Part of the string to be retrieved

D : Device where the characters retrieved are stored

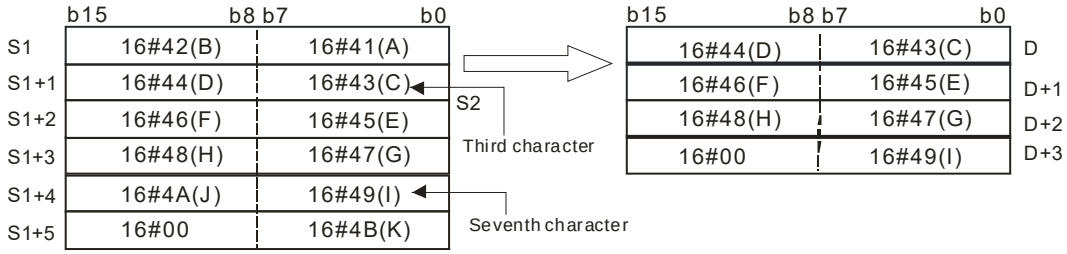
Explanation

6

1. Suppose the values in **S_2** and **S_2**+1 are n and m respectively. This instruction retrieves m characters starting from the n^{th} character in the string in **S_1**, and stores them in **D**.



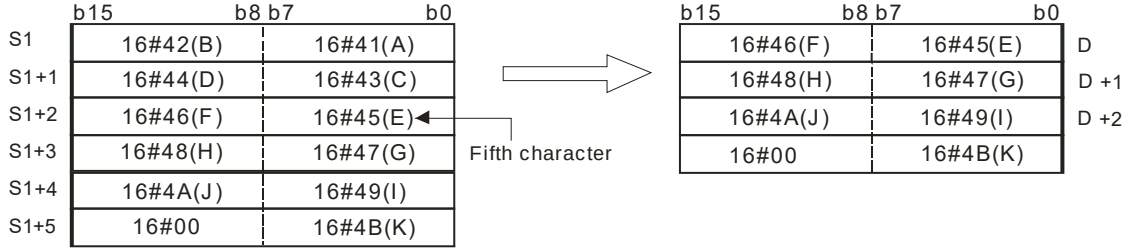
2. If the data in **S_1** is ABCDEFGHIJK, the value in **S_2** is 3, and the value in **S_2**+1 is 7, the instruction retrieves the seven characters starting from the third characters in the string from the left. The conversion result is as follows.



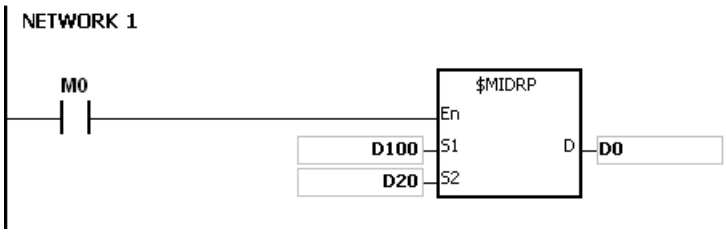
3. If the value of S_2 is S_2<=0, S_2+1<-1or S_2+1=0, the instruction is not executed.

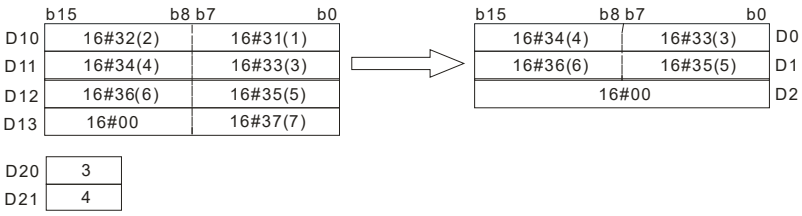
4. If the value in S_2+1 is -1, the instruction retrieves the characters in S_1 starting from the character indicated by the value in S_2 to the last character in S_1.

5. If the data in S_1 is ABCDEFGHIJK, the value in S_2 is 5, and the value in S_2+1 is -1, the conversion result is as follows.



Example





D20	3
D21	4

Additional remarks

1. If the operand **S_1** is not a string ($) but a device with a string, the string in **S_1** can be up to 256 characters (16#00 ending character included). If the string in **S_1** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the value of **S_2** is **S_2**<=0 or **S_2**+1<-1, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in **S_2** or **S_2** + **S_2**+1 is larger than the length of the string in **S_1**, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in **S_2**+1 is larger than the number of characters that can be retrieved from the string in **S_1**, SM0 is ON, and the error code in SR0 is 16#2003.

5. If **D** is not sufficient to contain **S_2**+1 characters, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6. if you declare the operand **S_2** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

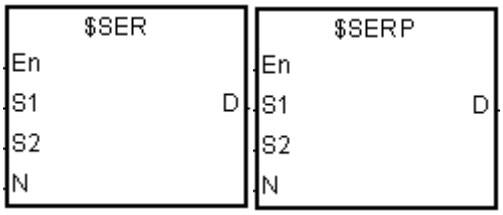
API	Instruction code		Operand	Function
2115	\$SER	P	**S_1, S_2, n, D**	Searching a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	
n					●	●		●	●		○	○	○	○		
D					●	●		●			○	○				

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							●
n		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String to be searched

S_2 : String to be searched for

n : **n**th character in **S_2** where the search begins

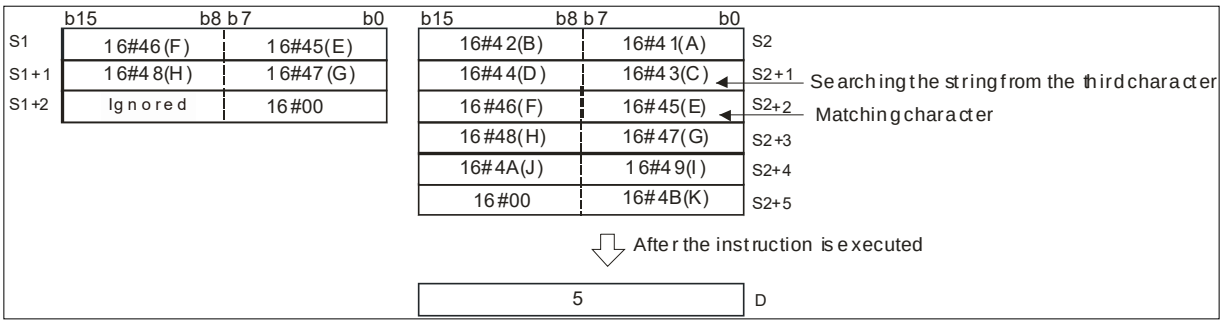
D : Search result

Explanation

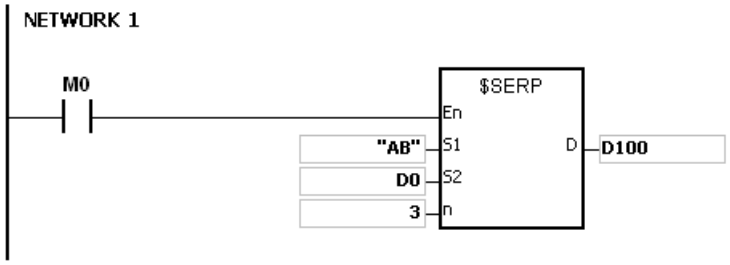
1. This instruction searches the string in **S_2** from the **n**th character for the string in **S_1**, and stores the search result in **D**.

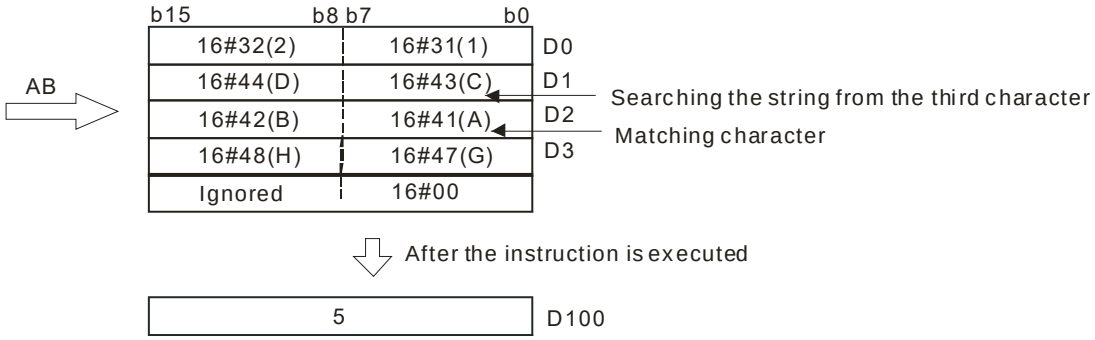
2. When the **n**th character exceeds the length of the string in **S_2**, or **S_1>S_2**, **D** is 0.

3. Suppose the string in **S_2** is "ABCDEFGHIJK", the string in **S_1** is "EFGH", and **n** is 3. The search begins from the third character in **S_2**, and the value in **D** is 5.



Example





Additional remarks

1. If the operand S_1 is not a string ($) but a device with a string, the string in S_1 can be up to 256 characters(16#00 ending character included). If the string in S_1 does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the operand S_1 is not a string ($) but a device with a string, the string in S_2 can be up to 256 characters(16#00 ending character included). If the string in S_1 does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

3. If **n** is less than or equal to 0, SM0 is ON, and the error code in SR0 is 16#2003.

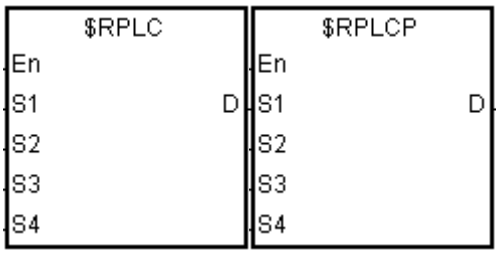
API	Instruction code			Operand	Function
2116		$RPLC	P	**S_1, S_2, S_3, S_4, D**	Replacing the characters in a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	
S_3					●	●		●	●		○	○	○	○		
S_4					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							●
S_3		●			●	●							
S_4		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String to be searched and matching characters replaced

S_2 : New string

S_3 : Number of characters in **S_1** to be replaced

S_4 : The string **S_2** replaces the characters in **S_1** starting from the character indicated by the value in **S_4**.

D : Device where the result is stored

Explanation

1. This instruction replaces the characters in **S_1** with the characters in **S_2**, starting from the character in **S_1** that matches the character in **S_4**. The number of characters that are replaced is indicated by the value in **S_3**, and the instruction stores the result in **D**.

2. For example, the instruction replaces the four characters starting from the sixth character in the string "12345**6789**0" with "MN", and the result is "12345**MN**0".

	b15 b8	b7 b0
S1	16#32(2)	16#31(1)
S1+1	16#34(4)	16#33(3)
S1+2	16#36(6)	16#35(5)
S1+3	16#38(8)	16#37(7)
S1+4	16#30(0)	16#39(9)
S1+5	Ignored	16#00

After the instruction is executed

b15	b8 b7	b0	
16#32(2)	16#31(1)		D
16#34(4)	16#33(3)		D+1
16#4D(M)	16#35(5)		D+2
16#30(0)	16#4E(N)		D+3
16#0000			D+4

S2	16#4E(N)	16#4D(M)
S2+1	Ignored	16#00

S3	4	Number of characters which are replaced
S4	6	Character which is replaced

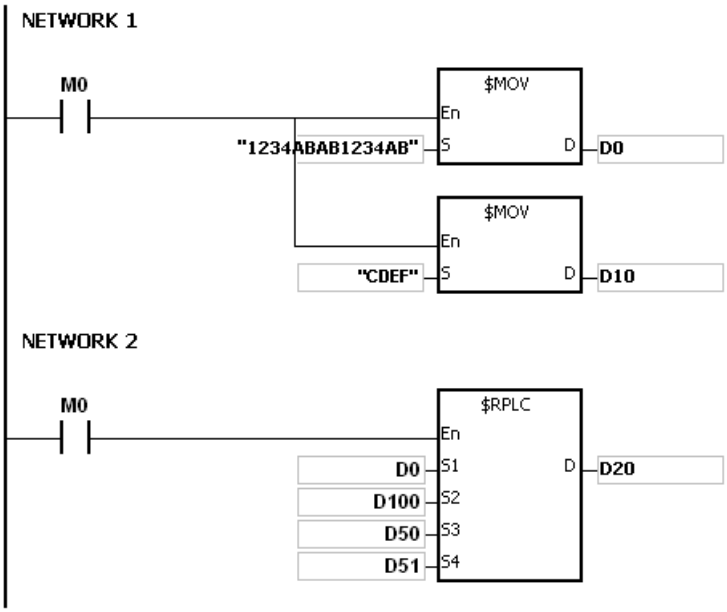
3. If the string in **S_2** is 16#00, the instruction deletes the matching characters.

4. If the value in **S_3** is larger than the number of characters that the instruction can replace in the string in **S_1**, the instruction replaces the characters in **S_1** starting from the character indicated by the value in **S_4** to the last character in **S_1**.

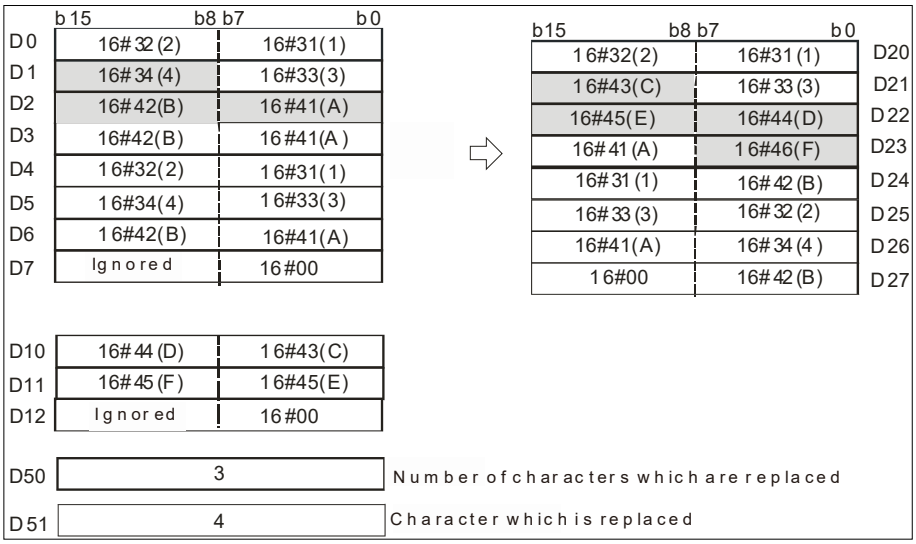
5. If the value in **S_3** is equal to 0, the instruction is not executed.

Example

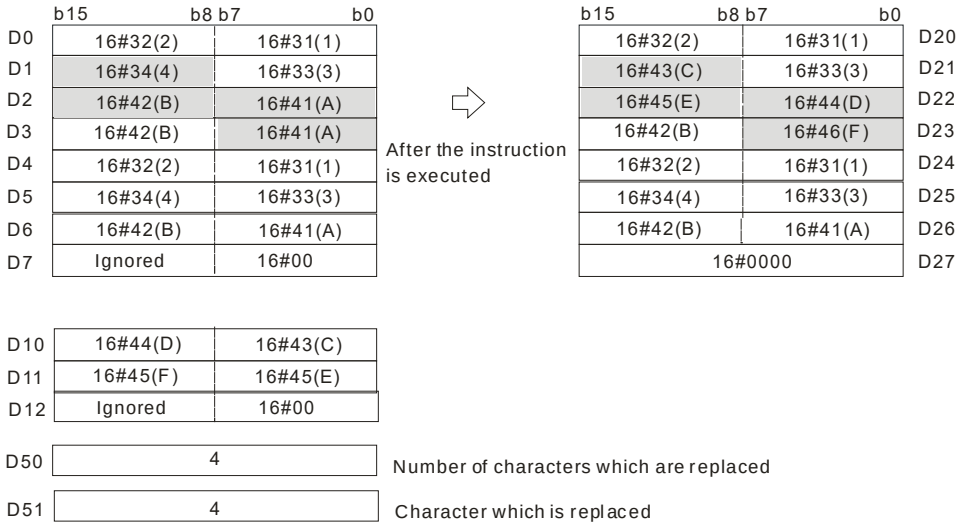
When M0 is ON, the data in D0–D7 is “1234ABAB1234AB”, and the data in D10–D11 is “CDEF”. When the $RPLC instruction executes, the instruction replaces the characters in D0–D7 starting from the character indicated by the value in D51 with the characters in D10–D11. The number of characters to replace is indicated by the value in D50, and the instruction stores the result in D20–D27.



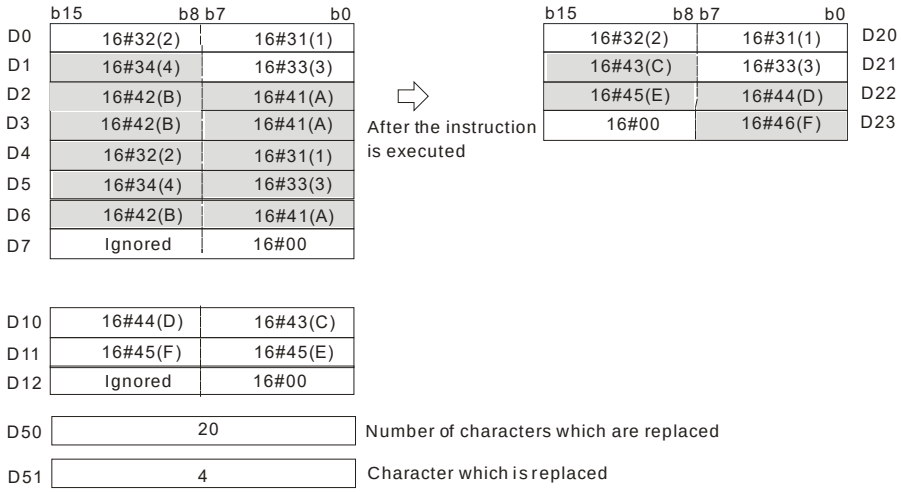
If the values in D50 and D51 are 3 and 4 respectively, the execution result is as follows.



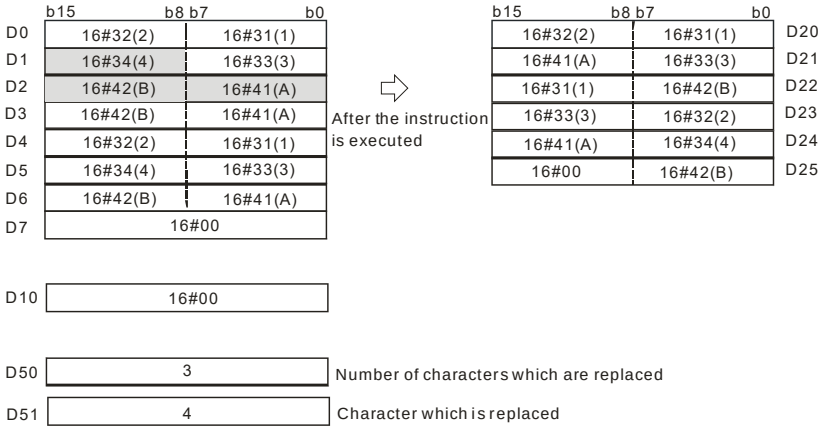
If the values in D50 and D51 are 4 and 4 respectively, the execution result is as follows.



If the values in D50 and D51 are 20 and 4 respectively, the execution result is as follows.



If the values in D50, D51, and D10 are 3, 4, and 16#00 respectively, the execution result is as follows. The three characters in D0-D7 starting from the fourth character are deleted.



Additional remarks

1. If the string in **S_1** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in **S_2** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
3. If the value in **S_3** <0 or **S_4**<=0 or the value in **S_4** is larger than the length of the string in **S_1**, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the value in the string (16#00 the ending character included) in **S1** after the replacement is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

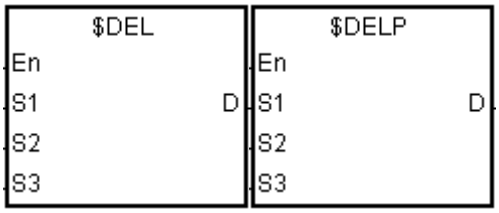
API	Instruction code			Operand	Function
2117		$DEL	P	**S_1, S_2, S_3, D**	Deleting the characters in a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●		○	○	○	○		
S_3					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							
S_3		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String

S_2 : Number of characters to be deleted

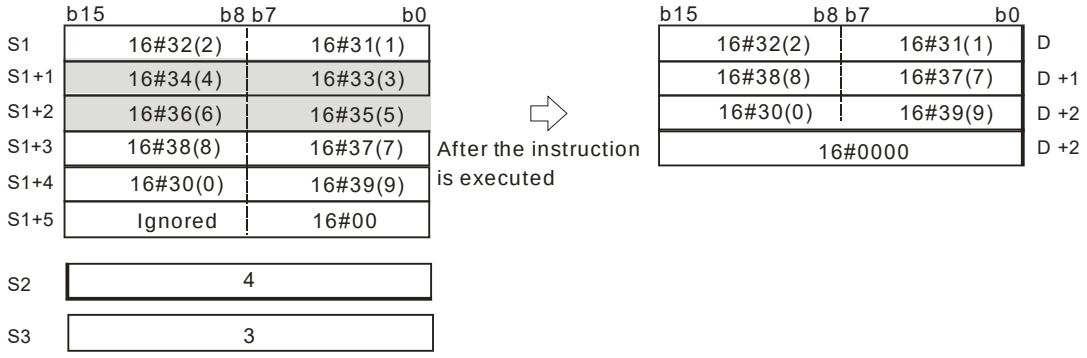
S_3 : Characters in **S_1** are deleted starting from the character indicated by the value in **S_3**.

D : Device where the execution result is stored

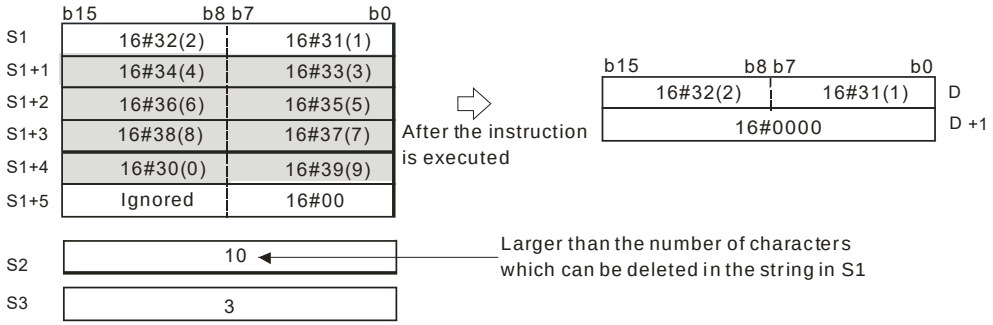
Explanation

1. This instruction deletes the characters in **S_1** starting from the character indicated by the value in **S_3**. The number of characters that are deleted is indicated by the value in **S_2**, and the instruction stores the result in **D**.

2. For example, the instruction deletes the four characters starting from the third character in the string "12**3456**7890" in **S_1**, and stores the result "127890"in **D**.



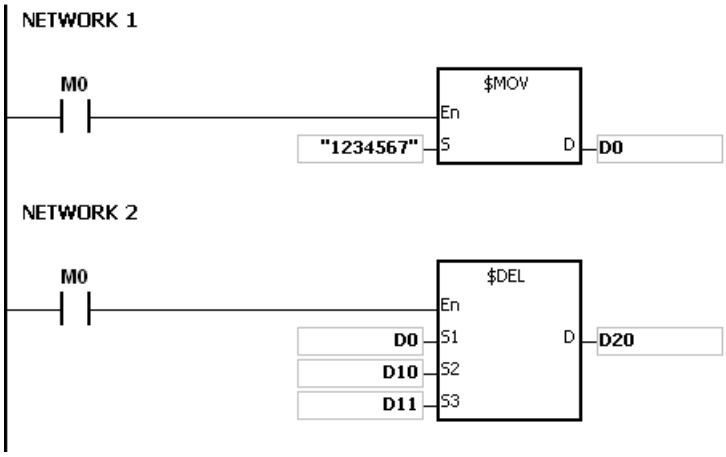
3. If the value in **S_2** is larger than the number of characters which can be deleted in the string in **S_1**, the instruction deletes the characters in **S_1** starting from the character indicated by the value in **S_3** to the last character in **S_1**, and stores 16#00 in **D**.



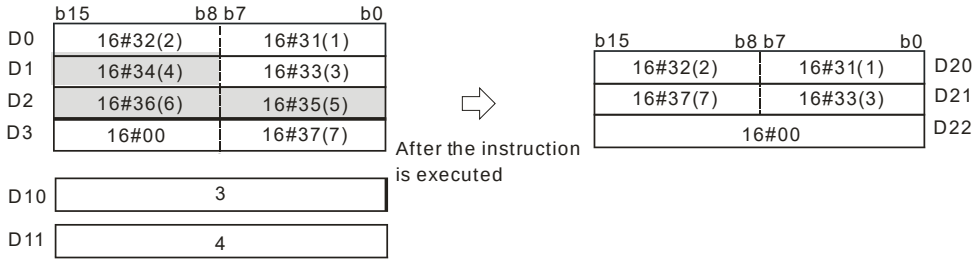
4. If the value in **S_2** is equal to 0, the instruction is not executed.

Example

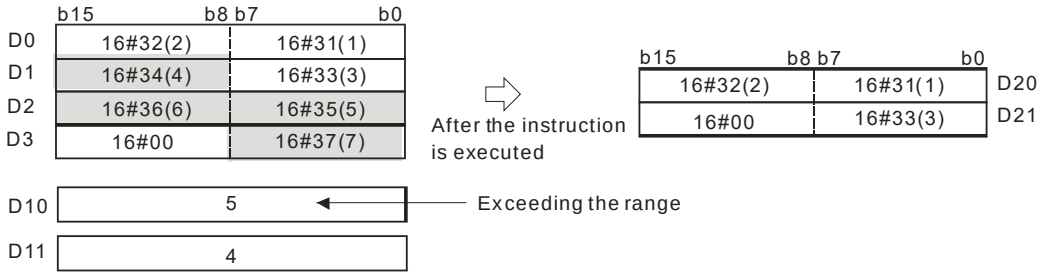
When M0 is ON, the data in D0–D3 is “1234567”. When the $DEL instruction executes, it deletes the characters in D0–D3 starting from the character indicated by the value in D11. The number of characters to delete is indicated by the value in D10, and the instruction stores the result in D20–D23.



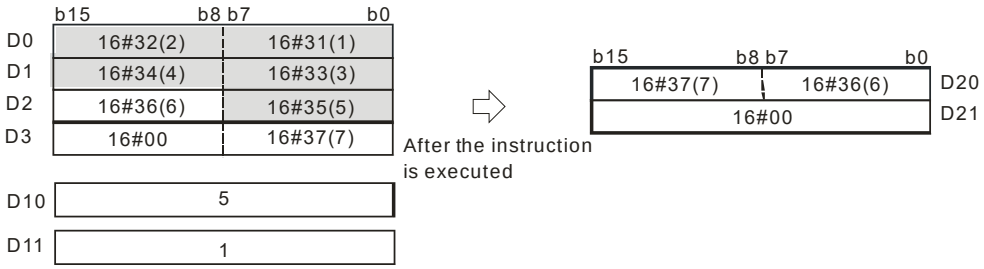
If the values in D10 and D11 are 3 and 4 respectively, the execution result is as follows.



If the values in D10 and D11 are 5 and 4 respectively, the execution result is as follows. Note that the number of characters to delete exceeds the range, the instruction deletes the characters in D0–D3 starting from the fourth character to the last character.



If the values in D10 and D11 are 5 and 1 respectively, the execution result is as follows.



Additional remarks

1. If the string in S_1 does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

2. If the value in S_2 is less than 0, the value in S_3 is less than or equal to 0, or the value in S_3 is larger than the length of the string in S_1, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code			Operand	Function
2118		$CLR	P	**S**	Clearing a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S : String to be cleared

Explanation

1. This instruction clears the string in **S**. If the string in **S** does not end with 16#00, the instruction clears up to 255 characters.

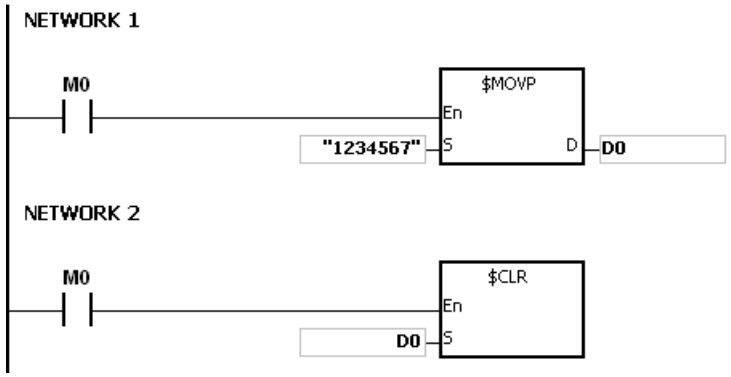
	b15 ~ b8	b7 ~ b0
S	16#31(1)	16#2D(-)
S +1	16#33(3)	16#32(2)
S +2	16#34(4)	16#35(5)
S +3	16#36(6)	16#2E(.)
S +4	16#38(8)	16#37(7)
S +5	16#00	

After the instruction is executed

S	16#0000
S +1	16#0000
S +2	16#0000
S +3	16#0000
S +4	16#0000
S +5	16#0000

Example

The instruction clears the string in D0 as illustrated below.



	b15 b8	b7 b0
D0	16#32(2)	16#31(1)
D1	16#34(4)	16#33(3)
D2	16#36(6)	16#35(5)
D3	16#00	16#37(7)

After the instruction is executed

16#0000	D0
16#0000	D1
16#0000	D2
16#0000	D3

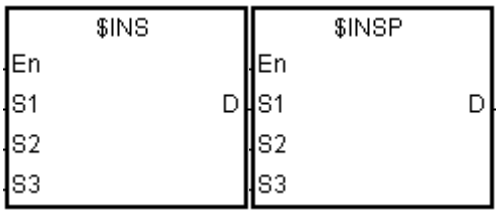
API	Instruction code			Operand	Function
2119		$INS	P	**S_1, S_2, S_3, D**	Inserting a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●						○	
S_2					●	●		●	●						○	
S_3					●	●		●	●		○	○	○	○		
D					●	●		●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							●
S_3		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String

S_2 : String to be inserted

S_3 : The string **S_2** is inserted into **S_1** after the character indicated by the value in **S_3**.

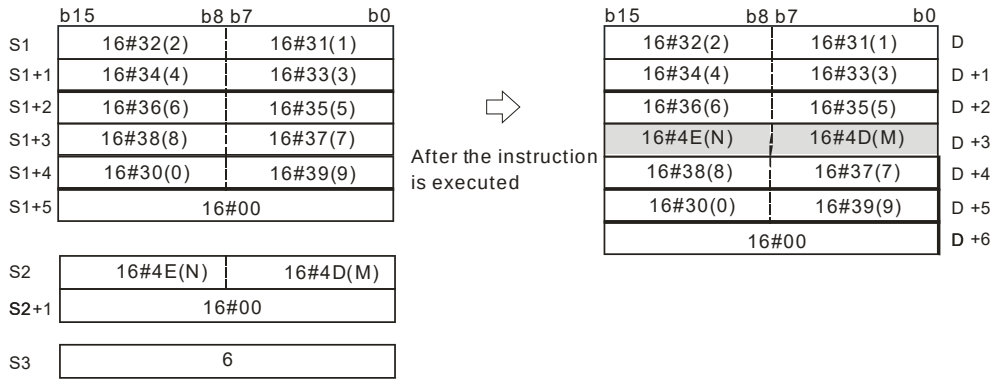
D : Device in which the execution result is stored

Explanation

1. This instruction inserts the string in **S_2** into the string in **S_1** after the character indicated by the value in **S_3**, and stores the result in **D**.

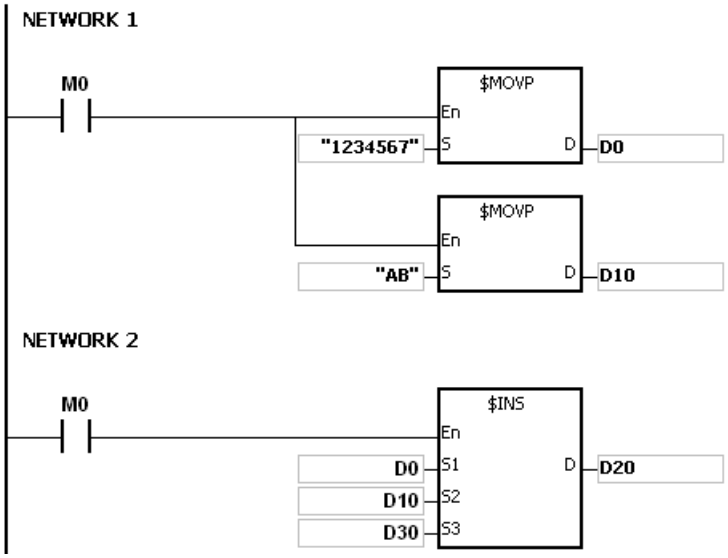
2. If the string in either **S_1** or **S_2** is a null string, the instruction stores the other string (**S_1** or **S_2**) that is not a null string in **D**.

3. If the strings in **S_1** and **S_2** are null strings, the instruction stores 16#0000 in **D**.

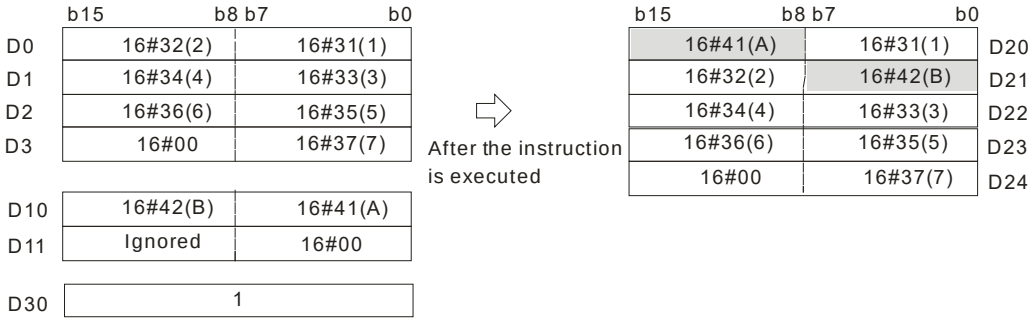


Example

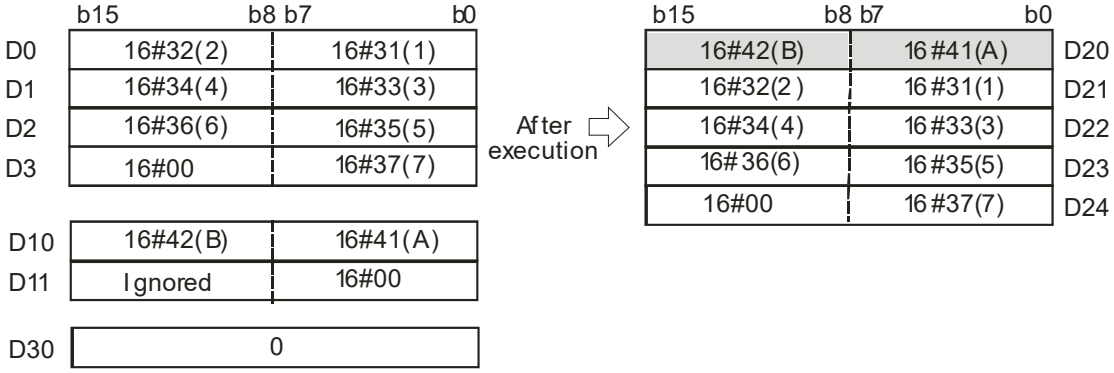
When M0 is ON, the data in D0–D3 is “1234567”, and the data in D10 is “AB”. When the $INS instruction executes, it inserts “AB” into the string in D0–D3 after the character indicated by the value in D30, and stores the result in D20–D24.



If the value in D30 is 1, the execution result is as follows.



If the value in D30 is 0, the execution result is as follows.



Additional remarks

1. If the string in **S_1** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in **S_2** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
3. If the value in **S_3** is less than 0, or if the value in **S_3** is larger than the length of the string in **S_1**, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the inserted value in the string (16#00 the ending character included) in **S1** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

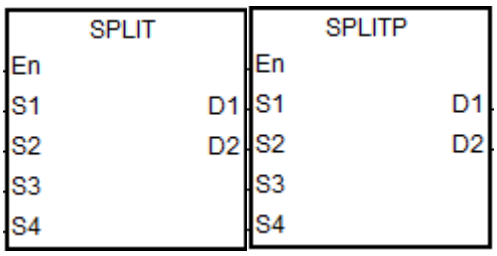
API	Instruction code			Operand	Function
2122		SPLIT	P	S_1 · S_2 · S_3 · S_4 · D_1 · D_2	Splitting a string

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●							
S_2								●					○	○	○	
S_3								●					○	○	○	
S_4								●					○	○		
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							●
S_3		●			●	●							●
S_4		●			●	●							
D_1		●			●	●							
D_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String

S_2 : Splitter

S_3 : Designated ending character of a splitter

S_4 : Storing Length for a string after a splitter

D_1 : Device in which the string after a splitter is stored

D_2 : Device in which the quantity of split strings is stored

Explanation

1. S_1 is the string source. For characters in a string, only the characters before an ending character or the value in S_3 will be used as a valid string.

2. S_2 is the splitter that separates strings. S_3 is a designated ending character of a splitter. Only values in the lower 8-byte are valid for S_2 and S_3. However, if you input a value other than the value in the lower 8-byte, PLC does not send any error message. When S_2 and S_3 use D devices to store values, only use lower 8-byte for values in ASCII format. Values in S_2 and S_3 can be a constant number, an ASCII or a string.

3. S_4 is the storing length for a string after a splitter; unit: WORD. The ending characters, such as 16#00 and 16#0000 should be included in the length. The maximum WORDs a string can contain is S_4 and a WORD equals to 2 bytes. An ending character 16#00 takes a byte. For example, if S_4 is 2, the maximum WORDs a string can contain is 2

WORDs and 2 WORDs equals to 4 bytes. And that means a string can contain 3 characters and 1 ending character (16#00). The setting range for **S_4** is 1-100. (If the input value is out of the range, the value will be seen as the minimum or the maximum during operation and PLC does not send any error message.)

4. **D_2** is the sum of all the split strings.

5. The result of **D_1~ D_1+ S_4 *D_2** –1 is the D devices that are occupied.

   For example, if **D_2** is 3, that means the 1st string will be stored from **D_1**+0 and the 2nd string will be stored in **D_1+ S_4** and the 3rd string will be stored in **D_1+ S_4** x2.

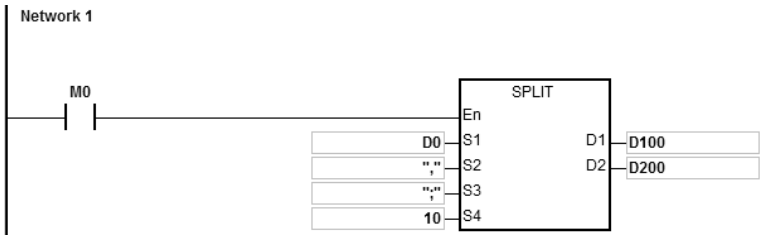
6. The instruction looks for a string for a length defined in **S_4** (WORD) till there is an ending character or a splitter. Add 1 to the value in **D_2**. Whenever one string is found,

7. Up to 256 characters can be stored in **S_1** and up to 20 sets of strings can be stored in **D_2**. When any of the limitation is reached even when there is no ending character, the splitting stops.

Example

Use "," as a splitter to split a string "abcd,1234,5.67,8910;ef" and use ";" as an ending character that stored in D0-D11. And store the results in the register starting from D100 in a unit of 10 WORDs.

Note: the string to be split does not include the characters "".



Explanation

1. The results of this example D0~D11="abcd,1234,5.67,89103;edf".

Device	Value	String
D0	16#6261	"ba"
D1	16#6463	"dc"
D2	16#312C	"1,"
D3	16#3332	32
D4	16#2C34	",4"
D5	16#2E35	".5"

Device	Value	String
D6	16#3736	“76”
D7	16#382C	“8,”
D8	16#3139	“19”
D9	16#3330	“30”
D10	16#453B	“e;”
D11	16#4644	“fd”

2. Starting contact is M0

3. The quantity of split strings D200=4

4. The split strings are as below.

1st set: D100~D109 are occupied; D100-D101=“abcd”	D100	D101	D102	D103	D104	D105	D106	D107	D108	D109
HEX	6261	6463	0000	0000	0000	0000	0000	0000	0000	0000
String	“ba”	“dc”	16#0000	-	-	-	-	-	-	-
2nd set: D110~D119 are occupied; D110~D111=“1234”	D110	D111	D112	D113	D114	D115	D116	D117	D118	D119
HEX	3231	3433	0000	0000	0000	0000	0000	0000	0000	0000
String	“21”	“43”	16#0000	-	-	-	-	-	-	-

3rd set: D120~D129 are occupied; D120~D121=“5.67”

Device	D120	D121	D122	D123	D124	D125	D126	D127	D128	D129
HEX	2E35	3736	0000	0000	0000	0000	0000	0000	0000	0000
String	“.5”	“76”	16#0000	-	-	-	-	-	-	-

4th set: D130~D139 are occupied, D130~D131="89103"

Device	D130	D131	D132	D133	D134	D135	D136	D137	D138	D139
HEX	3938	3031	0033	0000	0000	0000	0000	0000	0000	0000
String	"98"	"01"	Upper: 3" Lower: 16#00	-	-	-	-	-	-	-

Additional remarks

1. Up to 256 characters can be stored in **S_1** if there is no ending character (16#00) or splitter, the instruction will execute on the 256 characters. SM0 will not be ON.
2. Ending character 16#00 or 16#0000 will be added in a string after it is split (**D_1**). See the example above.
3. Ending character 16#00 or 16#0000 will be added in a string after it is split (**D_1**) even when there is no string behind the split string.

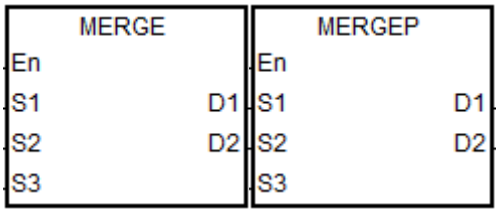
API	Instruction code			Operand	Function
2123		MERGE	P	**S_1**、**S_2**、**S_3**、**D_1**、**D_2**	Merging strings

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●						○	
S_2								●	●						○	
S_3								●					○	○	○	
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							●
S_2		●			●	●							●
S_3		●			●	●							●
D_1		●			●	●							
D_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : String source 1

S_2 : String source 2

S_3 : Splitter

D_1 : Device in which the merged string is stored

D_2 : Device in which the quantity of merged strings is stored

Explanation

1. **S_1** and **S_2** are the string source 1 and 2 respectively. **S_3** is the splitter that separates strings. Only values in the lower 8-byte are valid for **S_3**. **S_1** merges **S_2** directly, when **S_3** is 0 (16#00). **S_1** adds the splitter and then merges **S_2**, when **S_3** is not 0.

2. **D_1** is where to store a merged string. The merging order is string 1 + splitter + string 2 + ending character 0 (16#00). You can self-define an ending character. This self-defined ending character will be added before the official one, 0 (16#00).

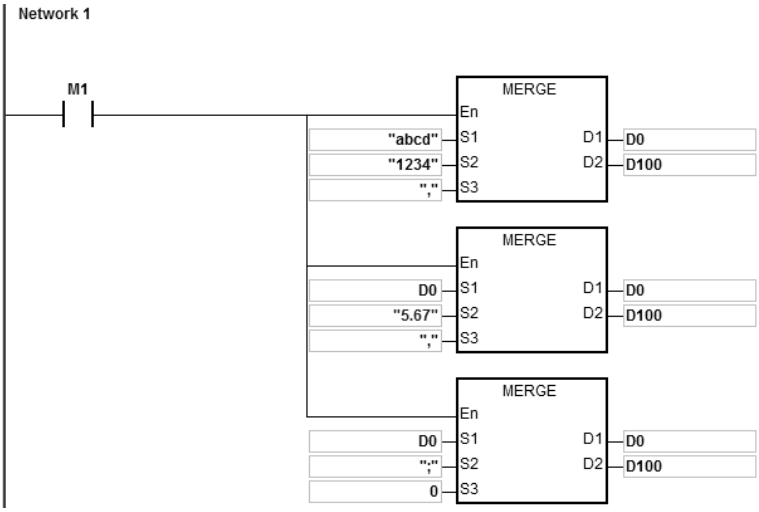
3. If the values in **S_1**, **S_2**, **S_3** are all 0, **D_1** is 16#0000.

4. **D_2** is the sum of all the merged strings (ending character 16#00 excluded). The accumulated length for the string is the length of string 1 or the length of string 1 + splitter + the length of string 2.

5. If the string sources for **S_1** and **S_2** are both 0 or when strings are merged, the length for the merged string is up to 255 words (ending character 16#00 excluded). For example, there are 250 words in String 1 and 100 words in String 2. After they are merged, the length for the merged string is 250 words from String 1 + splitter *1 + first 4 words from String 2.

Example

Use “,” as a splitter to split a string and use “;” as an ending character and then merge 3 strings “abcd”, ”1234” and ”5.67”.



And store the results “abcd,1234,5.67;” D100=15, in the register starting from D0-D7.

Explanation

1. Starting contact is M0
2. The 1st program: merge the 1st string “abcd” and 2nd string “1234” together and put a splitter “,” in between and then store the result in the registers starting from D0.

   The merged result: D0~D4= “abcd,1234”; the quantity of merged strings: D100=9

Device	D0	D1	D2	D3	D4
HEX	6261	6463	312C	3332	0034
String	“ba”	“dc”	“1,”	“32”	Upper 8-byte: 16#00 ; lower 8-byte: 4

3. The 2nd program: merge the 1st string D0=“abcd,1234” and 2nd string ”5.67” together and put a splitter “,” in between and then store the result in the registers starting from D0.

   The merged result: D0~D7= “abcd,1234,5.67”; the quantity of merged strings: D100=14

Device	D0	D1	D2	D3	D4	D5	D6	D7
HEX	6261	6463	312C	3332	2C34	2E35	3736	0000
String	“ba”	“dc”	“1,”	“32”	“,4”	“.5”	“76”	16#0000

4. The 3rd program: merge the 1st string D0="abcd,1234,5.67" and 2nd string "," together and put a splitter "0" in between (merge the string 1 and 2 directly) and then store the result in the registers starting from D0.

   The merged result: D0~D7= "abcd,1234,5.67;"; the quantity of merged strings: D100=15.

Device	D0	D1	D2	D3	D4	D5	D6	D7
HEX	6261	6463	312C	3332	2C34	2E35	3736	003B
String	"ba"	"dc"	"1,"	"32"	",4"	".5"	"76"	Upper 8-byte: 16#00 ; lower 8-byte: ";"

Additional remarks

1. If the string in **S_1** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in **S_2** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
3. If **D_1** + (**D_2**+1)/2 – 1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. Ending character 16#00 or 16#0000 will be added behind the merged string. See the examples above.

6.22 Ethernet Instructions

6.22.1 List of Ethernet Instructions

The following table lists the String Processing instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2200	SOPEN	–	✓	Opening a socket
2201	SSEND	–	✓	Sending data through a socket
2203	SCLOSE	–	✓	Closing a socket
2204	MSEND	–	✓	Sending an email
2206	INTOA	–	✓	Converting an integer IP address into a string IP address
2207	IATON	–	✓	Converting a string IP address into an integer IP address
2208	EIPRW	–	–	Reading and writing EtherNet/IP data
2209	SCONF	–	✓	Setting the TCP/UDP Socket parameters
2210	MCONF	–	✓	Reading and writing Modbus TCP data
2211	EMCONF1	–	✓	Setting email server parameter values
2212	EMCONF2	–	✓	Setting email address

6.22.2 Explanation of Ethernet Instructions

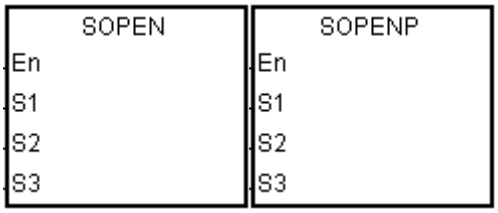
API	Instruction code		Operand	Function
2200	SOPEN	P	S_1, S_2, S_3	Opening a socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●			○	○	○		
S_2					●	●		●	●			○	○	○		
S_3					●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Socket mode

S_2 : Socket number

S_3 : Communication mode

Explanation

1. This instruction opens a TCP or UDP socket. Set S_1 to 1 to open a TCP socket, and set S_1 to 0 to open a UDP socket.

2. S_2 is the socket number. It should correspond to the Ethernet socket number; up to 4 can be set.

3. The ES3 Series PLC functions as a client and sends a TCP connection request to the server if S_3 is 1, and the ES3 Series PLC functions as a sever and waits for a TCP connection request from the client if S_3 is 0. To start a UDP connection, set S_3 to 0 or 1.

4. The operand S_1 must be either 0 or 1 (default: 0); the operand S_2 must be between 1–4 (default: 1); the operand S_3 must be either 0 or 1 (default: 0). If exceeding the range, error code H'2003 shows on the PLC.

5. Before using the instruction, set the following setting in HWCONFIG in ISPSoft.

   - PLC Parameter Setting → Ethernet-Basic → Setting IP address and netmask address
   - PLC Parameter Setting → Ethernet-Advanced → Socket → Enable Socket Function

- PLC Parameter Setting → Ethernet-Advanced → Socket → TCP/UDP Socket Connection and Setting sockets to be used

6. Refer to the SCONF instructions (API 2209) for more on the TCP and UDP Socket parameters.

7. When the TCP Socket is opened, use the following settings for the Socket IP and communication ports.

S_3 Start Mode	**Remote IP**	**Local Communication Port**	**Remote Communication Port**	**Description**
1	Specific IP address	0	0	Illegal
1	Specific IP address	0	Not equal to 0	Client mode: Specifies IP address, but not local communication port
1	Specific IP address	Not equal to 0	0	Illegal
1	Specific IP address	Not equal to 0	Not equal to 0	Client mode: Specifies IP address, local communication port, and remote communication port
1	0.0.0.0	No limit to the value	No limit to the value	Illegal
0	Specific IP address	0	No limit to the value	Illegal
0	Specific IP address	Not equal to 0	0	Server mode: Specifies IP address, but not remote communication port
0	Specific IP address	Not equal to 0	Not equal to 0	Server mode: Specifies IP address and remote communication port
0	0.0.0.0	0	No limit to the value	Illegal
0	0.0.0.0	Not equal to 0	0	Server mode: Does not specify IP address or remote communication port
0	0.0.0.0	Not equal to 0	Not equal to 0	Server mode:

S_3 Start Mode	Remote IP	Local Communication Port	Remote Communication Port	Description
				Does not specify IP address, but does specify remote communication port

8. If data is sent through a TCP socket, and no error occurs after the instruction is executed, the socket starts to establish the connection with the remote device and the flag for starting connection is ON. If the connection is made successfully, the flag for successful connection is ON, and the flag for starting connection is OFF. If an error occurs, the error flag is ON.

TCP Socket Number	Connection Started	Data Received	Data Sent	Connection Starting	Connection Closed	Data Sending	Error Flag
1	SM1270	SM1271	SM1272	SM1273	SM1274	SM1275	SM1277
2	SM1278	SM1279	SM1280	SM1281	SM1282	SM1283	SM1285
3	SM1286	SM1287	SM1288	SM1289	SM1290	SM1291	SM1293
4	SM1294	SM1295	SM1296	SM1297	SM1298	SM1299	SM1301

9. If data is transmitted through a UDP socket, and no error occurs after the instruction is executed, the flag for started connection is ON. If an error occurs, the error flag is ON.

UDP Socket Number	Connection Started	Data Received	Data Sending	Connection Closed	Error Flag
1	SM1334	SM1335	SM1336	SM1337	SM1338
2	SM1339	SM1340	SM1341	SM1342	SM1343
3	SM1344	SM1345	SM1346	SM1347	SM1348
4	SM1349	SM1350	SM1351	SM1352	SM1353

10. If data is transmitted through a TCP Socket in Server mode or through a UDP Socket, it is not allowed to use the following communication port number for ES3 series PLC.

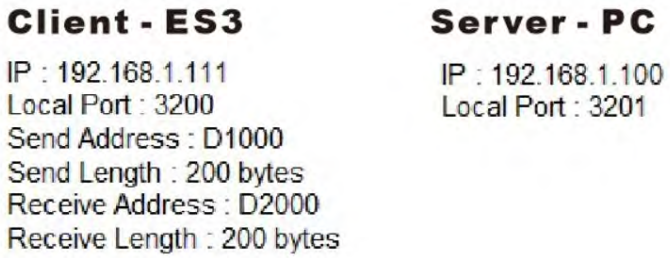
Socket Type	Communication Port Number	Communication Protocol
TCP	80	HTTP
TCP	502	MODBUS TCP
TCP	44818	EtherNet/IP
UDP	68	DHCP/BOOTP

Socket Type	Communication Port Number	Communication Protocol
UDP	2222	EtherNet/IP
UDP	44818	EtherNet/IP
UDP	20006	Self-defined protocol for ES3 series

11. In general, the pulse SOPENP instruction is used.

Example 1

1. This example illustrates how to establish a TCP connection between a computer as server and an ES3 Series PLC as client.



2. ISPSoft → HWCONFIG (Ethernet-Basic)

Ethernet Configuration	
IP Address Mode	Static
IP Address	192.168.1.111
Netmask Address	255.255.255.0
Gateway Address	192.168.1.1

3. a) ISPSoft → HWCONFIG (Ethernet-Advanced → Socket → TCP Socket Connection)

   b) SCONF instruction

TCP Socket Connection	
Remote IP	192.168.1.110
Remote Port	3201
Local Port	3200
Send Data Address	D1000
Send Data Length	200
Receive Data Address	D2000
Receive Data Length	200
Keep Alive Timer	10

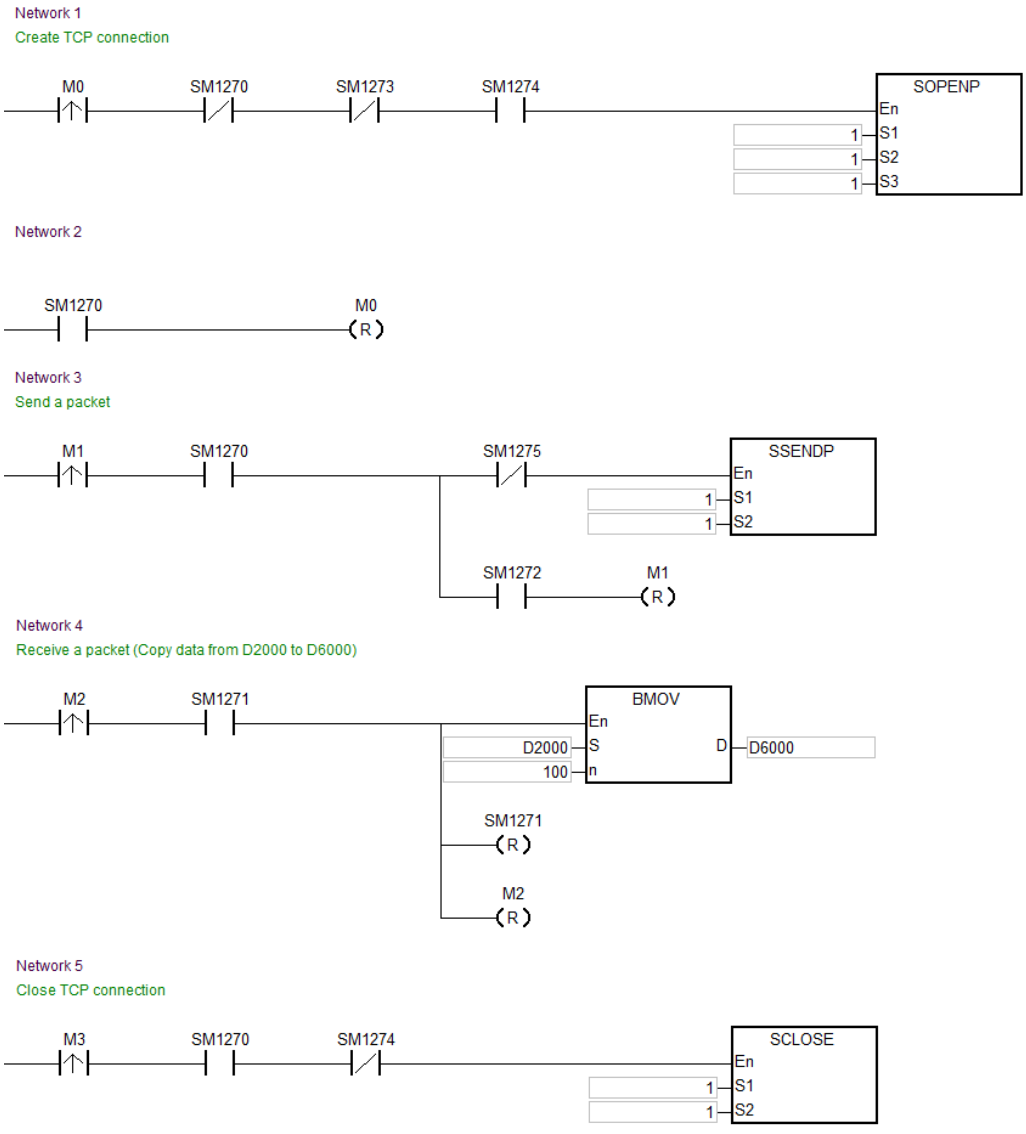
4. First, set the parameters related to TCP Socket1 in HWCONFIG in ISPSoft or in the SCONF instruction.

5. When M0 is ON, the system checks whether the socket is closed, has been connected, or is being connected. If the socket is not closed, has not been connected, and is not being connected, the system performs the connection procedure. After the socket has been connected, M0 is switched to OFF.

6. When M1 is ON, the system checks whether the socket has been connected and no data is being sent. If the socket has been connected and no data is being sent, the system sends the data. If the socket has not been connected, the system does not execute the instruction. After data has been sent, M1 is switched to OFF.

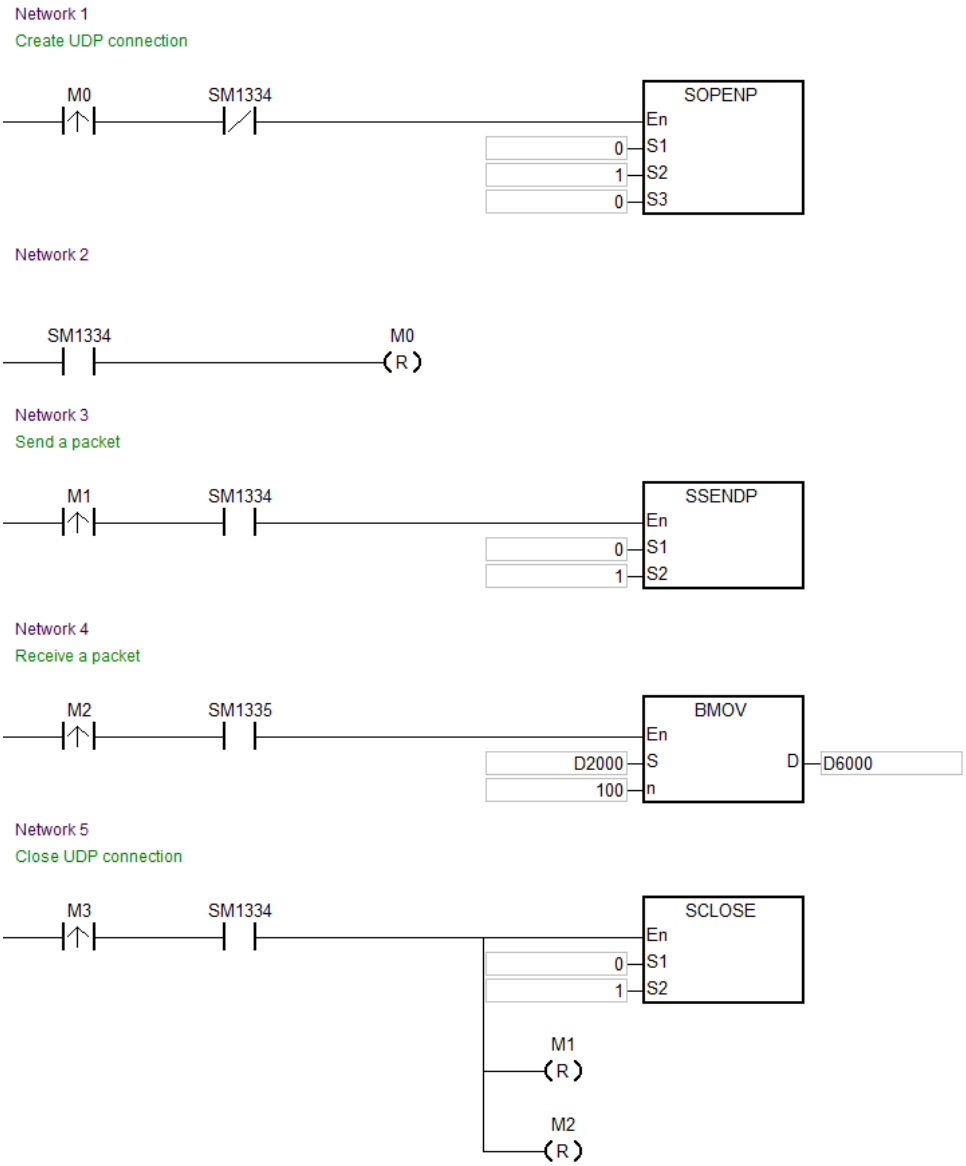
7. When M2 is ON, the system checks whether the socket has been connected and the flag for received data is ON. If the socket has been connected and the flag for received data is ON, then the data has already been received.

8. When M3 is ON, the system checks whether the socket has been connected. If the socket has been connected, the system closes the connection. If the socket has not been connected, the system does not execute the instruction. After the connection is closed, M2 and M3 are switched to OFF.



Example 2

1. This example illustrates how to establish a UDP connection between a computer and an ES3 Series PLC.
2. When M0 is ON, the system checks whether the socket has been connected. If the socket has not been connected, the system performs the connection procedure. After the socket has been connected, M0 is switched to OFF.
3. When M1 is ON, the system sends the data. After the data has been sent, M1 is switched to OFF.
4. When M2 is ON, the system checks whether the socket has been connected and the flag for received data is ON. If the socket has been connected and the flag for received data is ON, the data has already been received.
5. When M3 is ON, the system checks whether the socket has been connected. If the socket has been connected, the system closes the connection. If the socket has not been connected, the system does not execute the instruction. After the connection is closed, M2 and M3 are switched to OFF.



Additional remarks

1. The following table lists TCP connection errors.

Error Code (SR180)	Error Flag	Description
16#2003	SM0	**S_1**, **S_2**, or **S_3** exceeds the range
16#600C	SM1109	Local socket not used
16#600D	SM1100	Ethernet network not connected
16#6200	TCP Socket error flag	Illegal IP address
16#6201	TCP Socket error flag	Illegal TCP Socket communication mode setting
16#6202	TCP Socket error flag	Illegal TCP Socket mode setting
16#6203	TCP Socket error flag	Illegal address for sending data
16#6204	TCP Socket error flag	Length of sent data exceeds the range
16#6205	TCP Socket error flag	Source data device exceeds the range
16#6206	TCP Socket error flag	Illegal address for receiving data
16#6207	TCP Socket error flag	Length of received data exceeds the range
16#6208	TCP Socket error flag	Target data device exceeds the range
16#6212	TCP Socket error flag	TCP Socket communication timeout
16#6213	TCP Socket error flag	Size of data received larger than dataset
16#6214	TCP Socket error flag	TCP Socket connection rejected by remote equipment
16#6215	TCP Socket error flag	TCP Socket not connected
16#6217	TCP Socket error flag	TCP Socket connection triggered
16#6218	TCP Socket error flag	Sending data through TCP Socket triggered
16#621A	TCP Socket error flag	Disabling TCP Socket connection triggered

2. The following table explains UDP connection errors.

Error Code (SR180)	Error Flag	Description
16#2003	SM0	**S_1**, **S_2**, or **S_3** exceeds the range
16#600C	SM1109	Local socket used
16#600D	SM1100	Ethernet network not connected

Error Code (SR180)	Error Flag	Description
16#6209	UDP Socket error flag	Illegal IP address
16#620A	UDP Socket error flag	Illegal communication mode
16#620C	UDP Socket error flag	Illegal address for sending data
16#620D	UDP Socket error flag	Length of sent data exceeds the range
16#620E	UDP Socket error flag	Source data device exceeds the range
16#620F	UDP Socket error flag	Illegal address for receiving data
16#6210	UDP Socket error flag	Length of data received exceeds the range
16#6211	UDP Socket error flag	Target data device exceeds the range
16#6213	UDP Socket error flag	Size of data received larger than dataset
16#6215	UDP Socket error flag	UDP Socket not connected
16#6217	UDP Socket error flag	UDP Socket connection triggered
16#6218	UDP Socket error flag	Sending of data through UDP Socket triggered
16#621A	UDP Socket error flag	Disabling UDP Socket connection triggered

3. When the client and server are both ES3 Series PLC and the communication timeout settings are set the same, the server automatically cuts the connection if a timeout occurs on the server first. In this case, the error flag on the client is not ON. On the other hand, if a timeout occurs on the client first, the error flag on the client is ON and the connection is cut.

API	Instruction code			Operand	Function
2201		SSEND	P	S_1, S_2	Sending data through a socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●			○	○	○		
S_2					●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

S_1 : Socket mode

S_2 : Socket number

Explanation

1. This instruction sends data through the socket specified by **S_1**. Set **S_1** to 1 to open a TCP socket, and set **S_1** to 0 to open a UDP socket.

2. **S_2** is the socket number. It should correspond to the Ethernet socket number; up to 4 can be set.

3. The operand **S_1** must be either 0 or 1 (default: 0); the operand **S_2** must be between 1–4 (default: 1). If exceeding the range, error code H'2003 shows on the PLC.

4. Before using this instruction, use the SOPEN instruction (API 2200) to open the socket. If the flags for the successful connection of the TCP socket or the flag for started connection of the UDP socket is ON, you can use this instruction.

5. If data is sent through a TCP socket, and no error occurs after the instruction is executed, the data is sent, and the flag for sending data is ON. If data is sent successfully, the flag for sent data is ON, and the flag for sending data is OFF. If an error occurs, the error flag is ON.

TCP Socket Number	Data Sending	Data Sent	Error Flag
1	SM1275	SM1272	SM1277
2	SM1283	SM1280	SM1285
3	SM1291	SM1288	SM1293
4	SM1299	SM1296	SM1301

6. If data is sent through a UDP socket, and no error occurs after the instruction is executed, the flag for sent data is ON. If an error occurs, the error flag is ON.

UDP Socket Number	Data Sending	Error Flag
1	SM1336	SM1338
2	SM1341	SM1343
3	SM1346	SM1348
4	SM1351	SM1353

7. In general, the SSENDP pulse instruction is used.

Example

Refer to the example in the SOPEN instructions (API 2200).

Additional remarks

1. The following table explains TCP connection errors.

Error Code	Error Flag	Description
16#2003	SM0	**S_1** or **S_2** exceeds the range
16#600D	SM1100	Ethernet network not connected
16#6202	TCP Socket error flag	Illegal TCP Socket mode setting
16#6203	TCP Socket error flag	Illegal address for sending data
16#6204	TCP Socket error flag	Length of sent data exceeds the range
16#6205	TCP Socket error flag	Source data device exceeds the range
16#6212	TCP Socket error flag	TCP Socket connection timeout
16#6214	TCP Socket error flag	TCP Socket connection rejected by remote equipment
16#6215	TCP Socket error flag	TCP Socket not connected
16#6218	TCP Socket error flag	Sending of data triggered

2. The following table explains UDP connection errors.

Error Code	Error Flag	Description
16#2003	SM0	**S_1** or **S_2** exceeds the range
16#600D	SM1100	Ethernet network not connected
16#620A	UDP Socket error flag	Illegal UDP Socket communication mode setting
16#620C	UDP Socket error flag	Illegal address for sending data
16#620D	UDP Socket error flag	Length of sent data exceeds the range
16#620E	UDP Socket error flag	Source data device exceeds the range
16#6218	UDP Socket error flag	Sending of data triggered

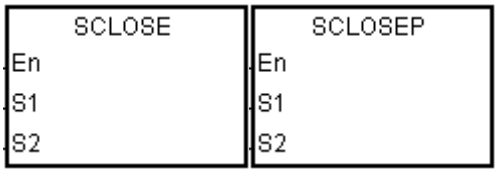
API	Instruction code			Operand	Function
2203		SCLOSE	P	S_1, S_2	Closing a socket

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●			○	○	○		
S_2					●	●		●	●			○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Socket mode

S_2 : Socket number

Explanation

1. This instruction closes the TCP or UDP socked specified by **S_1**. Set **S_1** to 0 to close a TCP socket, and set **S_1** to 1 to close a UDP socket.
2. **S_2** is the socket number. It should correspond to the Ethernet socket number; up to 4 can be set.
3. The operand **S_1** must be either 0 or 1 (default: 0); the operand **S_2** must be between 1–4 (default: 1). If exceeding the range, error code H'2003 shows on the PLC.
4. Before closing the socket, make sure that the socket is connected or else the instruction is not executed.
5. If the client closes a TCP socket, the server continues to be connected to the local communication port (the flag for successful connection is ON). If the server closes a TCP socket, the server is no longer connected to the local communication port after the execution of the instruction is complete. After a TCP socket is closed, the corresponding flags are OFF.
6. After the instruction is executed to close a UDP socket, the corresponding flag is OFF.
7. If a TCP socket is closed, and no error occurs after the instruction is executed, the connection with the remote device starts to close and the flag for closing connection is ON. If the connection is closed successfully, the flag for closing connection is OFF. If an error occurs, the error flag is ON.

TCP Socket Number	Connection Closed	Error Flag
1	SM1274	SM1277
2	SM1282	SM1285

TCP Socket Number	Connection Closed	Error Flag
3	SM1290	SM1293
4	SM1298	SM1301

8. If a UDP socket is closed, and no error occurs after the instruction is executed, the flag for started connection is OFF. If an error occurs, the error flag is ON.

UDP Socket Number	Error Flag
1	SM1338
2	SM1343
3	SM1348
4	SM1353

9. In general, the pulse SCLOSEP instruction is used.

Example

Refer to the example for the SOPEN instruction (API 2200).

Additional remarks

1. The following table lists the TCP connection errors.

Error Code	Error Flag	Description
16#2003	SM0	S_1 or S_2 exceeds the range
16#600D	SM1100	Ethernet network not connected
16#6212	TCP Socket error flag	TCP Socket communication timeout
16#6214	TCP Socket error flag	TCP Socket connection rejected by remote device
16#621A	TCP Socket error flag	Disabling TCP Socket connection triggered

2. The following table explains UDP connection errors.

Error Code	Error Flag	Description
16#2003	SM0	S_1 or S_2 exceeds the range
16#600D	SM1100	Ethernet network not connected
16#621A	UDP Socket error flag	Disabling UDP Socket connection triggered

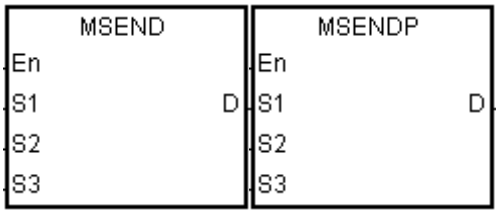
API	Instruction code			Operand	Function
2204		MSEND	P	**S_1, S_2, S_3, D**	Sending an email

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●		●	●			○	○	○		
S_2					●	●		●	●							
S_3					●	●		●	●							
D	●	●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1 : Remote email number

S_2 : Email subject

S_3 : Email body

D : Completion of the instruction

Explanation

1. This instruction creates and sends an email by setting **S_1**, **S_2**, and **S_3**.

2. Before using the instruction, set the following values in ISPSoft.

   - PLC Parameter Setting → Ethernet-Basic → Setting IP address and netmask address
   - PLC Parameter Setting → Ethernet-Advanced → Email → Setting SMTP server, port, local email address, and SMTP subject
   - PLC Parameter Setting → Ethernet-Advanced → Email and Trigger Configuration → Setting email address
   - If account identification is required: PLC Parameter Setting → Ethernet-Advanced → Email → Setting user name and password

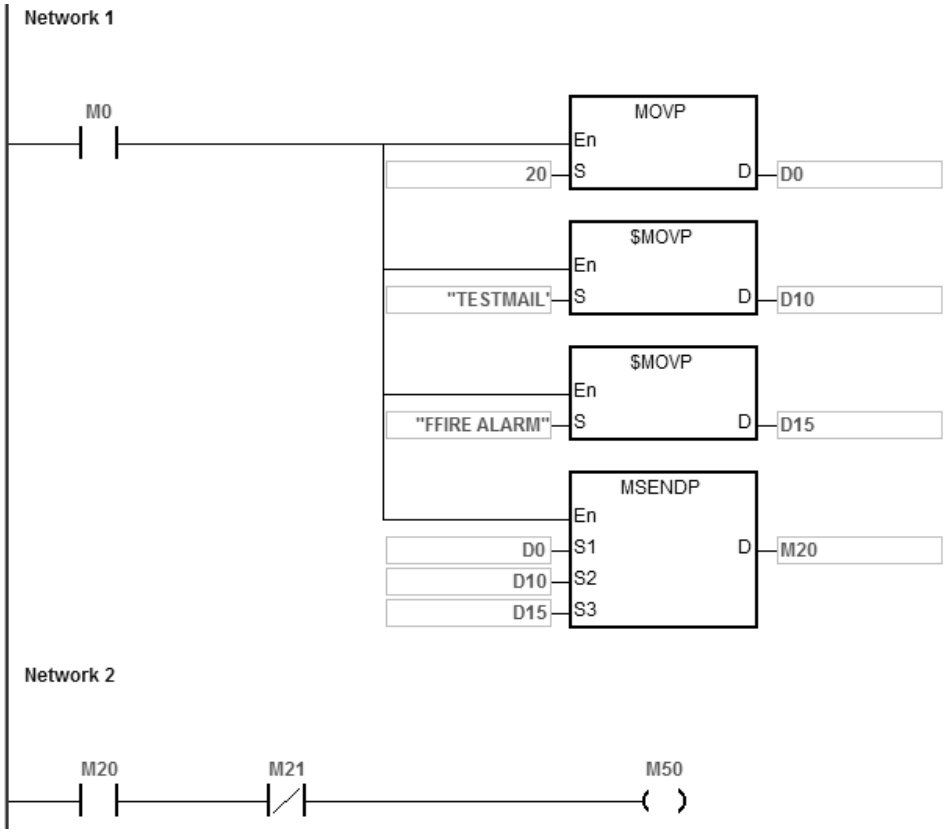
3. The email is set as follows.

Operand	Description	Setting Range
S_1	Remote email number	The value in **S_1** must be between 1–15. Set Bit 0–bit 3 in ISPSoft to indicate the remote email addresses. You can set up to four email addresses in ISPSoft. Bit 0 corresponds to email 1, bit 1 corresponds to email 2, and so on. If you want to send an email, you must set the corresponding bit in ISPSoft.
S_2	Email subject	Size of email subject: up to 16 words
S_3	Email body	Size of email body: up to 64 words
D	Completion of the instruction	After the execution of the instruction is complete, the bit is ON. If the execution of the instruction is abnormal, the next bit is ON.

4. In general, the MSENDP pulse instruction is used.

Example

Suppose the value in D0 is 00010100. When M0 is ON, the email is sent to email numbers 3 and 5. After the communication with the SMTP sever is complete, M20 is ON. If no error occurs during the communication, M21 is OFF and M50 is ON.



Additional remarks

1. For the length of the string in **S_2** or **S_3**, the system captures the data with the end of 16#00. If the length of the string is larger than the maximum value (with 16#00 as the end), the length of the string in **S_2** or **S_3** is processed as the maximum value.

2. If you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

3. Reserve one word between **S_2** and **S_3** for the interrupt character.

4. The following table explains errors in the execution of the email instruction.

Error Code	Error Flag	Description
16#2003	SM0	1. **D**+1 exceeds device range 2. **S_1**<1 or **S_1**>15
16#600D	SM1100	Ethernet network not connected
16#6100	D+1	Communication conflicts
16#6107	D+1	Email communication timeout
16#6108	D+1	Error in password authentication of SMTP server account
16#6111	D+1	Invalid remote email address

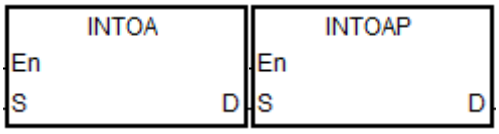
API	Instruction code			Operand	Function
2206		INTOA	P	**S**，**D**	Converting an integer IP address into a string IP address

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●	●							
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S ： Source value

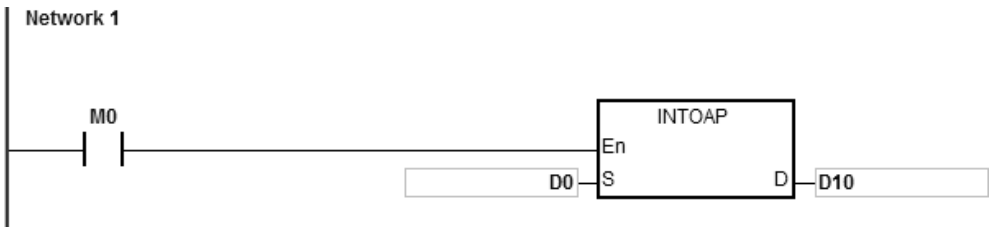
D ： Conversion result

Explanation

1. This instruction converts the IP address in **S** and **S**+1 into a string IP address, and stores the conversion result in **D**.

2. The operand **D** occupies eight consecutive devices.

Example



D0	16#C0(192)	16#A8(168)
D1	16#00 (0)	16#01 (1)

After the instruction is executed

D10	"9"	"1"
D11	"."	"2"
D12	"6"	"1"
D13	"."	"8"
D14	"0"	"0"
D15	"."	"0"
D16	"0"	"0"
D17	16#00	"1"

"192.168.0.1"

Additional remarks

1. If you declare the operand **S** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

2. If you declare the operand **D** in ISPSoft, the data type is ARRAY [8] of WORD/INT.

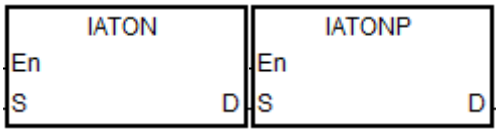
API	Instruction code			Operand	Function
2207		IATON	P	**S**，**D**	Converting a string IP address into an integer IP address

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
S								●	●						○	
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							●
D		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S ： Source value

D ： Conversion result

Explanation

1. This instruction converts the string IP address in **S** into an integer IP address, and stores the conversion result in **D** and **D**+1.

2. The operand **S** occupies eight consecutive devices.

"192.168.0.1"

S+0	"9"	"1"
S+1	"."	"2"
S+2	"6"	"1"
S+3	"."	"8"
S+4	"0"	"0"
S+5	"."	"0"
S+6	"0"	"0"
S+7	16#00	"1"

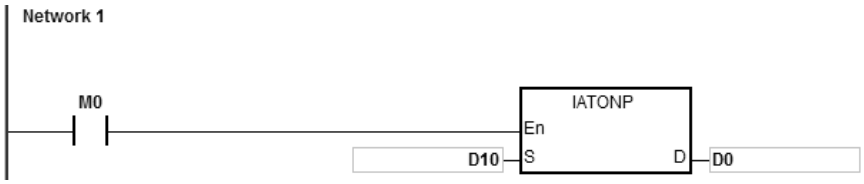
After the instruction is executed ⇨

D+0	16#C0(192)	16#A8(168)
D+1	16#00 (0)	16#01 (01)

3. There are 1–3 characters in every section of the string IP address in **S**. These sections are separated by “.” (16#2E). For example, you can enter “192.168.0.1” instead of “192.168.000.001”.

4. The value of each section of the string IP address in **S** must be between 0–255.

Example 1



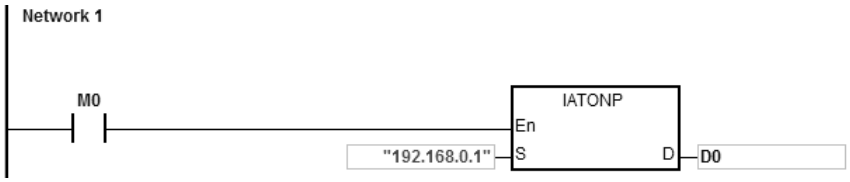
"192.168.0.1"

D10	"9"	"1"
D11	"."	"2"
D12	"6"	"1"
D13	"."	"8"
D14	"0"	"0"
D15	"."	"0"
D16	"0"	"0"
D17	16#00	"1"

After the instruction is executed

D0	16#C0(192)	16#A8(168)
D1	16#00 (0)	16#01 (1)

Example 2



"192.168.0.1"

After the instruction is executed

D0	16#C0(192)	16#A8(168)
D1	16#00(0)	16#01(1)

Additional remarks

1. If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. In the string in **S**, except for the code representing the decimal point, the binary codes must be between 16#30–16#39 or the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If there are not three decimal points "." in the string in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the value of any section of the string IP address in **S** is not between 0–255, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. The number of characters in any section of the string IP address in **S** must be between 1–3 or the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. If you declare the operand **S** in ISPSoft, the data type is ARRAY [8] of WORD/INT.
7. If you declare the operand **D** in ISPSoft, the data type is ARRAY [2] of WORD/INT.

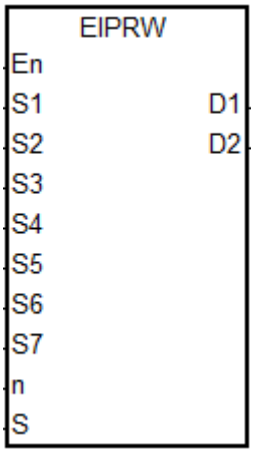
API	Instruction code			Operand	Function
2208		EIPRW		S_1-S_7，n，S，D_1，D_2	Reading and writing EtherNet/IP data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●							
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6	●	●	●	●												
S_7								●	●				○	○		
n								●	●				○	○		
S								●								
D_1								●								
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1-D_2	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1-S_7，n，S ： See Explanation below.

D_1-D_2 ： See Explanation below.

Explanation

1. This instruction reads and writes EtherNet/IP data. The following table lists the names and explanations of **S_1–S_7, n, S** and **D_1–D_2**.

Operand	Name	Description	Data Type	Remarks
S_1	IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example, if the IP address is 192.168.1.5, **S_1**=16#C0A8 and **S_1**+1=16#0105.	WORD[2]	Occupies two consecutive words
S_2	Connection mode	0: UCMM	WORD	
S_3	Function code (Service code)	Range: 16#0000–00FF. If the function code exceeds the range, the instruction is not executed.	WORD	
S_4	Class ID	Refer to EtherNet/IP protocol.	WORD	
S_5	Instance ID	Refer to EtherNet/IP protocol.	WORD	
S_6	Attribute ID switch	ON: Enable; OFF: Disable	BOOL	
S_7	Attribute ID	Refer to EtherNet/IP protocol.	WORD	
n	Length of read and write data	Size of the data to be written or read; unit: byte; maximum: 200 bytes	WORD	
S	Register for the read/write data	Source register for the sent data or where the received data is stored	WORD[**n**]	
D_1	Communication status	0：Communication not triggered 1：Communication in process 2：Communication completed without errors 3：Communication error 4： Error in parameter setting	WORD	
D_2	Error code	Major error code and extended error code	WORD[2]	Occupies two consecutive words

2. When you enable this instruction for the first time, it sends the communication command. If the number of connected slaves reaches the upper limit, the communication status value in **D_1** is 0, and the communication command is not sent.

3. When the parameter exceeds the range, the instruction is not executed and the communication status value in **D_1** is 4.

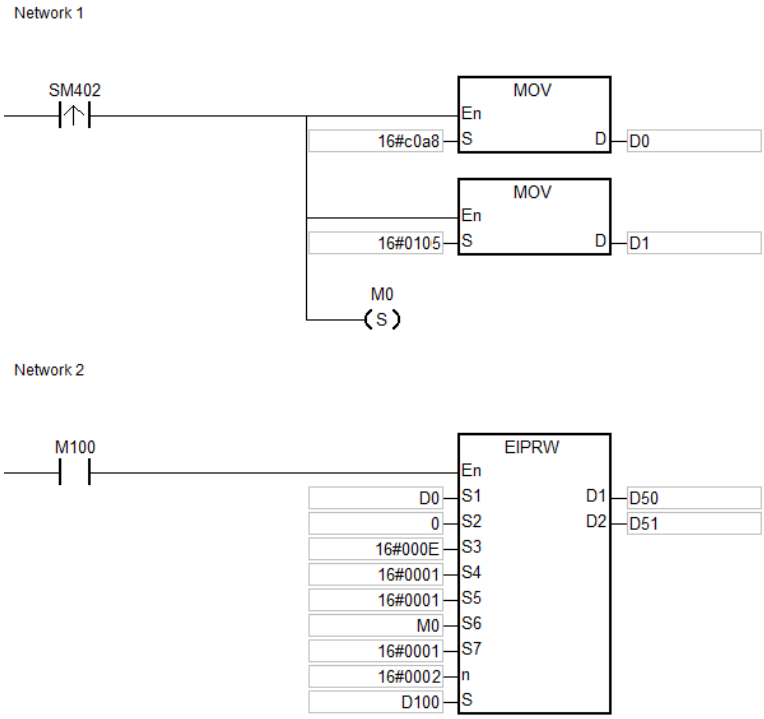
Example 1

The instruction reads the vendor code of the device at 192.168.1.5 and stores the value in D100. It sets the EtherNet/IP Object parameters as shown below.

(1) Class ID = 1

(2) Instance ID = 1

(3) Attribute ID = 1



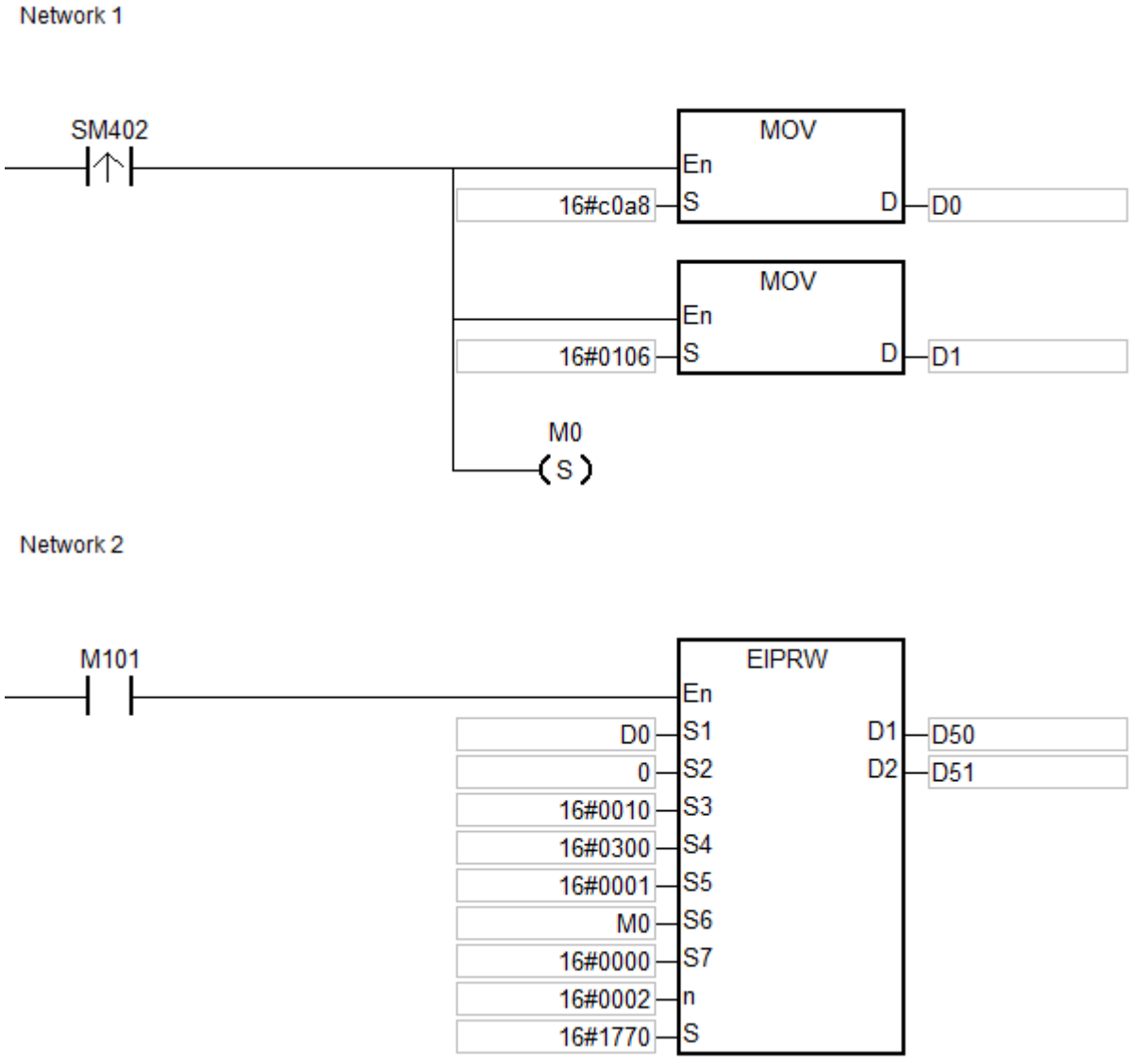
Example 2

The maximum frequency (01-00) of the AC motor drive (at 192.168.1.6) is set to 60.00Hz. The EtherNet/IP Object parameters are set as shown below.

(1) Class ID = 16#0300

(2) Instance ID = 16#0001

(3) Attribute ID = 16#0000



Additional remarks

1. If you declare the operand S_1 in ISPSoft, the data type is ARRAY [2] of WORD.

2. If you declare the operand **S** in ISPSoft, the data type is ARRAY [**n**] of WORD, where **n** is the size of the read/written data.

3. If you declare the operand **D_2** in ISPSoft, the data type is ARRAY [2] of WORD.

4. The following table explains the errors in the execution of the instruction.

Error Code	Error Flag	Description
16#2003	SM0 / **D_1**	1. Value in **S_1**, **S_2**, **S_3**, or **S_6** exceeds the range 2. **S**+n out of range of the device address
16#200B	SM0	Operand **n** out of range
16#600D	SM1100	Ethernet network not connected
16#6301	**D_1**	Connection with remote device broken
16#6302	**D_1**	Remote device response timeout
16#6303	**D_1**	Illegal IP address
16#6304	**D_1**	Error in response command service code
16#6305	**D_1**	Error in response command length
16#6306	**D_1**	Communication conflicts

5. The following table explains the status codes in D2 [0].

Status Code	Description	What to Do
16#00	Connection successful	
16#01	Connection error	Ensure the EDS file of the slave is correct.
16#02	Connected devices unavailable	1. Ensure the number of devices connected to the master does not exceed the limit. 2. Ensure the number of devices connected to the slave does not exceed the limit.
16#03	Error in parameter	Ensure the read/written data in **S** is correct.
16#04	Path error	Ensure the settings are correct for Class ID (**S_4**), Instance ID (**S_5**), and Attribute ID (**S_7**).
16#05	Destination path does not exist	Ensure the settings are correct for Class ID (**S_4**), Instance ID (**S_5**), and Attribute ID (**S_7**).

Status Code	Description	What to Do
16#07	Connection broken	1. Ensure the Ethernet port of the slave device is connected properly. 2. Ensure the keep alive timer setting is correct for the slave device.
16#08	Service code not supported	Ensure the function code (**S_3**) is correct.
16#09	Invalid attribute value	Ensure both the registers involved in reading/writing data and their contents are correct.
16#0A	Error in attribute list	Ensure the slave device object attribute allows PLC to perform the Get_Attribute_List and Set_Attribute_List functions.
16#0B	Transmission conflicts	Ensure the service setting is repeated.
16#0C	Object status conflicts	Ensure the Owner IO connection is established.
16#0D	Object existed	Ensure the slave supports the defined object. The service does not need to execute if the defined object is supported.
16#0E	Attribute not writable	Ensure the object attribute supports the write function.
16#0F	No privilege to perform service code	Ensure the slave device is allowed to perform the service code.
16#10	Device cannot currently perform service	Ensure the Owner IO connection is established.
16#11	Size of response data too large	Ensure the length of data in the object attribute does not exceed the limit (100 words).
16#12	Data access sequence error when tag is accessed	Ensure the data length and the data type are correct.
16#13	Transmitted data too short	Ensure the length (n) of read/written data is correct.
16#14	Attribute value unsupported	Ensure the Attribute ID switch (**S_6**) and Attribute ID (**S_7**) are correct.
16#15	Transmitted data are too long	Ensure the read/written data length (n) is correct.
16#16	Object does not exist	Ensure the Class ID (**S_4**) is correct.

Status Code	Description	What to Do
16#17	Data access sequence error when tag is accessed	1. Ensure the Ethernet network connection is correct. 2. Ensure no packets are lost in the Ethernet communication.
16#18	Attribute value not stored	Ensure no slave device state error is occurring.
16#19	Attribute value storage error	Ensure no slave device hardware error is occurring.
16#1A	Router error: length of request packet exceeds limit	Ensure the read/written data length (n) does not exceed the limit of the router.
16#1B	Router error: length of response packet exceeds limit	Ensure the read/written data length (n) does not exceed the limit of the router.
16#1F	User-defined object access error	Refer to the definition of the slave device error.
16#20	Illegal parameter value	Ensure the read/written value in **S** is correct.

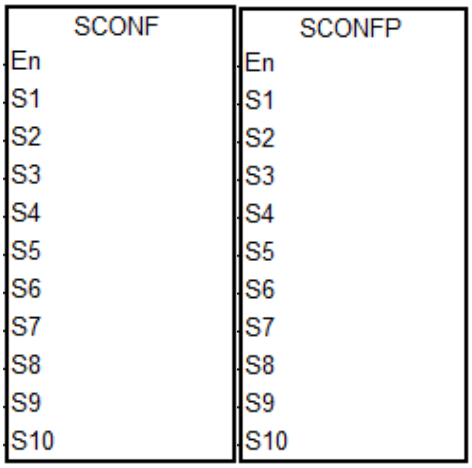
API	Instruction code			Operand	Function
2209		SCONF	P	S_1-S_{10}	Setting the TCP/UDP Socket parameters

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7								●	●				○	○		
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1-S_{10}	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1-S_{10} : See Explanation below

Explanation

1. This instruction sets the TCP/UDP Socket parameters. The following table lists the names and explanations of S_1–S_{10}.

Operand	Name	Description	Data Type	Remarks
S_1	TCP/UDP selection	0: TCP, 1: UDP	WORD	
S_2	Socket number	Range: 1–4	WORD	

Operand	Name	Description	Data Type	Remarks
S3	Remote IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example, if the IP address is 192.168.1.5, **S3**=16#C0A8 and **S3**+1=16#0105.	WORD[2]	Occupies two consecutive words
S4	Remote port	Range: 0–65535; 0 indicates any port	WORD	
S5	Local port	Range: 0–65535; 0 indicates any port	WORD	
S6	Register where sent data comes from	Specifies the number of a D device. Range: 0–29999. Example: The setting value 100 means **S7** bytes of data are sent from the registers starting at D100 (from the low byte to the high byte).	WORD	
S7	Size of sent data	Maximum: 512 bytes	WORD	
S8	Register where received data is stored	Specifies the number of a D device. Range: 0–29999. Example: The setting value 200 means **S9** bytes of data are received and stored in the registers starting at D200 (from the low byte to the high byte).	WORD	
S9	Size of received data	Maximum: 512 bytes	WORD	
S10	Connection time	Range: 1–30000, unit: second	WORD	Applicable to TCP mode only

2. In general, the SCONFP pulse instruction is used.

3. The Socket parameters uses the setting values in HWCONFIG by default. Use the SCONFP instruction when the parameters need to be changed during communication.

4. If you set the parameters while the socket is communicating, the settings do not take effect until the communication is complete. The setting should be changed after the socket is no longer in use.

5. If any one of the parameters is out of range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

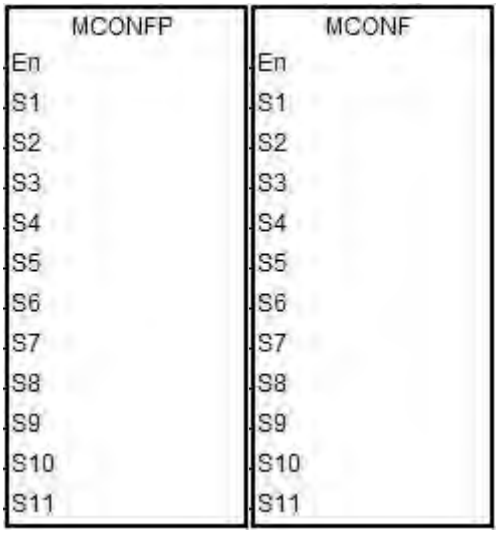
API	Instruction code			Operand	Function
2210		MCONF	P	**S_1-S_{11}**	Reading/Writing Modbus TCP data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●							
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
S_7			●					●								
S_8								●	●				○	○		
S_9								●	●				○	○		
S_{10}								●	●				○	○		
S_{11}			●					●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1-S_{11}	Refer to Explanation in the instruction.												

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S_1-S_{11} : See Explanation below

Explanation

1. This instruction reads and writes Modbus TCP data. The following table lists the names and explanations of **S_1–S_{11}**.

Operand	Name	Description	Data Type	Remarks
S_1	Data Exchange Table No.	Modbus TCP data exchange table number, range: 1–32	WORD	
S_2	Remote IP address	The first two sections of the IP address occupy the first word and the remaining two sections of the IP address occupy the second word. For example, if the IP address is 192.168.1.5, **S_2**=16#C0A8 and **S_2**+1=16#0105.	WORD[2]	Occupies two consecutive words
S_3	Station address of remote slave	Range: 0–255. If the setting exceeds the range, the value in the low byte is used automatically.	WORD	
S_4	Read function code	Range: 16#01, 02, 03, 04, and 17. If the function code is out of range, incoming parameters, including address and data length, are not modified.	WORD	
S_5	Read remote communication address	16#0000–16#FFFF	WORD	
S_6	Read data length	Function code 16#01-02 for reading Bit device, ranging from 1 to 256 Function code 16#03-04 for reading Word device, ranging from 1 to 100 When the setting value is 0 or less than 0, this function is not enabled. If the setting value is bigger than the maximum value, it will be seen as the maximum value.	WORD	

Operand	Name	Description	Data Type	Remarks
S_7	Local register where received data is stored	Function code for reading data in Bit device; M device only Function code for reading data in Word device; D device only	BOOL WORD	
S_8	Write function code	Range: 16#05, 06, 0F, and 10. If the function code is out of range, outgoing parameters, including address and data length, are not modified.	WORD	
S_9	Write remote communication address	Range:16#0000–16#FFFF	WORD	
S_{10}	Write data length	Function code 16#0F for writing Bit device, ranging from 1 to 256; the value in function code 16#05 can only be 1 or 0. Function code 16#10 for writing Word device, ranging from 1 to 100; the value in function code 16#06 can only be 1 or 0. When the setting value is 0 or less than 0, this function is not enabled. If the setting value is bigger than the maximum value, it will be seen as the maximum value.	WORD	
S_{11}	Local register where remote data is written	Function code for writing data in Bit device; M device only Function code for writing data in Word device; D device only	BOOL WORD	

2. In general, use the MCONFP pulse instruction.

3. The following points explain the Modbus communication function codes in **S_4** and **S_8**.

   When the ES3 Series PLC reads the data from multiple bit devices (not discrete input devices), the function code sent is 1 (16#01).

When ES3 Series PLC reads the data from multiple bit devices (only discrete input devices), the function code sent is 2 (16#02).

When the ES3 Series PLC reads the data from multiple word devices (not input registers), the function code sent is 3 (16#03).

When the ES3 Series PLC reads the data from multiple word devices (only input registers), the function code sent is 4 (16#04).

When the ES3 Series PLC writes the status to one bit device, the function code sent is 5 (16#05).

When the ES3 Series PLC writes the data to one word device, the function code sent is 6 (16#06).

When the ES3 Series PLC writes the status to multiple bit devices, the function code sent is 15 (16#0F).

When the ES3 Series PLC writes the data to multiple word devices, the function code sent is 16 (16#10).

When the ES3 Series PLC writes the data to multiple word devices, the function code sent is 23 (16#17).

At present, the ES3 Series PLC only supports the function codes listed above.

4. The parameters in the instruction are valid only while PLC is running. When PLC is powered back on after being powered off, it takes the settings from the data exchange table by default. If you need to change some parameters during data exchange, this instruction modifies those parameters.

5. If the specified socket is already communicating, the new parameters become effective after the current communication is complete.

6. When any parameter is not within the valid range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

7. The ES3 Series PLC does not support the Modbus TCP communication instruction or anything similar to the instruction ETHRW DVP series PLC. If you use the PLC program for Ethernet Modbus TCP communication control, enable “Program control” mode in the data exchange table first and use the SM numbers from the following table.

SM Number	Attribute	Explanation of Ethernet Data Exchange Parameters
SM1167	R/W	Flag to enable data exchange
SM1168–SM1199	R/W	Flag to enable data exchange connections 1–32
SM1200–SM1231	R	Success flag for data exchange connections 1–32
SM1232–SM1263	R	Error flag for data exchange connections 1–32

When data is received successfully, the success flag is ON. When there is an error receiving data, the error flag is ON. The success flag and the error flag of any one connection socket are never switched to ON at the same time.

8. The following table explains the (read-only) SR numbers used in data exchange.

SR Number	Explanation
SR1120–SR1151	Indicates the actual communication times for connections 1–32.
SR1152–SR1183	Indicate the communication error codes for connections 1–32.

The following table defines the error codes in SR1152–SR1183.

Error Code	Explanation
16#00XX	Remote device response error
16#F000	Ethernet network not connected
16#F001	Remote device response timeout
16#F003	TCP connection timeout
16#F007	Response command error
16#F009	Connection with remote device disconnected

API	Instruction			Operand	Description
2211		EMCONF1	P	**Server ~ Passw**	Setting email server parameter values

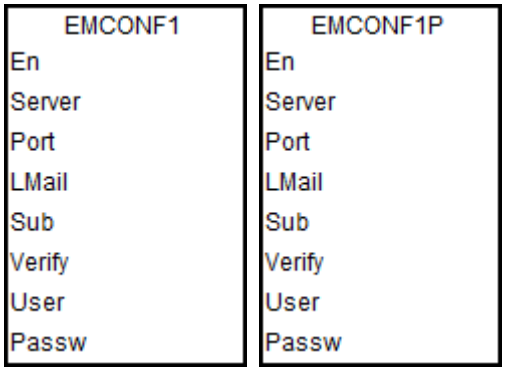
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Server								●	●						○	
Port								●	●				○	○		
LMail								●	●						○	
Sub								●	●						○	
Verify								●	●				○	○		
User								●	●						○	
Passw								●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
	Refer to following instruction explanation												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol

Server~Passw : Refer to following explanation



6

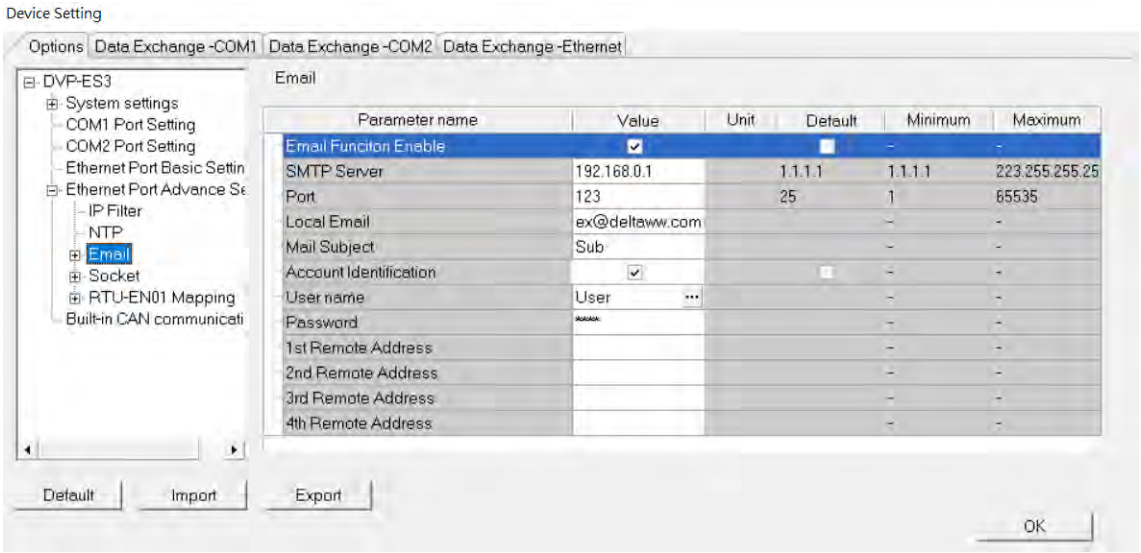
Explanation

1. See the operand name, use explanation and data type in the following table.

Operand	Name	Explanation	Data type	Remark
Server	Email server	**Server** +0: Data length, unit: bytes (0 or 4; 0: no modification) **Server** +1~ **Server** +2: data (Data are placed from the lower 8 bits.) For example: 192.168.0.1, D0 = 16#0004, D1= 16#0001, D2= 16#C0A8;	WORD[N]	Refer to **Server** +0 for address occupation. 3 consecutive words at most can be occupied.
		Up to 15 words can be used in one string; every section in one string can contain 1-3 words, and use a dot "." to separate each string, for example 192.168.0.1	String	
Port	Communication port	Range: 1~65535, 0: no modification	WORD	
LMail	Local email address	**LMail** +0: data length, unit: bytes (range: 0~63, 0: no modification.) **LMail** +1~ **LMail** +32: data (Data are placed from the lower 8 bits.) For example: ex@deltaww.com D0 = 16#000E, D1 = 16#7865, D2 = 16#6440, D3 = 16#6C65, D4 = 16#6174, D5 = 16#7777, D6 = 16#2E63, D7 = 16#6D6F	WORD[N]	Refer to **LMail** +0 for address occupation. 33 consecutive words at most can be occupied.
		Up to 63 words can be used in one string	String	
Sub	Email subject	**Sub** +0: data length, unit: bytes (range: 0~31, 0: no modification.) **Sub** +1~ **Sub** +16: data content (Data are placed from the lower 8 bits.) For example: Sub 0 = 16#0003, D1 = 16#7553, D2 = 16#0062	WORD[N]	Refer to **Sub** +0 for address occupation. 17 consecutive words at most can be occupied.
		Up to 31 words can be used in one string	String	
Verify	Enable/ disable verification	0: no modification, 1: Disable verification, 2: Enable verification	WORD	

Operand	Name	Explanation	Data type	Remark
User	User name	**User** +0: data length, unit: bytes (range: 0~31, 0: no modification.) **User** +1~ **User** +16: data content (Data are placed from the lower 8 bits.) For example: User D0 = 16#0004, D1 = 16#7355, D2 = 16#7265	WORD[N]	Refer to **User** +0 for address occupation. 17 consecutive words at most can be occupied.
		Up to 31 words can be used in one string	String	
Passw	Password	**Passw** +0: data length, unit: bytes (range: 0~20, 0: no modification.) **Passw** +1~ **Passw** +10: data content (Data are placed from the lower 8 bits.) For example: Passw D0 = 16#0005，D1 = 16#5061， D2 = 16#7373，D3 = 16#0077	WORD[N]	Refer to **Passw** +0 for address occupation. 11 consecutive words at most can be occupied.
		Up to 20 words can be used in one string	String	

You can set up the email server related settings in ISPSoft > HWCONFIG > Setting > Options > Email. And from here you can also check the parameters which are set through instrucitons. But only the settings that are set in ISPSoft are retainable after power-off.



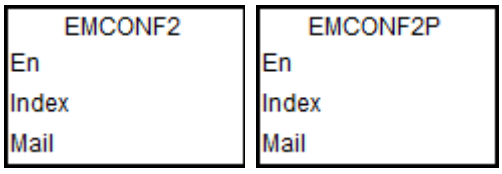
API	Instruction			Operand	Description
2212		EMCONF2	P	**Index，Mail**	Setting email address

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Index								●	●				○	○		
Mail								●	●						●	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
	Refer to following instruction explanation												

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



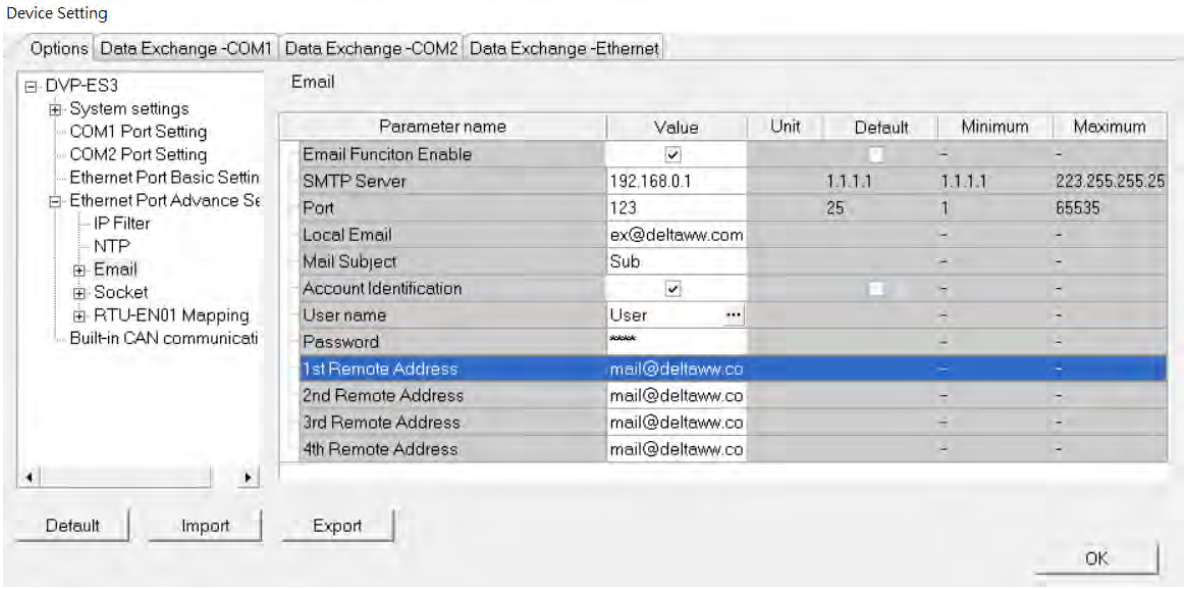
Index, Mail : Refer to following explanation

Explanation

1. See the operand name, use explanation and data type in the following table.

Operand	Name	Explanation	Data type	Remark
Index	Mail address number	**Index**: The mail address number to be modified (Range: 0~4, 0: Not modify the mail address.)	WORD	
Mail	Mail address	**Mail** +0: the data length of **Index** value with the unit of bytes (range: 0~63, 0: Not modify the mail address.) **Mail** +1~ **Mail** +32: The data content of **Index** (Data are placed from the lower 8 bits.) For example: ex@deltaww.com D0 = 16#000E, D1 = 16#7865, D2 = 16#6440, D3 = 16#6C65, D4 = 16#6174, D5 = 16#7777, D6 = 16#2E63, D7 = 16#6D6F	WORD[N]	Refer to **Mail** +0 for address occupation. 33 consecutive words at most can be occupied.
		Up to 63 words can be used in one string	String	

You can set up the email server related settings in ISPSoft > HWCONFIG > Setting > Options > Email. And from here you can also check the parameters which are set through instrucitons. But only the settings that are set in ISPSoft are retainable after power-off.



6.23 Memory Card / File Register Instructions

6.23.1 List of Memory Card / File Register Instructions

The following table lists the Memory Card / File Register instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2300	MWRIT	–	✓	Writing data from the PLC into the memory card
2301	MREAD	–	✓	Reading data from the memory card into the PLC
2302	MTWRIT	–	✓	Writing a string into the memory card
2303	MEMW	–	✓	Writing data into the file register
2304	MDEL	–	✓	Deleting files on the memory card

6.23.2 Explanation of Memory Card / File Register Instructions

API	Instruction code			Operand	Function
2300		MWRIT	P	**C, S, S_1, S_2, S_3, S_4**	Writing data from the PLC into the memory card

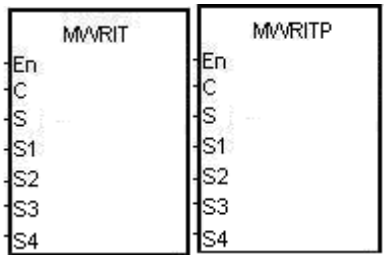
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
C								●	●				○	○		
S								●	●							
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●						○	
S_4								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●											
S		●											
S_1			●										
S_2		●											
S_3													●
S_4			●										

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

6

Symbol



C : Control parameter

S : Data source

S_1 : Data length

S_2 : Line advance

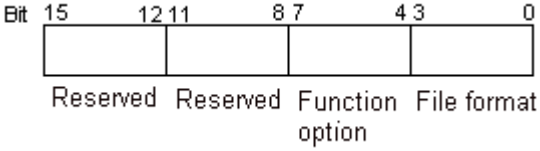
S_3 : File name

S_4 : Data address in the file

Explanation

1. This instruction writes data from the PLC to the memory card. The operands are described as follows.

   - **C**: The control parameter



Item	Code	Description
File format	0	Binary value Default value
		The file name extension is .dmd.
		The unit of the value is a word.
	1	The values are separated by a comma.
		The unit of the value is a word.
		The file name extension is .cvs.
		Use ASCII codes..
		The value that is stored is a hexadecimal format.
	2	The values are separated by a comma.
		The unit of the value is a double word.
		The file name extension is .cvs.
		Use ASCII codes.
		The value that is stored is a hexadecimal format.
	3	The values are separated by a tab.
		The unit of the value is a word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal format.
	4	The values are separated by a tab.
		The unit of the value is a double word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal format.
	5	The values are not separated by any mark.
		The unit of the value is a word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal format.
	6	The values are not separated by any mark.

Item	Code	Description
		The unit of the value is a double word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal format.
	7	The values are separated by a comma.
		The unit of the value is a Word.
		The file name extension is.csv.
		Use ASCII codes.
		The value that is stored is a decimal format.
Function option	0	Appending The data that is written into the memory card is appended after the last value in the file. This is the default value
		If the file does not exist, it is created automatically.
	1	Overwriting The data that is written into the memory card replaces the values in the file starting from the value indicated by the value in **S_4**.
		If the file does not exist, it is created automatically.
Reserved	-	The values of bit 8–bit 15 are 0.

- **S**: The data source
- **S_1**: The length of the data that the instruction writes into the file

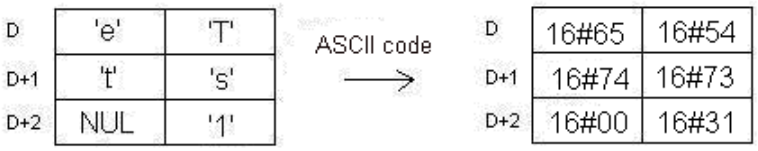
  If the value in **S_1** is 0, the instruction does not write any data into the file.

Item	Description
Value unit	If the file format is 0, 1, 3, 5, or 7, the unit of the value is a word. If the file format is 2, 4, or 6, the unit of the value is a double word.
Parameter unit	Double word
Length of the data	The devices where the instruction stores the data cannot exceed the device range, and the size of the data that the instruction writes into the file cannot be more than four gigabytes. Refer to Chapter 2 for more information about devices.

- **S_2**: The line advance

The value in **S_2** must be between 0–256.

- ES3 series PLC supports using string (less than 9 characters) in the file name **S_3**.
- **S_3**–**S_3**+4: **S_3** occupies five devices. The file name contains a maximum of nine characters, including 16#00. If the string does not end with 16#00, an error occurs. When the instruction reads the ending character, it stops reading, and checks if the file name is legal. The characters in the file name can be A–Z, a–z, and 0–9. The file name extension depends on the file format. The file that the instruction creates is in the default folder. If the file name is “Test1”, the instruction writes the characters into the devices as follows.



- The default folder path:

Model name	Folder path
ES3 Series	PLC CARD\ES3\UserProg

- **S_4**: The value in the file that the instruction overwrites is indicated by the value in **S_4**.

Item	Description
Value unit	If the file format is 0, 1, 3, 5 or 7, the unit of the value is a word. If the file format is 2, 4, or 6, the unit of the value is a double word.
Parameter unit	Double word.
Usage	If the function option is 0, **S_4** is not used.
	If the function option is 1, the instruction replaces the values in the file with the data to be written into the memory card starting from the value indicated by the value in **S_4**.
	The value in **S_4** must indicate a value in the file. If the value in **S_4** is 0, the instruction overwrites the data in the file starting at the first value in the file.

2. Instruction flags:

Flag	Description
SM450	ON: the memory card is in the CPU module.
SM451	State of the write protection switch on the memory card ON: the memory card is write protected. OFF: the memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card to the PLC.
SM453	ON: an error occurs during the operation of the memory card. If the flag is ON, you must reset it to OFF. The error code is stored in SR453.

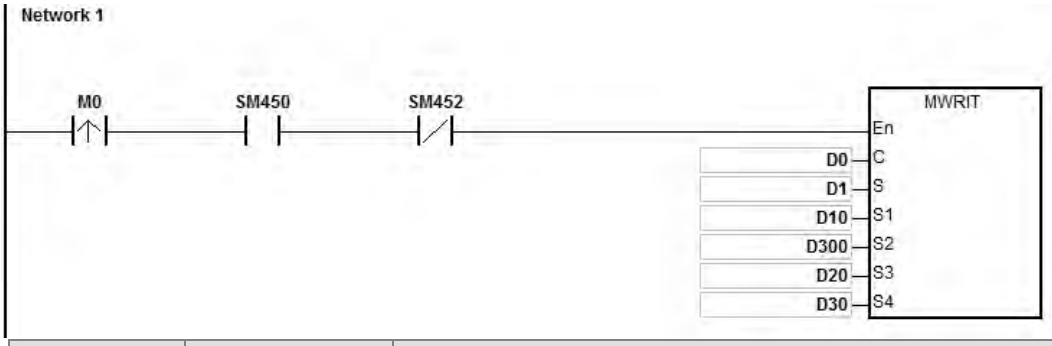
3. Related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.
16#0064	The data cannot be read from the memory card.
16#0065	The file is a read-only file.

4. If the format of the target file on the memory card is 0, the format of the source file in the PLC must also be 0. Otherwise, the instruction cannot read the data, and SM453 is ON. The same applies to the other file formats.

Example

SM450 is ON when you insert the memory card into the CPU module; SM452 is ON when the MWRIT instruction executes; SM452 is OFF when the MWRIT instruction completes. You cannot use MWRITP (the pulse instruction) continuously. Executing this pulse instruction to write data into the memory card continuously may exceed the writing limitation and may lead to memory card damage.



Operand	Setting value	Description
D0	16#0011	The file into which the data is written The file format: The values are separated by a comma. The unit of the value is a word. The file name extension is .cvs. Use ASCII codes.
D1	-	The data that is written into the file
D10, D11	16#00000030	The size of the data that is written into the file is 48 words.
D300	16#000A	Ten values are written into every line.
D20	D20=16#6554 D21=16#7473 D22=16#0031	The file name is “Test1”.
D30, D31	16#00000000	The data that is written into the memory card replaces the values in the file starting from the first value.

Additional remarks

1. If the value in **C** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_1}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_2}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in $\mathbf{S_3}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

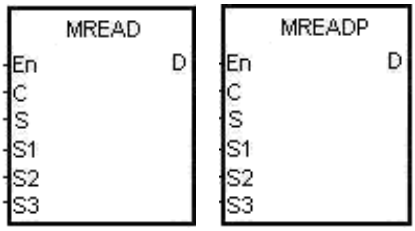
API	Instruction code			Operand	Function
2301		MREAD	P	**C, S, S_1, S_2, S_3, D**	Reading data from the memory card into the PLC

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
C								●	●				○	○		
S								●	●						○	
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
D								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●											
S													●
S_1			●										
S_2		●											
S_3			●										
D		●											

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



C : Control parameter

S : File name

S_1 : Data address in the file

S_2 : Reserved

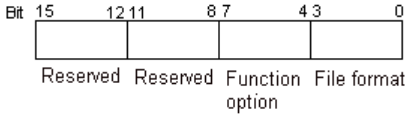
S_3 : Data length

D : Data destination

Explanation

1. This instruction reads data from the memory card into the PLC. The operands are described as follows.

   - **C**: The control parameter



Item	Code	Description
File format	0	Binary value The default value is 0.
		The file name extension is .dmd.
		The unit of the value is a word.
	1	The values are separated by a comma.
		The unit of the value is a word.
		The file name extension is .cvs.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	2	The values are separated by a comma.
		The unit of the value is a double word.
		The file name extension is .cvs.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	3	The values are separated by a tab.
		The unit of the value is a word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	4	The values are separated by a tab.
		The unit of the value is a double word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	5	The values are not separated by any mark.
		The unit of the value is a word.
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	6	The values are not separated by any mark.
		The unit of the value is a double word.

Item	Code	Description
		The file name extension is .txt.
		Use ASCII codes.
		The value that is stored is a hexadecimal value.
	7	The values are separated by a comma.
		The unit of the value is a Word.
		The file name extension is.csv.
		Use ASCII codes.
		The value that is stored is a decimal format.
Function option	0	The values in the file starting from the value indicated by the value in **S_1** are read. The default value is 0.
	1	The number of values is stored in **D** and **D**+1.
		If the file format is 0, 1, 3, or 5, the unit of the value is a word. If the file format is 2, 4, or 6, the unit of the value is a double word.
Reserved	-	The values of bit 8–bit 15 are 0.

- ES3 series PLC supports using string (less than 9 characters) in the file name **S**.

- **S**–**S**+4: **S** occupies five devices. The file name contains a maximum of nine characters, including 16#00. If the string does not end with 16#00, an error occurs. When the instruction reads the ending character, the instruction stops reading characters, and checks if the file name is legal. The characters in a file name can be A–Z, a–z, and 0–9. The file name extension depends on the file format. The file that the instruction creates is in the default folder. If the file name is “Test1”, the instruction writes the characters into the devices as follows.

- The default folder path:

Model name	Folder path
ES3 Series	PLC CARD\ES3\UserProg

- **S_1**: The value in the file that the instruction reads is indicated by the value in **S_1**.

Item	Description
Value unit	Refer to operand C for its file format. If the file format is 0, 1, 3, 5 or 7, the unit of the value is a word. If the file format is 2, 4, or 6, the unit of the value is a double word.
Parameter unit	The parameter unit is a double word.
Usage	The value in **S_1** must indicate a value in the file. If the value in **S_1** is 0, the instruction reads from the first value in the file.

- **S_3**: The length of the data that the instruction reads from the file

  The devices in which the instruction stores the data cannot exceed the device range. If the value in **S_3** is larger than the number of values in the file, the length of the data read from the file is the number of values in the file. The unit **S_3** is a double word.

- **D**: The initial device where the data is stored.

2. Instruction flags:

Flag	Description
SM450	ON: the memory card is in the CPU module
SM451	The write protection switch on the memory card ON: the memory card is write protected. OFF: the memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card into the PLC.
SM453	ON: an error occurs during the operation of the memory card. If the flag is ON, you must reset it to OFF. The error code is stored in SR453.

3. Related error codes (SR453):

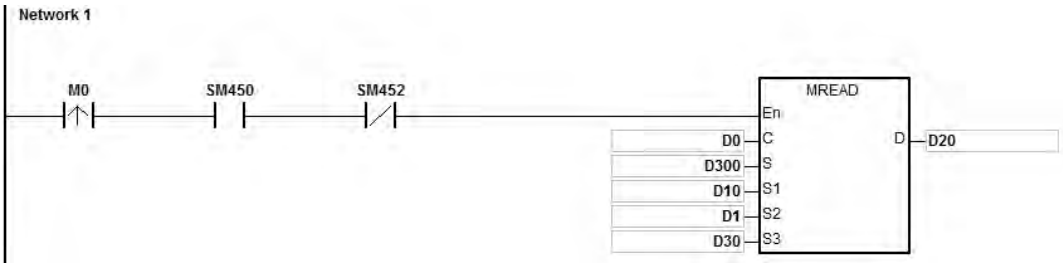
Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.

Error code	Description
16#0064	The data cannot be read from the memory card.

4. If the format of the target file into which the instruction writes data is 0, the format of the file from which the instruction reads data must also be 0. Otherwise, the instruction cannot read the data, and SM453 is ON. The same applies to the other file formats.

Example

SM450 is ON when the memory card is inserted into the CPU module; SM452 is ON when the MREAD instruction executes; SM452 is OFF when the MREAD instruction completes.



Operand	Setting value	Description
D0	16#0011	The file from which the data is read The file format: The values are separated by a comma. The unit of the value is a word. The file name extension is .csv. Use ASCII codes.
D300	D300=16#6554 D301=16#7473 D302=16#0031	The file name is “Test1”.
D10, D11	16#00000000	The values in the file starting from the first value are read.
D1	16#000A	Ten values are read from every line.
D30, D31	16#00000020	The size of the data that is read from the file is 32 words.
D20	-	The data that is read is stored in D20.

Additional remarks

1. If the value in **C** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_2}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_3}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

4. If the value in **D** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

API	Instruction code	Operand	Function

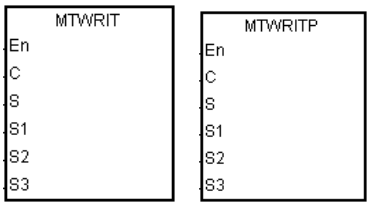
2302		MTWRIT	P	**C, S, S_1, S_2, S_3**	Writing a string into the memory card

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
C								●	●				○	○		
S								●	●							
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●											
S		●											
S_1		●											
S_2		●											
S_3													●

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



C : Control parameter

S : Data source

S_1 : Data length

S_2 : Separation mark

S_3 : File name

Explanation

1. This instruction writes a string into the memory card. The operands are described as follows.

   - **C**: The control parameter

Parameter value	**Description**
0 (Appending)	If the file exists, the data that is written into the memory card is appended after the last byte in the file.
	If the file does not exist, it is created automatically.
1	If the file exists, the new data that is written to the memory card replaces the old data in the

(Overwriting)	file. The size of the file is the size of the new data.
	If the file does not exist, it is created automatically.

- **S**: The data source

  If the string that the instruction writes into the file is “12345”, the instruction stores the characters in the devices as follows. The basic unit is a byte, and so the instruction stores the first character in the low byte in D300. The instruction stores the second character in the high byte in D300. The same applies to other characters. The instruction stores “16#00” in the high byte in D300+2, and indicates the end of the string.

S300		S300+1		S300+2	
byte 2	byte 1	byte 4	byte 3	byte 6	byte 5
16#32	16#31	16#34	16#33	16#00	16#35

- **S_1**: The length of the data that is written into the memory card

  The basic unit is a byte. The devices where the instruction stores the data cannot exceed the device range, and the length of the data that the instruction writes into the memory card cannot be more than 255 bytes.

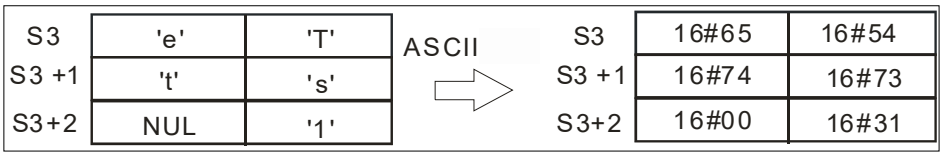
- **S_2**: The separation mark

  If the value in **S_1** is N, the instruction writes the value in **S_2** into the memory card as follows.

S_2Operand		**Description**
High byte	**Low byte**	
16#00	16#00 or not 16#00	The N-byte data is written into the file.
Not 16#00	16#00	The N+1-byte data is written into the file. The value in the high byte in **S_2** is the value in the N+1st byte.
Not 16#00	Not 16#00	The N+2-byte data is written into the file. The value in the high byte in **S_2** is the value in the seventh byte, and the value in low byte in **S_2** is the value in the N+2th byte.

- ES3 series PLC supports using string (less than 9 characters) in the file name **S_3**.

- **S_3–S_3+4**: **S_3** occupies five devices. The file name contains a maximum of nine characters, including 16#00. If the string does not end with 16#00, an error occurs. When the instruction reads the ending character, the instruction stops reading characters, and checks if the file name is legal. The characters in the file name can be A–Z, a–z, and 0–9. The file name extension depends on the file format. The file that the instruction creates is in the default folder. If the file name is “Test1”, the instruction writes characters into the devices as follows.



- The default folder path:

Model name	Folder path
ES3 Series	PLC CARD\ES3\UserProg

2. Instruction flags

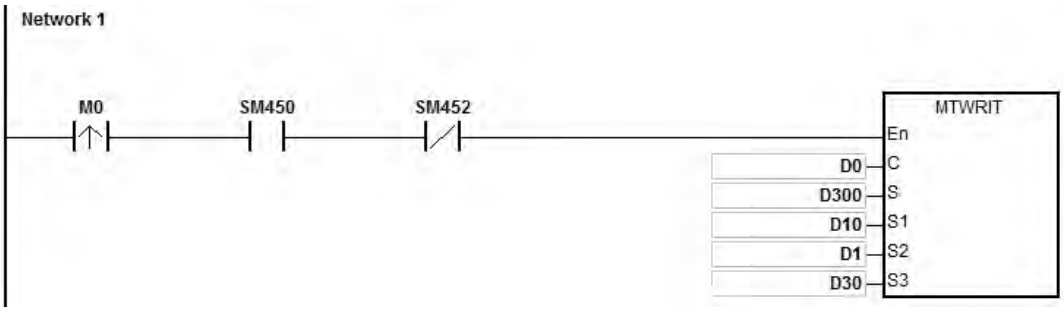
Flag	Description
SM450	ON: the memory card is in the CPU module.
SM451	The write protection switch on the memory card ON: the memory card is write protected. OFF: the memory card is not write protected.
SM452	The data is being written from the PLC into the memory card, or the data is being read from the memory card into the PLC.
SM453	ON: an error occurs during the operation of the memory card. If the flag is ON, you must reset it to OFF. The error code is stored in SR453.

3. Related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.
16#0064	The data cannot be read from the memory card.
16#0065	The file is a read-only file.

Example:

SM450 is ON when the memory card is inserted into the CPU module; SM452 is ON when the MTWRIT instruction executes; SM452 is OFF when the MTWRIT instruction completes.



Operand	Setting value	Description
D0	16#0001	The file into which the data is written The file format: The unit of the character is a byte. The file name extension is .txt. Use ASCII codes. The data in D300 is written to the file.
D300	-	The data that is written to the file
D10	16#000A	The size of the string that is written to the file is 10 bytes.
D1	16#0A00	After the data is written to the file, the separation mark is appended after the last byte in the file.
D30	D30=16#6554 D31=16#7473 D32=16#0031	The file name is “Test1”.

Additional remarks

1. If the value in **C** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in $\mathbf{S_1}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3. If the value in $\mathbf{S_3}$ exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

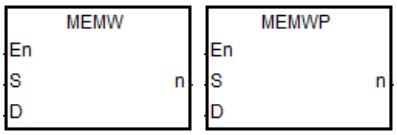
API	Instruction code			Operand	Function
2303		MEMW	P	**S, D, n**	Writing data into the file register

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S					●	●		●								
D									●							
n					●	●		●				●	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●											
D		●											
n		●											

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : The first address of the data source

D : The first address where the data is stored

n : Data length

Explanation

1. This instruction writes data into the file register. The description of the operands is as follows.
   - **S**: The initial address of the data source; it is suggested that you declare an array type variable.
   - **D**: The initial address where the instruction stores data. It is suggested that you assign an address for the file register and declare an array type variable.
   - **n**: The length of the data that the instruction writes into the file register, between 1–2048. If the value exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
   - If the device **S** or **D** exceeds the allowed range, SM0 is ON, and the error code in SR0 is 16#2003.
2. Since it takes 60–120 ms for the instruction to write the data, it is suggested that you use this instruction when the PLC is idle. For example, when there is no external interrupt task, no high-speed output, or any immediate events for the PLC to process.
3. The instruction writes only when the contact switches from OFF to ON and writes only once.
   NOTE: You can write into this file register a maximum of 100,000 times.

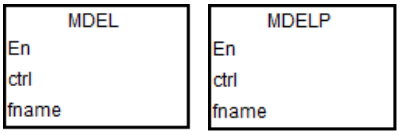
API	Instruction code			Operand	Function
2304		MDEL	P	**ctrl, fname**	Deleting files on the memory card

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
ctrl								●					○	○		
fname								●	●						○	

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
ctrl		●			●	●							
fname													●

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



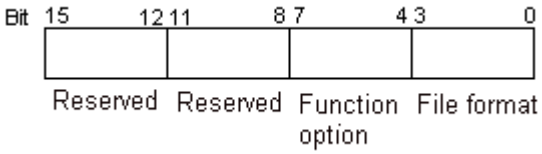
ctrl : Control parameter

fname : Data source

Explanation

1. This instruction writes data from the PLC to the memory card. The operands are described as follows.

   - **ctrl**: The control parameter



Item	Code	Description
File format	0	The file name extension is .dmd.
	1, 2, 7	The file name extension is .cvs.
	3, 4, 5, 6	The file name extension is .txt.
Reserved	-	The values of bit 8–bit 15 are 0.

   - You can use strings (less than 9 characters) in the in **fname** (file name).
   - **fname** occupies five devices. The file name contains a maximum of nine characters, including 16#00. If the string does not end with 16#00, an error occurs. When the instruction reads the ending character, it stops

reading, and checks if the file name is legal. The characters in the file name can be A–Z, a–z, and 0–9. The file name extension depends on the file format. The file that the instruction creates is in the default folder. If the file name is “Test1”, the instruction writes the characters into the devices as follows.

- The default folder path:

Model name	Folder path
ES3 Series	PLC CARD\ES3\UserProg

2. Instruction flags:

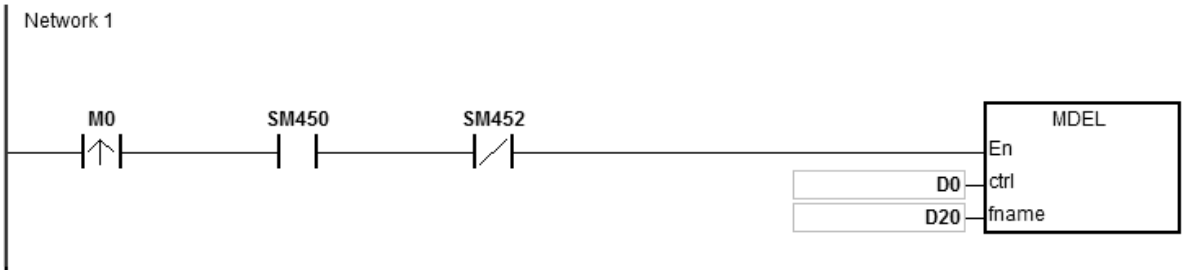
Flag	Description
SM450	ON: the memory card is in the CPU module.
SM451	State of the write protection switch on the memory card ON: the memory card is write protected. OFF: the memory card is not write protected.
SM452	The data is being written from the PLC to the memory card, or the data is being read from the memory card to the PLC.
SM453	ON: an error occurs during the operation of the memory card. If the flag is ON, you must reset it to OFF. The error code is stored in SR453.

3. Related error codes (SR453):

Error code	Description
16#005E	An error occurs when the memory card is initialized.
16#005F	The path is incorrect, or the file does not exist.
16#0060	The default folder cannot be created.
16#0061	The memory space is insufficient.
16#0062	The memory card is write protected.
16#0063	An error occurs when the data is written into the file.

Example

SM450 is ON when you insert the memory card into the CPU module; SM452 is ON when this instruction executes; SM452 is OFF when this instruction completes.



Operand	Setting value	Description
D0	16#0001	The file name extension is .cvs.
D20	D20=16#6554 D21=16#7473 D22=16#0031	The file name is “Test1”.

Additional remarks

1. If the value in **ctrl** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

2. If the value in **fname** exceeds the range, an operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6.24 Task Control Instructions

6.24.1 List of Task Control Instructions

The following table lists the Task Control instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2400	TKON	–	✓	Enabling a cyclic task
2401	TKOFF	–	✓	Disabling a cyclic task

6.24.2 Explanation of Task Control Instructions

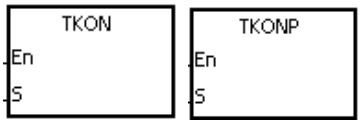
API	Instruction code			Operand	Function
2400		TKON	P	**S**	Enabling a cyclic task

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●	●		○	○	○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Task number

Explanation

1. This instruction enables the cyclic task specified by **S**.
2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSoft.
3. The operand **S** must be between 0-31.
4. Refer to the ISPSoft User Manual for more information about creating and enabling tasks.

Example

When the PLC runs, cyclic task (0) is enabled. When the TKON instruction in cyclic task (0) executes, it enables cyclic task (1), and M10 is ON.

You create cyclic tasks in ISPSoft, and you define their default enabled/disabled state in ISPSoft. Cyclic task (0) is enabled when the PLC runs, and cyclic task (1) is not enabled when the PLC runs.

Cyclic task (1) is enabled by the TKON instruction in cyclic task (0).

Cyclic task (1) is executed.

Additional remarks

Refer to the ISPSoft User Manual for more information on tasks.

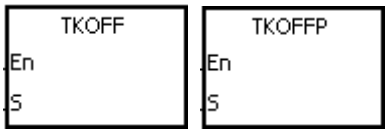
API	Instruction code			Operand	Function
2401		TKOFF	P	**S**	Disabling a cyclic task

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●	●		○	○	○			

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
ES3	ES3	-

Symbol



S : Task number

Explanation

1. This instruction disables the cyclic task specified by **S**.
2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSoft.
3. The operand **S** must be between 0-31.
4. Refer to the ISPSoft User Manual for more information about creating and enabling tasks.

Example

When the PLC runs, cyclic task (0) and cyclic task (1) are enabled. When the TKOFF instruction in cyclic task (0) executes, it disables cyclic task (1), and M10 is OFF.

You create cyclic tasks in ISPSoft and you define their default enabled/disabled state in ISPSoft. Cyclic task (0) and cyclic task (1) are enabled when the PLC runs, and cyclic task (1) is disabled when the TKOFF instruction in cyclic task (0) executes. Cyclic task (1) is disabled by the execution of the TKOFF instruction in cyclic task (0).

Cyclic task (1) is not executed.

Additional remarks

Refer to the ISPSoft User Manual for more information on tasks.

6.25 SFC Instructions

6.25.1 List of SFC Instructions

The following table lists the SFC (Sequential Function chart) instructions covered in this section.

API	Instruction code		Pulse instruction	Function
	16-bit	32-bit		
2500	SFCRUN	–	–	SFC Run
2501	SFCPSE	–	–	SFC Pause
2502	SFCSTP	–	–	SFC Stop

6.25.2 Explanation of Task Control Instructions

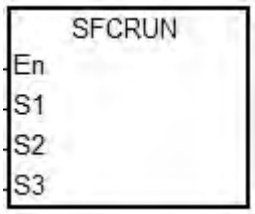
API	Instruction code	Operand	Function
2500	SFCRUN	**S_1, S_2, S_3**	SFC Run

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1																
S_2								●					○	○		
S_3																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2		●			●	●							
S_3													

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Name of the SFC POU

S_2 : Function code

S_3 : Device address

Explanation

1. This instruction runs the SFC (Sequential Function Chart) program specified by **S_1** according to the function code in **S_2**.

2. The instruction runs the SFC POU (Program Organization Unit) specified by **S_1** only when the PLC is scanning the SFC POU.

3. The operands are described as follows.

   - **S_1** defines the name of the SFC POU.
   - When the designated SFC POU of **S_1** executes, the instruction clears the parameters (such as SFC/STEP/ACTION/TRANSITION) for the SFC program when **S_2**=0 or 1, and the SFC execution starts according to the value specified in **S_2**.
   - IF **S_2**=0, the system executes the SFC POU from the initial step.
   - IF **S_2**=1, the system executes the SFC POU from the designated step in **S_3**.

- If **S_2**=2, the instruction does NOT clear the status and the parameters (such as SFC/STEP/ACTION/TRANSITION) for the SFC and the system starts executing from where it paused.
- **S_3** designates where to start (which step) in the SFC program in **S_1**.

4. The range of **S_2** is between 0–2. If it is out of range, it is evaluated as 0.
5. When the state of the SFC POU is RUN, executing this instruction is invalid.

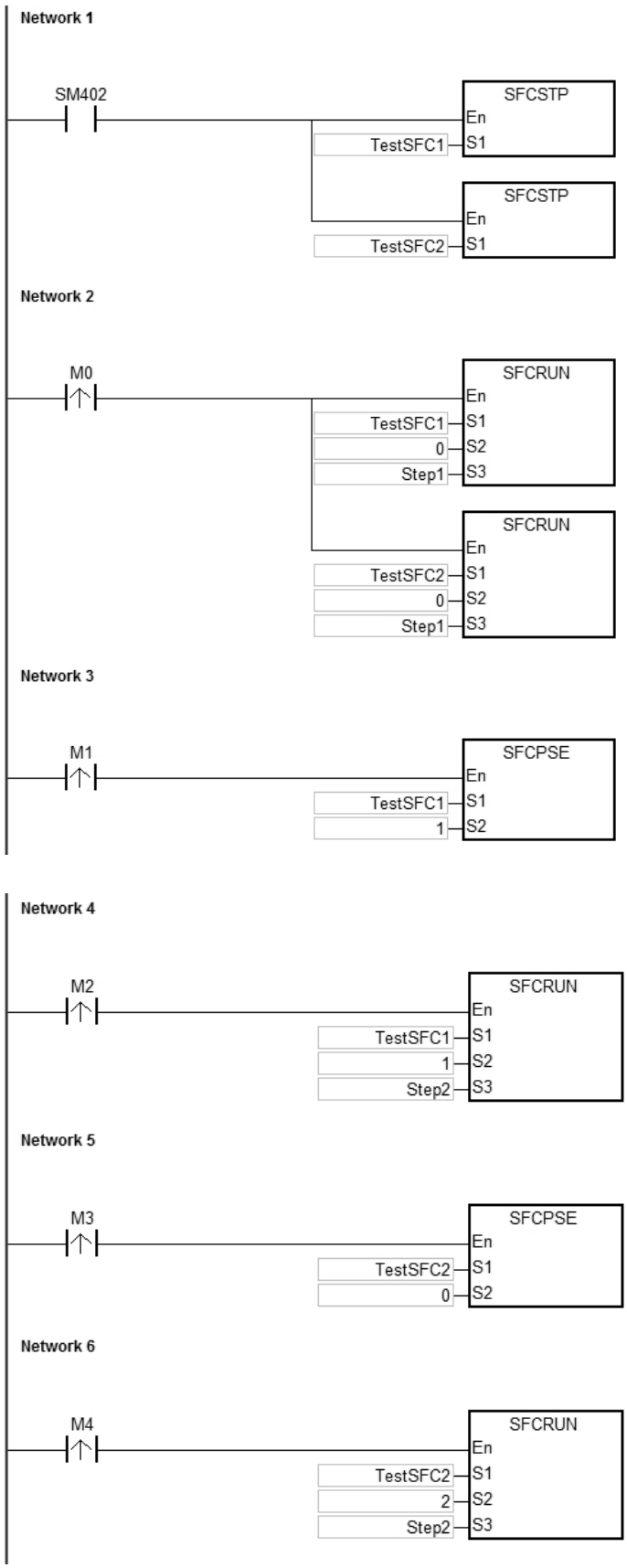
Example

Set up one LD (ladder) POU and specify its POU name as “Main”, and then set up two SFC POUs with the names of “TestSFC1” and “TestSFC2”.

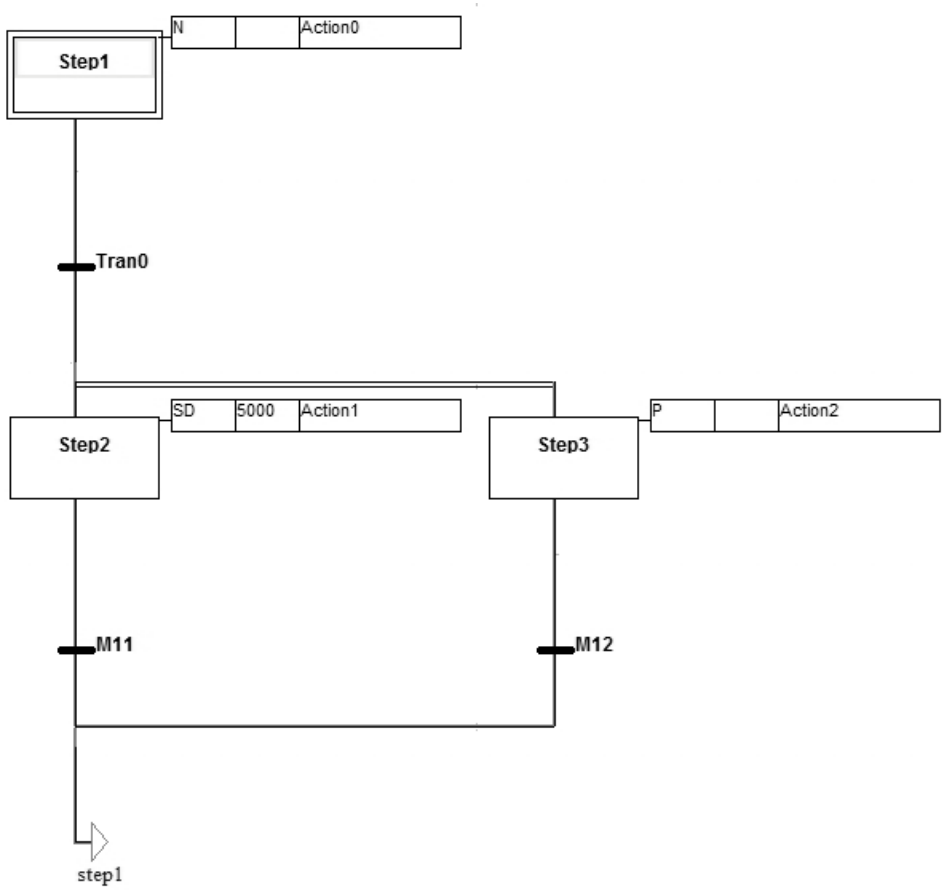
1. When the PLC executes the Main program (RUN), TestSFC1 and TestSFC2 execute the SFCSTP instruction (API 2502), and the two SFC POUs stop executing.
2. When M0 switches from OFF to ON, TestSFC1/ TestSFC2 POU execute the SFCRUN* instruction. See below for the contents of the TestSFC1 and TestSFC2 for execution details. When **S_2**=0, the SFC status and parameters are cleared and begin to execute from STEP 1. When **S_2**=1, the SFC status and parameters are cleared and begin to execute from the designated STEP in **S_3**.
3. When M1 switches from OFF to ON, TestSFC1 POU pauses. When **S_2**=1, all the SFC executing actions and outputs are cleared, and the system runs the final scan.
4. When M2 switches from OFF to ON, TestSFC1 POU executes. When **S_2**=1, the SFT status and parameters are cleared, and the system begins to execute from STEP 2.
5. When M3 switches from OFF to ON, TestSFC2 POU pauses. When **S_2**=0, all the SFC executing actions and outputs are kept, and the system does not run the final scan.
6. When M4 switches from OFF to ON, TestSFC1 POU executes. When **S_2**=2, the SFC status and parameters are and the SFC begins to execute from where it paused.

*SFCRUN activates the SPC POU at the next scan.

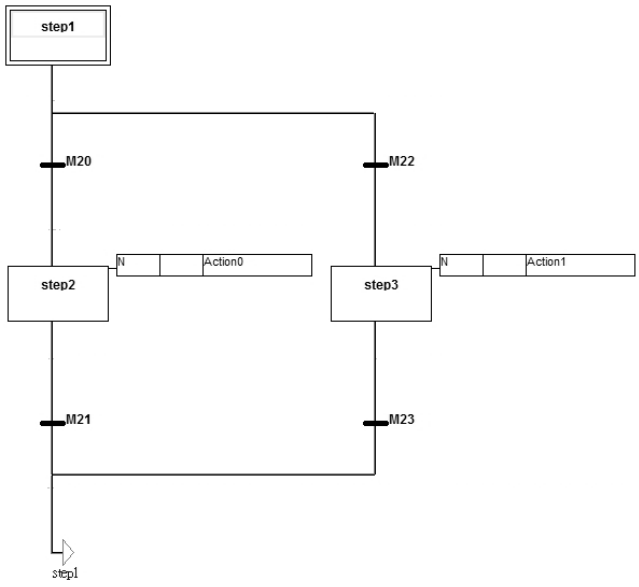
Main POU


Network 1
SM402
SFCSTP
En
TestSFC1
S1
SFCSTP
En
TestSFC2
S1
Network 2
M0
SFCRUN
En
TestSFC1
S1
0
S2
Step1
S3
SFCRUN
En
TestSFC2
S1
0
S2
Step1
S3
Network 3
M1
SFCPSE
En
TestSFC1
S1
1
S2
Network 4
M2
SFCRUN
En
TestSFC1
S1
1
S2
Step2
S3
Network 5
M3
SFCPSE
En
TestSFC2
S1
0
S2
Network 6
M4
SFCRUN
En
TestSFC2
S1
2
S2
Step2
S3

TestSFC1 POU



TestSFC2 POU



Additional remarks

Refer to the ISPSoft User Manual for more information on Sequential Function Charts.

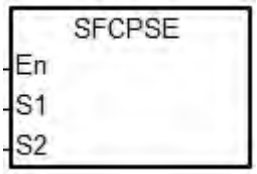
API	Instruction code	Operand	Function
2501	SFCPSE	S_1 · S_2	SFC Pause

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1																
S_2								●					○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S_1 : Name of the SFC POU

S_2 : Function code

Explanation

1. This instruction pauses the SFC POU specified by S_1 according to the function code in S_2.
2. The instruction can pause the SFC POU specified by S_1 only when the PLC is scanning the SFC POU.
3. When paused, the SFC status and parameters (such as SFC/STEP/ACTION/TRANSITION) are stored.
4. The operands are described as follows.
   - S_1 defines the name of the SFC POU.
   - When S_2=0, the instruction preserves all the executing actions of the SFC and the outputs, and the system does not run the final scan.
   - When S_2=1, the instruction clears all the executing actions and the outputs of the SFC POU, and the system runs the final scan..
5. The range of S_2 is 0–1. When it is out of range, it is processed as 0.
6. When the state of the SFC POU is PAUSE/STOP, executing this instruction is invalid.

Example

Refer to the SFCRUN (API 2500) programming example for more information.

Additional remarks

Refer to the ISPSoft User Manual for more information on SFCs.

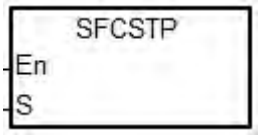
API	Instruction code	Operand	Function
2502	SFCSTP	**S**	SFC Stop

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S																

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1													

Pulse instruction	16-bit instruction	32-bit instruction
-	ES3	-

Symbol



S : Name of the SFC POU

Explanation

1. This instruction stops the SFC POU specified by **S**.
2. This instruction stops the SFC POU specified by S_1 only when the PLC is scanning the SFC POU.
3. When stopping, the instruction clears the SFC status and parameters, and the system runs the final scan.
4. When the state of the SFC POU is STOP, executing this instruction is invalid.

Example

Refer to the SFCRUN (API 2500) programming example for more information.

Additional remarks

Refer to the ISPSoft User Manual for more information on SFCs.

6.26 High-speed Output Instructions

6.26.1 List of High-speed Output Instructions

The following table lists the High-speed Output instructions covered in this section.

API	Instruction code		Pulse instruction	Function	Even-numbered axis*1	Odd-numbered axis*2
	16-bit	32-bit				
2700	–	DPLSY	–	High-speed pulse output (without ramp-up/down process)	V	V
2701	–	DPLSR	–	High-speed pulse output (with ramp-up/down process)	V	V
2702	PWM	DPWM	–	Pulse width modulation	V	V
2703	JOG	DJOG	–	JOG output	V	V
2704	–	DZRN	–	Zero return	V	–
2705	–	DPLSV	–	Adjustable pulse output	V	V
2706	–	DDRVI	–	Relative position control	V	V
2707	–	DDRVA	–	Absolute position control	V	V
2708	CSFO	–	–	Catch speed and proportional output	V	–
2709	–	DDRVM	–	Mark alignment positioning	V	–
2710	–	DPPMR	–	2-Axis relative-coordinate point-to-point synchronized motion	V	–
2711	–	DPPMA	–	2-Axis absolute-coordinate point-to-point synchronized motion	V	–
2712	–	DCICR	–	2-Axis relative-position clockwise arc interpolation	V	–
2713	–	DCICA	–	2-Axis absolute-position clockwise arc interpolation	V	–
2714	–	DCICCR	–	2-Axis relative-position counterclockwise arc interpolation	V	–
2715	–	DCICCA	–	2-Axis absolute-position counterclockwise arc interpolation	V	–
2716	–	DCCMR	–	Relative-position circle drawing	V	–
2717	–	DCCMA	–	Absolute-position circle drawing	V	–
2718	TPO	–	–	Position planning table controls the output	V	–
2719	–	DTPWS	✓	Setting single-axis output parameters in the position planning table	V	–
2720	–	DTPWL	✓	Setting linear interpolation parameters in the position planning table	V	–
2721	–	DTPWC	✓	Setting arc interpolation parameters in the position planning table	V	–

API	Instruction code		Pulse instruction	Function	Even-numbered axis*1	Odd-numbered axis*2
	16-bit	32-bit				
2723	–	DPPGB	–	Point to point go back and forth	V	–
2724	–	DZRN2	–	Zero return 2 (directional output can be defined)	V	V

NOTE:

*1: Even-numbered axes include Y0, Y2, Y4, and Y6.

*2: Odd-numbered axes include Y1, Y3, Y5, and Y7.

6.26.2 Explanation of High-speed Output Instructions

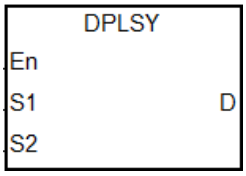
API	Instruction			Operand	Description
2700	D	PLSY		S_1, S_2, D	High-speed pulse output (without ramp-up/down process)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Pulse output frequency

S_2 : Number of pulses to output

D : Pulse output device

Explanation

1. This instruction specifies the pulse output (S_1, S_2) for the device specified in **D** with no ramp up or ramp down when the frequency changes. S_1 specifies the pulse output frequency range: for open collector output models, the range is between 0 Hz–200 kHz. The unit of the output frequency is 1 Hz. The error rate for 200 kHz is about 0.02% and for 100 kHz it is about 0.01%. The error rate decreases with decreasing frequency. For example, if the output frequency setting is 199990 Hz, the actual output is 199960 Hz. If the output frequency setting is 99999 Hz, the actual output is 99990 Hz. If the output frequency is out of the valid range, the PLC automatically processes it as the maximum or minimum pulse output frequency.

2. You can change the output frequency specified by S_1 during the execution of the instruction without using the ramp up/down process. The time to change the frequency is when the PLC is scanning the instruction and the instruction has completed output of a full pulse.

3. S_2 specified the number of output pulses. The range is between 0–2,147,483,647. When you set the number of output pulses to 0, then the number of pulses is not restricted and pulses are output constantly until the instruction is disabled. When the number of output pulses is set to less than 0, no pulses are output.

4. After the instruction starts to output pulses, the number of output pulses specified by S_2 cannot be changed.

5. **D** only allows Y0–Y7 as the output devices. After the instruction is executed, its output function becomes the high-speed output, and the basic instruction output point control is invalid. It is suggested that you do not use the general output function after using the high-speed output function in the program.

6. The ratio of Duty-OFF Time and Duty-ON Time for the pulse output is 1:1.

7. There is no limit to how many times you can use the instruction a program. But only one output instruction that uses the same output point can be executed in the same scan cycle. When several high-speed pulse output instructions start the same output point in the program, the PLC first performs the output based on the instruction that is executed first.

8. After every high-speed output instruction for every output point is executed, other instruction occupying the same output point can not start until the high-speed output instruction in execution is disabled.

9. When the high-speed output instruction is executed in an interrupt program or is not in the main process, it is suggested that you use the instruction with the auto-reset function when output is completed and the PLC updates the output state in the END instruction.

10. After the stop flag is set, the PLC stops the output and clears the busy flag only after executing the start instruction a second time and outputting a full pulse. The PLC continues to output pulses when the stop flag is reset and the previously stopped pulse output is not finished. You set and reset the stop flag is set in the program.

11. The high-speed output points and corresponding SM/SR are listed in the following table.

Output point number	Attribute#2	Y0	Y1	Y2	Y3	Y4	Y5
Busy flag	R	SM460	SM472	SM480	SM492	SM500	SM512
Completion flag#3	R/W	SM461	SM473	SM481	SM493	SM501	SM513
Stop flag	R/W	SM463	SM474	SM483	SM494	SM503	SM514
Output completion auto-reset#4	R/W	SM470	SM475	SM490	SM495	SM510	SM515
Present output position#1 （32-bit）	R/W	SR460 SR461	SR474 SR475	SR480 SR481	SR494 SR495	SR500 SR501	SR514 SR515
Output point number	Attribute #2	Y6	Y7				
Busy flag	R	SM520	SM532				
Completion flag #3	R/W	SM521	SM533				
Stop flag	R/W	SM523	SM534				

Output point number	Attribute[#2]	Y0	Y1	Y2	Y3	Y4	Y5
Output completion auto-reset [#4]	R/W	SM530	SM535	SM550	SM555	SM570	SM575
Present output position [#1] (32-bit)	R/W	SR520 SR521	SR534 SR535	SR540 SR541	SR554 SR555	SR560 SR561	SR574 SR575

Notes:

#1: All present output positions are latched when power is off.

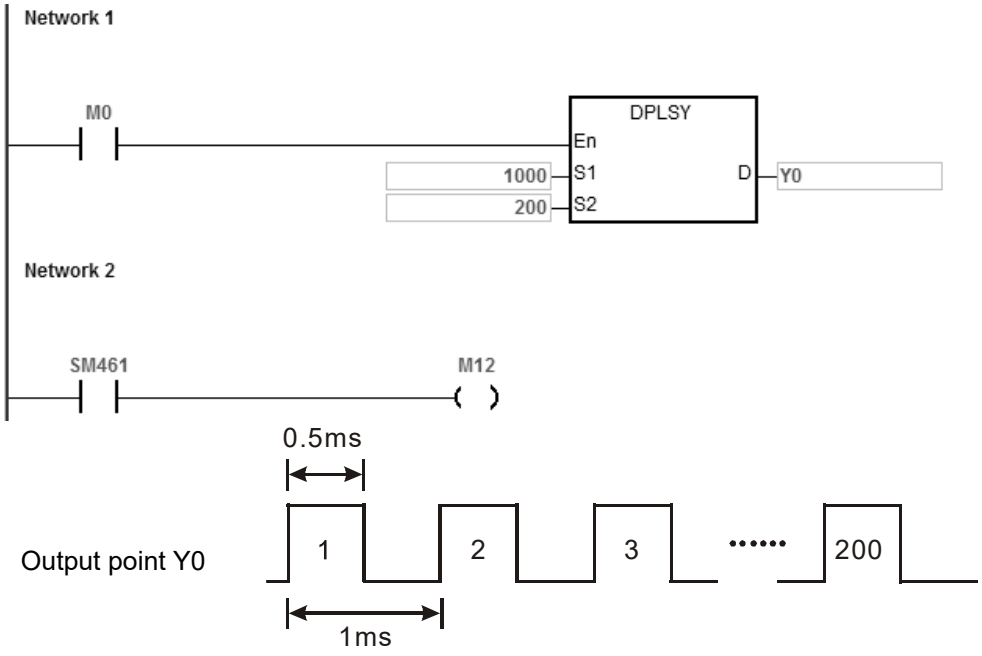
#2: R means Read-only registers, and the data in the registers cannot be modified. R/W means the data in the registers can be read and written.

#3: It is suggested that you clear the completion flag. If the completion flag is not cleared, it is cleared automatically when the high-speed output instruction is executed the next time.

#4: You set the output completion auto-reset flag. The PLC clears the flag automatically after the output is completed.

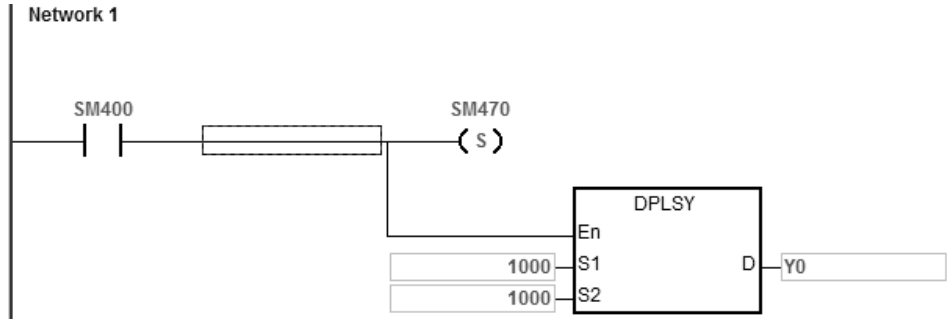
Example 1

1. When M0 is ON, Y0 outputs 200 pulses at 1kHz. SM461 is ON when the pulse output is completed and then M12 is ON.
2. When M0 is OFF, Y0 stops the output. It restarts the pulse output when M0 switches to ON again.

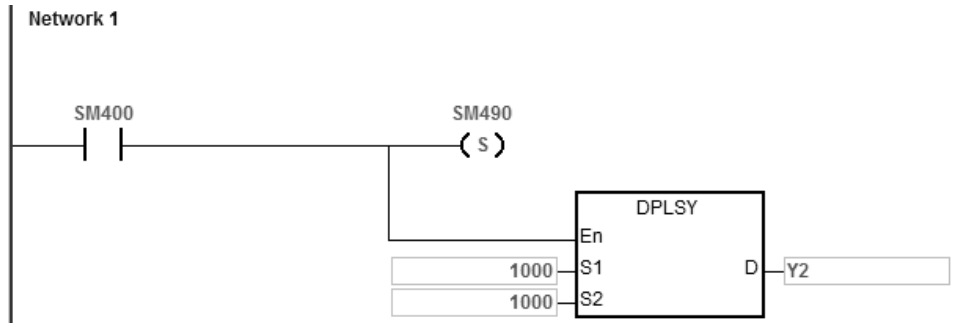


Example 2

External interrupt input X0



External interrupt input X1



Explanation

1. Y0 outputs 1000 pulses when X0 receives one external interrupt signal. Y2 outputs 1000 pulses when X1 receives one external interrupt signal.

2. When the external interrupt input X triggers the pulse output from Y, the interval time between the Y pulse output completion and the next external interrupt input X trigger must be one or more PLC scan cycles.

Example 3 (ST program)

```
0001 IF M0 THEN
0002     DPLSY(1000,1000,Y0);
0003     M1:=TRUE;
0004 ELSIF M1 AND SM461 THEN
0005     SM470:=TRUE;
0006     M1:=FALSE;
0007 END_IF;
```

Explanation

1. When M0 is ON, Y0 outputs 1000 pulses at 1kHz. When M1 is ON, it indicates Y0 is used for high-speed pulse output.

2. When the pulse output is completed, SM461 is ON. And if M1 is also ON, it triggers SM470 (output completion auto-reset for Y0) to be ON and Y0 is free and ready to be used again.

3. When M0 switches from OFF to ON, the pulse output is restarted.

4. When there are more than two high-speed output instructions in a program, do not use the variable name, M1, repeatedly.

5. It is suggested not to use ST language in a program if there is any chance the outputting needed to be stopped.

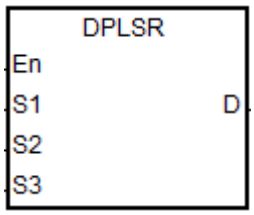
API	Instruction			Operand	Description
2701	D	PLSR		**S_1, S_2, S_3, D**	High-speed pulse output (with ramp-up/down process)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Target output frequency

S_2 : Number of output pulses

S_3 : Ramp-up/ down time

D : Pulse output device

Explanation

1. This instruction specifies the pulse output (**S_1**, **S_2**) for the device specified in **D** with ramp up or ramp down in the frequency change. **S_1** specifies the pulse output frequency between 0 Hz–200 kHz for the open collector output models. The unit of the output frequency is 1 Hz. The tolerable error rate for 200 kHz is about 0.02% and for 100 kHz it is about 0.01%. The error rate decreases with decreasing frequency. For example, if the output frequency setting is 199990 Hz, the actual output is 199960 Hz. If the output frequency setting is 99999 Hz, the actual output is 99990 Hz. If the output frequency is out of the valid range, the PLC automatically processes it as the maximum or minimum pulse output frequency.

2. After the instruction is executed, you can change the target output frequency specified by **S_1** and the PLC performs the ramp up or ramp down process according to the ramp up or ramp down time setting **S_3**. The time to change the frequency is when the PLC is scanning the instruction and the instruction has completed output of a full pulse.

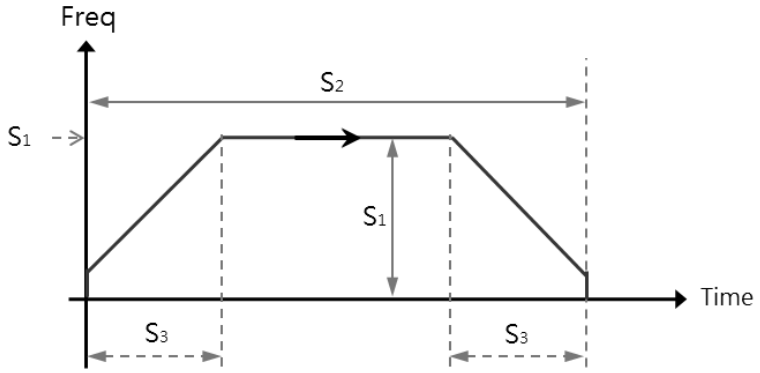
3. **S_2** is the number of output pulses. The range is between 0–2,147,483,647. When you set the number of pulses to 0, then the number of pulses is not restricted and pulses are output continuously until the instruction is disabled.

4. After the instruction starts to output pulses, the number of output pulses that **S_2** specifies cannot be changed.

5. **S_3** sets the ramp-up or ramp down time with the unit of 1ms. The value is effective when the instruction is executed for the first time. If the target frequency specified by **S_1** is modified in the ramp-up process, the ramp up or ramp down time is reloaded for execution. But if the target frequency is modified when the output enters the ramp-down process, the instruction ignores the change.

6. **D** only allows Y0–Y7 as the pulse output devices. After the instruction is executed, its output function becomes the high-speed output, and the general instruction output point control is invalid. It is suggested that you do not use the general output function after using the high-speed output function in the program.

7. The target output frequency **S_1**, number of output pulses **S_2** and ramp up/down time **S_3** are illustrated in the following graphic.



8. The ratio of Duty-OFF Time and Duty-ON Time for the pulse output is 1:1.

9. There is no limit to how many times you can use the instruction a program. But only one output instruction that uses the same output point can be executed in the same scan cycle. When several high-speed pulse output instructions start the same output point in the program, the PLC first performs the output based on the instruction that is executed first.

10. After the high-speed output instruction for one output point is enabled, other instructions that occupy the same output point can not start until the high-speed output instruction in execution is disabled.

11. When the high-speed output instruction is executed in an interrupt program or is not in the main program, it is suggested that you use the instruction with the auto-reset function when the output is completed and the PLC updates the output state in the END instruction.

12. After the stop flag is set, PLC performs the ramp-down stop; or it immediately stops the output only after the start instruction is executed a second time and outputs a full pulse.

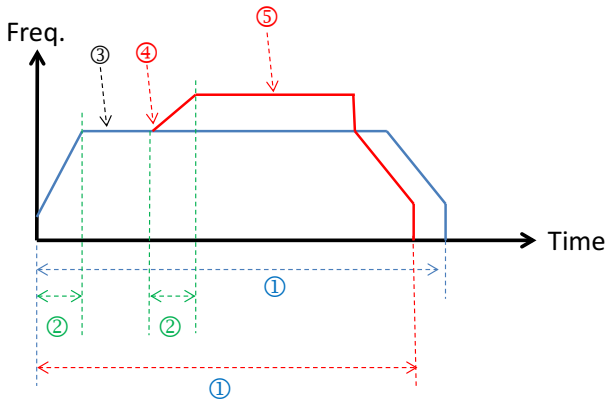
13. Refer to the PLSY instruction (API 2700) for an explanation of the high-speed output points and corresponding SM/SR.

14. Refer to Example 3 from the PLSY instruction (API 2700) for programming in ST language.

15. When the target output frequency is reached, you can change the set target output frequency. The frequencies for ramp up and ramp down are already set, if you change the set target output frequency during the instruction execution, the frequency goes either ramping up or ramping down. If more than one target output frequencies are set, it is suggested to set the last new target frequency the same as the original target frequency to avoid abrupt change when it needs to come to a stop.

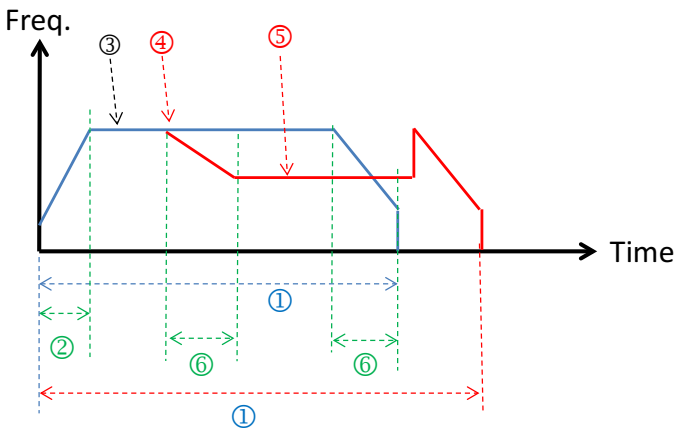
    See the example below. The blue line indicates the original target freueency and the red line indicates new target frequency.

    - Situation A: ramping up



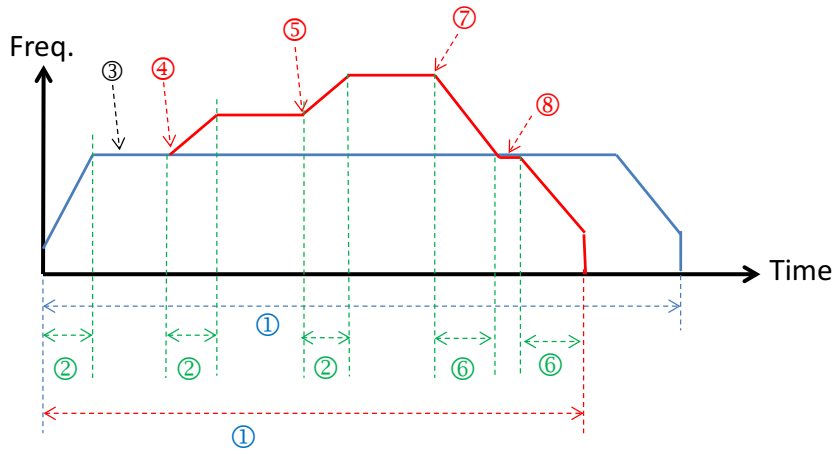
① Number of output pulses
② Ramp up time
③ The first target frequency
④ Position to change to the new target frequency
⑤ New target frequency

    - Situation B: ramping down



① Number of output pulses
② Ramp up time
③ The first target frequency
④ Position to change to the new target frequency
⑤ New target frequency
⑥ Ramp down time

- Suggested design: it is suggested to set the last new target frequency the same as the original target frequency to avoid abrupt change when it needs to come to a stop.



① Number of output pulses
② Ramp up time
③ The first target frequency
④ Position to change to the first new target frequency
⑤ Position to change to the second new target frequency
⑥ Ramp down time
⑦ Position to change to the third new target frequency
⑧ Position to change to the original set targe frequency and then start ramping down to stop.

API	Instruction			Operand	Description
2702	D	PWM		S_1, S_2, D	Pulse width modulation

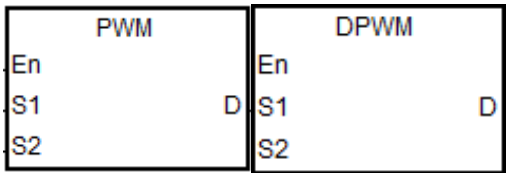
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●	●	●	●		○		○	○		
S_2	●	●			●	●	●	●	●		○		○	○		
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						

S_2		●	●		●	●	●						
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	ES3

Symbol



S_1 : Pulse output width

S_2 : Pulse output cycle

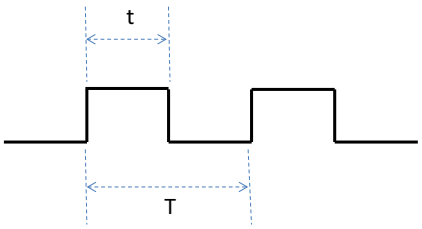
D : Pulse output device

Explanation

1. This instruction specifies the pulse output width. The 16-bit instruction uses 100 us as the output unit, while the 32-bit instruction uses 1 us as the output unit.

Output instruction	PWM	DPWM
Range of pulse output width S_1	0–30000	0–60000
Range of pulse output cycle S_2	0–30000	0–60000

2. S_1 (pulse output width--Duty ON) is defined as t, S_2 (pulse output cycle time--Cycle time) is T as shown below. It is recommended that $S_1 \leqq S_2$.



3. **D** only allows Y0–Y7 as the pulse output device.

4. There is no limit to how many times you can use the instruction a program. But only one output instruction that uses the same output point can be executed in the same scan cycle. When several high-speed pulse output instructions start the same output point in the program, the PLC first performs the output based on the instruction that is executed first.

5. If $S_1 \leqq 0$ or $S_2 \leqq 0$, it is processed as 0 (the pulse output is OFF). If $S_1 > S_2$, it is processed as $S_1 = S_2$. When $S_1 = S_2$ and S_2 is not 0, the pulse output is always ON.

6. You can modify the pulse output width S_1 and pulse output cycle S_2 when the PWM instruction is executed.

7. The relevant special registers SR are listed in the following table.

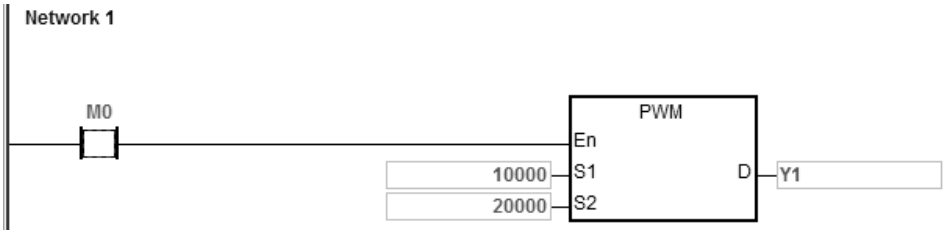
Output point number	Attribute	Y0	Y1	Y2	Y3	Y4	Y5
Present output position#1 （32-bit）	R/W	SR460 SR461	SR474 SR475	SR480 SR481	SR494 SR495	SR500 SR501	SR514 SR515
Output point number	Attribute	Y6	Y7				
Present output position #1 （32-bit）	R/W	SR520 SR521	SR534 SR535				

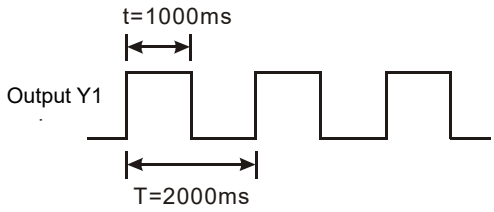
#1：All present output positions are latched when power is off.

8. Refer to Example 3 from the DJOG instruction (API 2703) for programming in ST language.

Example

When M0 is ON, Y1 outputs the following pulses. When M0 switches to OFF, Y1 output changes to OFF.





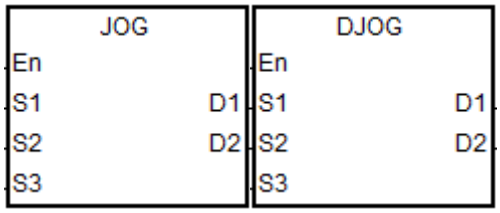
API	Instruction			Operand	Description
2703	D	JOG		**S_1, S_2, S_3, D_1, D_2**	JOG output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1					●	●	●	●	●		○		○	○		
S_2					●	●	●	●	●		○		○	○		
S_3					●	●	●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●	●		●	●	●						
S_2		●	●		●	●	●						
S_3		●	●		●	●	●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	ES3

Symbol



S_1 : Ramp-up time

S_2 : Target output frequency

S_3 : Ramp-down time

D_1 : Pulse output device

D_2 : Auxiliary output device

Explanation

1. This instruction specifies the JOG output (**S_1** , **S_2** , **S_3**) for the devices specified in **D_1** and **D_2**. **S_1** is the ramp-up time with the unit of 10 ms. For Example , the value 10 means the ramp-up time is 100 ms. After the instruction is executed, the output frequency can be divided by ten, and it speeds up once every 10ms. It reaches the JOG target frequency specified by **S_2** at the ramp-up time **S_1**.

2. When the instruction is disabled, its output frequency ramps down once every 10ms based on the ramp-down time in **S_3**. The output does not stop until reaching the ramp-down time. If you set the ramp-down time to 0, the output stops immediately.

3. The range of the ramp-up time **S_1** and the ramp-down time **S_3** is between 0–3000 (0–30 seconds). If the setting value is out of range, the PLC processes it as the minimum or maximum output value. The ramp-up time and

ramp-down time are affected by the scan time. If you require accurate ramp-up or ramp-down time, it is recommended that you use an output instruction with specified ramp up or ramp down time such as the DDRVI instruction (API 2706).

4. The range of the target output frequency **S_2** is between -200 kHz to 200 kHz. If the setting value is out of the range, the PLC processes it as the minimum or maximum output value. A positive number for the output frequency indicates forward output. A negative number for the output frequency indicates reverse output.

5. **D_1** allows only Y0–Y7 for the output point. For the auxiliary output point specified by **D_2,** refer to the output points in the following table. If you choose other output points or the M device, the value in SR indicating the output mode is invalid and the Pulse+direction mode is valid by default. **D_2** is a direction output point.

   D_1 selects the even output point number.

Output point for **D_1**	Y0	Y2	Y4	Y6
Direction output point for **D_2**	Y1	Y3	Y5	Y7
Busy flag	SM460	SM480	SM500	SM520
Output mode	SR462	SR482	SR502	SR522
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521

   D_1 selects the odd output point number.

Output point for **D_1**	Y1	Y3	Y5	Y7
Direction output point for **D_2**	Y20 or above or any M device (BOOL)			
Busy flag	SM472	SM492	SM512	SM532
Present output position	SR474 SR475	SR494 SR495	SR514 SR515	SR534 SR535

6. If SR=0, it indicates the Pulse+direction output mode. If SR=1, it indicates the A/B phase output mode. Note: you can select the output mode only when **D_1** uses an even output point and **D_2** uses a recommended direction output point.

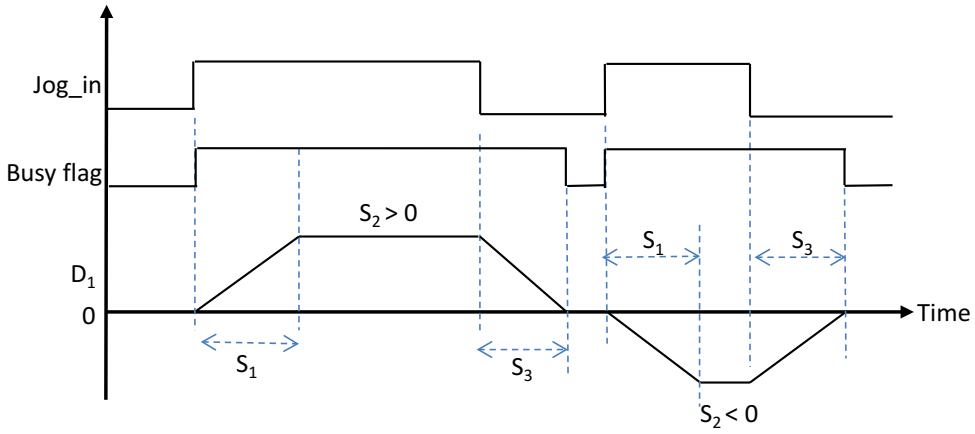
7. When the direction outputs are not using the default outputs to output, you can refer to the table below for setting up the outputting time that pulse outputting goes first before direction outputting does. So that you can be sure the outputting only happens when switching to the direction outputs. The unit for SR is millisecond and the range is 0-20 ms. Default is 0, indicating inactive.

Output point for D_1	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
SR number	SR640	SR641	SR642	SR643	SR644	SR645	SR646	SR647

Note: the time source here is from PLC scan time.

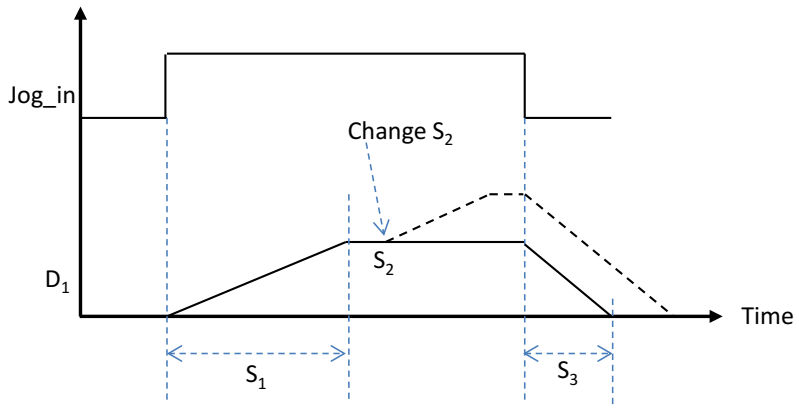
8. When the output begins, the Busy flag SM is ON. When the output is completed, the Busy flag is automatically reset to OFF and the Completion flag is not ON.

9. The following graph shows the output timing diagram. For the Busy flag in the following graph, refer to the Busy flag axis.



10. You can modify the target output frequency during instruction execution. When the new frequency is greater than the previous one, the instruction uses the ramp-up slope. When the new frequency is less than the previous one, the instruction uses the ramp-down slope.

11. The instruction sets the ramp-up/down slope through conversion of the set time and target frequency when the instruction is executed. The slope does not change with the changing target frequency in the output process. For example, the original target output frequency is 1 kHz and then it is modified to 2 kHz. The actual ramp-down time is different from the original time. The dotted line is the ramp-up and ramp down timing diagram after the target frequency is modified.



Example (ST program):

It is suggested to use ladder diagrams as the programming language in this instruction. If you need to use structured texts, you need to use the output completion auto-reset flag as well for the PLC to know the output point is free and ready to be used again when the output completes. Taking Y0 as the output point, the program should be written as below. See the following example.

```
0001 IF  M0  THEN
0002     DJOG(1000,5000,100,Y0,Y1);
0003     M1  :=TRUE;
0004 ELSIF  (NOT  M0)AND  M1  THEN
0005     SM470:=TRUE;
0006     M1:=FALSE;
0007 END_IF;
```

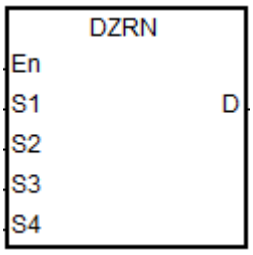
API	Instruction			Operand	Description
2704	D	ZRN		**S_1, S_2, S_3, S_4, D**	Zero return

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4	○															
D		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4	●												
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Target frequency for zero return

S_2 : JOG frequency for DOG

S_3 : Zero return mode

S_4 : Input device for DOG

D : Pulse output device

Explanation

1. This instruction causes the machine to return to the zero point. The range of the target frequency for zero return **S_1** is between 1 Hz–200 kHz. The JOG frequency **S_2** should be less than the target frequency **S_1**. The JOG frequency **S_2** is the start frequency. If **S_1** is less than **S_2**, **S_1** is automatically revised processed as equal to **S_2**.

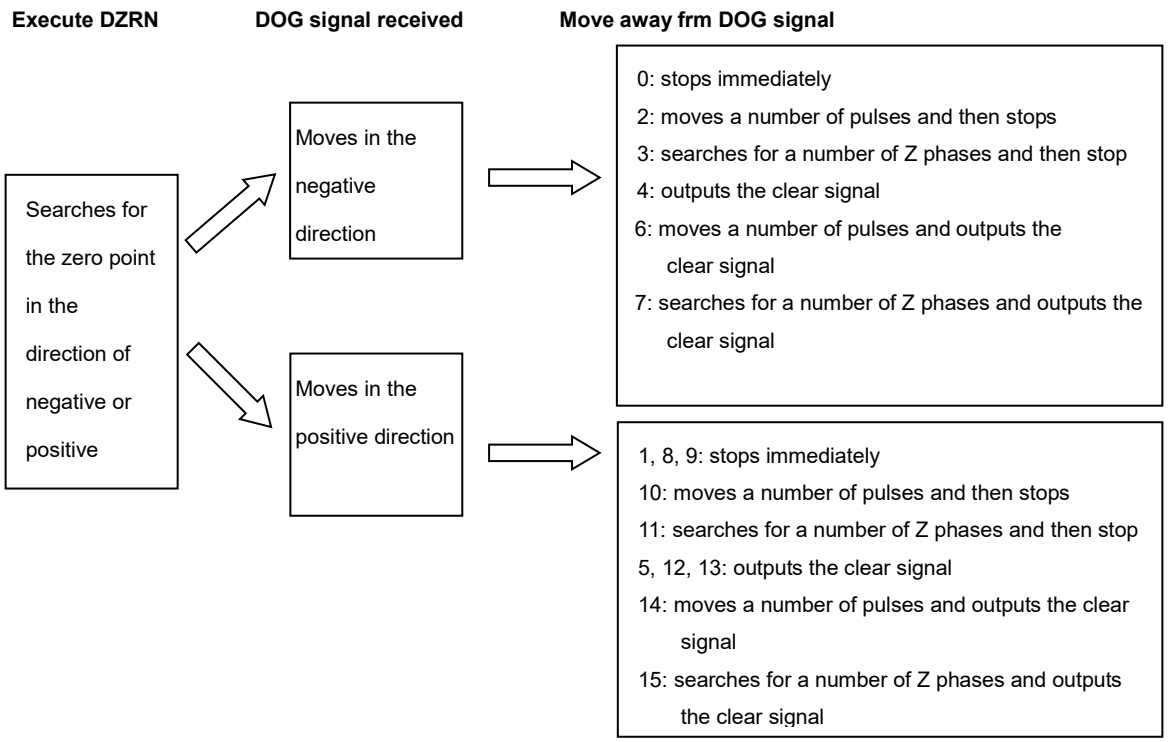
2. The input point for **S_4** and output point for **D** must match. Do not change them during instruction execution. The input point for **S_4** is suggested to use the 16 high-speed input points X0–X7 and X10–X17. They will not be affected by PLC instruction scan time. If you use X20 successive input points or M devices, they will be affected by the PLC instruction scan time. Refer to the following table for the selection of **D** output point and direction output point. If **D** is not the preset Pulse+direction output (default: 0), change the mode to A/B phase output by setting SR to 1.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Input point for **S_4**	Can be any one of the input points X0–X7 and X10–X17. But the same input point cannot be selected for different axis output. If the DOG point shakes or the switch bounces, set the input point filter time in HWCONFIG.			
Output point for **D**	Y0	Y2	Y4	Y6
Direction output point	Y1	Y3	Y5	Y7
Output mode	SR462	SR482	SR502	SR522
Busy flag	SM460	SM480	SM500	SM520
Completion flag	SM461	SM481	SM501	SM521
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521

3. Use **S_3** to select the zero return mode. The function code is set by the two high and low16-bit parameters. See the following table for details.

S_3: select the zero return mode				
High 16-bit	Low 16-bit			
b31~b16	b15~b6	b5	b4	b3~b0
Number of pulses for motion	reserved	Direction setting*1 0: in the negative direction 1: in the positive direction	Setting DOG signal 0: contact A 1: contact B	Mode settig 0~15 (F)
Number of Z phases				

See the diagram below for mode setting.



4. Use **S_3** to select the zero return mode. The function code is set by the two high and low16-bit parameters. See the following table for details.

Functions	Code		Explanation
	High 16-bit	Low 16-bit	
Leaves the zero point in the negative direction and then stops (Mode 0)	0	0	When the instruction is executed, the search for the zero point is in the negative direction with the target frequency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency is decreased to the JOG speed and the motion in the negative direction continues, and does not stop until the zero point signal changes from ON to OFF.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	1	When the instruction is executed, the search for the zero point is in the negative direction with the target frequency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency decreases to 0 immediately, and then the motion is in the positive direction at the JOG speed, and does not stop until the zero point signal changes from ON to OFF.
Mode 0 Moves again after returning to the zero point	Number of pulses for motion	2	Returning to the zero point is the same as that for the low 16-bit code. After the zero point is ON, the motion continues according to the number of specified pulses. When the high 16-bit code is a positive number, the search is in the positive direction. A negative value means that the search is in the negative direction.
Mode 0 Searches Z phase after returning to the zero point (Z phase input point is set in HWCONFIG)	Number of Z phases	3	Returning to the zero point is the same as that for the low 16-bit code. After returning to the zero point, the motion continues according to the number of Z phases. When the high 16-bit code is a positive number, the search is in the positive direction. A negative value indicates that the search is in the negative direction. Suppose you specified that the rising-edge trigger of X0 as the condition for the Z phase input in HWCONFIG. The counting is performed once whenever the rising-edge trigger for X0 occurs.
Mode 0 Outputs the clear signal after returning to the zero point. (Output clear point is set in HWCONFIG)	Number of pulses for motion or number of Z phases	4+0=4 4+1=5 4+2=6 4+3=7 (bit 2=ON)	Choosing a value between 4–7 means selecting the functions codes 0–3 respectively, and the specified output point sends an ON signal that is about 20ms wide when the function execution completes. The range of the output point is Y14–Y17 and Y20–Y27. For example, if you specify Y22 as the output point in HWCONFIG, it indicates Y22 is for the output of clear signals.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	8+0=8 8+1=9 (bit 3=ON)	The operation for zero point return is the same as that for code 1 (mode 1).

Mode 1 outputs the number of pulses after returning to the zero point	Number of pulses for motion	8+2=10 (bit 3=ON)	The operation for zero point return is the same as that for low 16-bit code 1. After returning to the zero point, the motion continues in accordance with the number of specified pulses. When the value of the high 16-bit code is a positive number, the motion is in the positive direction. A negative number indicates that the motion is in the negative direction.
Mode 1 Searches for Z phase after returning to the zero point (Z phase input point is set in HWCONFIG)	Number of Z phases	8+3=11 (bit 3=ON)	The operation for zero point return is the same as that for low 16-bit code 1. After returning to the zero point, the motion continues in accordance with the number of Z phases to seek. When the value of the high 16-bit code is a positive number, the motion is in the positive direction. A negative number indicates that the motion is in the negative direction. If the rising edge trigger for X1 is the condition for Z phase input, counting is performed once when the rising-edge trigger for X1 occurs.
Mode 1 Outputs the clear signal after returning to the zero point (Output clear point is set in HWCONFIG)	0 or number of pulses or number of Z phases	12–15 (bit 3=bit 2 =ON)	After returning to the zero point in mode 1, the 20ms-width clear signal is output.
DOG point is B point		+16 (bit 4=ON)	When in the low 16-bit code, bit 4 is ON, it means the zero point is ON as the DOG point changes from ON to OFF and the zero point is left as the DOG point changes from OFF to ON.

5. The execution sequence is based on the value of the low 16-bit code in the table above, and described below.

   →The direction of DOG signal is determined by the value of bit 5.

   → The DOG signal is determined by the value of bit 4.

   → Mode 0 or mode 1 for the zero point return, selected according to the value of bit 3.

   → The operation of the zero point return is performed according to the values of bit 1 and bit 0.

   → The operation of the clear signal specified by bit 2 is performed.

6. Set the input point and the rising edge trigger condition in HWCONFIG, when the position control system needs positive and negative limit input points. Note that the limit input points must not be the same as the zero point or Z phase input points.

7. The Completion flag is set to ON after the instruction finishes performing the specified function. For example, for function code 6, the PLC sets the Completion flag to ON only when the Z phase seeking completes.

8. After the DZRN instruction is executed, an interrupt service program is not executed till the DZRN instruction is disabled, if the specified input point for the zero point is the same as that for the external input interrupt in the program.

9. When the limit switch is specified in HWCONFIG, and there is an external input interrupt service program, the interrupt program will be executed at the same time.

10. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

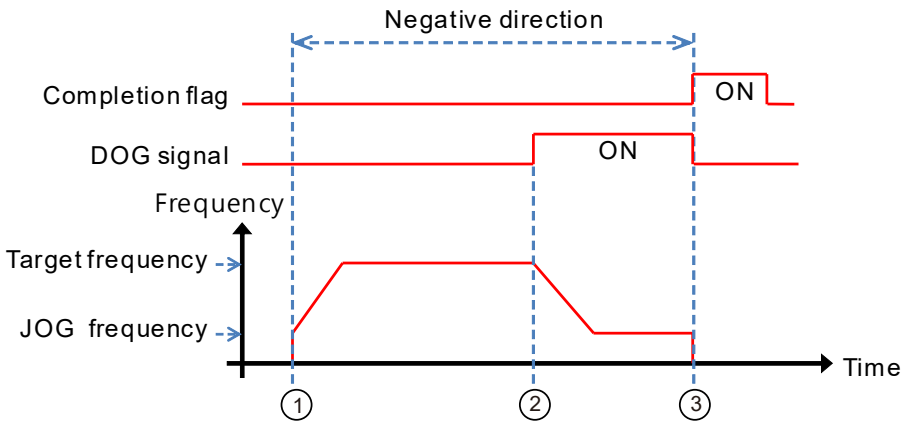
The steps for performing the functions are as below

Function code 0:

① The DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the DOG signal is received, the output frequency decreases to the JOG frequency. The output continues in the negative direction and does not stop until the zero point signal changes from ON to OFF.

③ The output stops when the signal changes from ON to OFF and the axis moves away from the DOG signal.

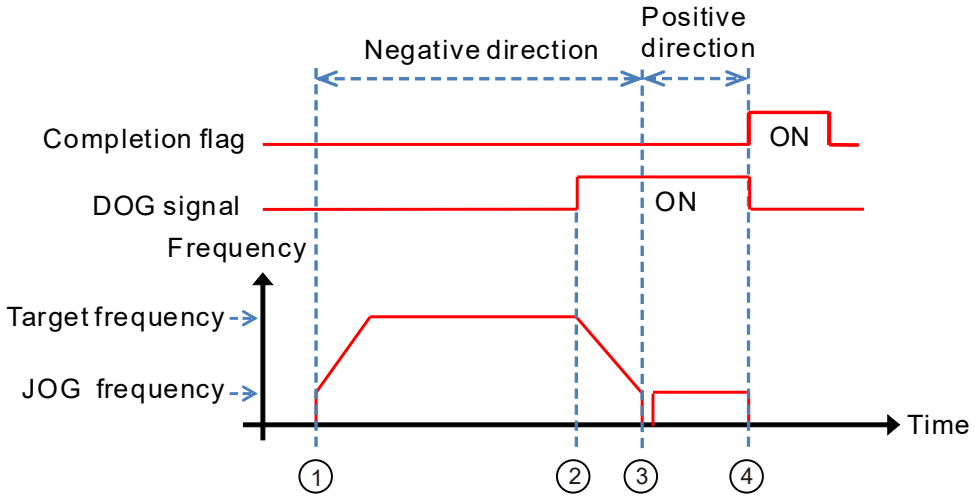


Function code 1:

① The DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the DOG signal is received, the output is in the positive direction with the JOG frequency after the output frequency decreases, and the motion direction reverses. The output does not stop until the zero point signal changes from ON to OFF.

③ The axis moves away from the DOG signal and PLC stops when the signal changes from ON to OFF.



Function code 2:

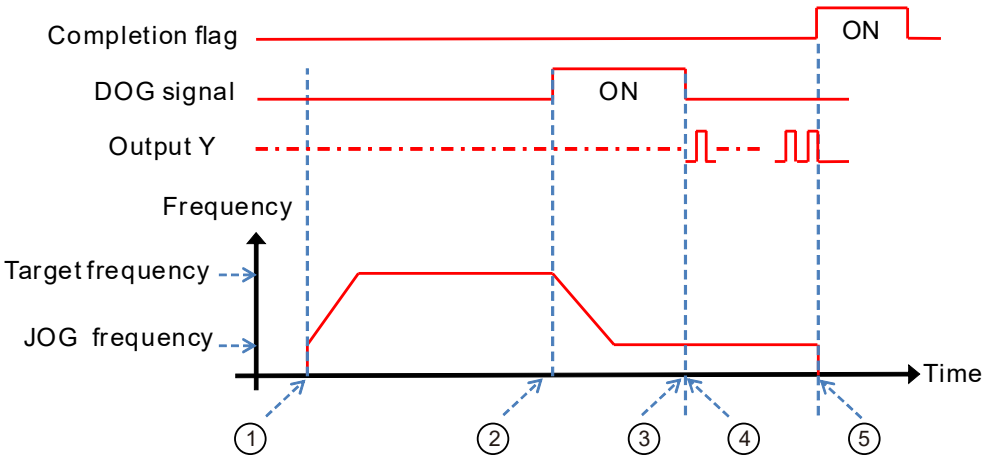
① The DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the DOG signal is received, the output decreases the frequency to the JOG frequency and continues in the negative direction.

③ When the DOG signal is left and the signal changes from ON to OFF, the specified number of pulses are output.

④ The first pulse output starts.

⑤ When the 100th pulse output completes, the PLC stops and the Completion flag is ON.



Function code 3:

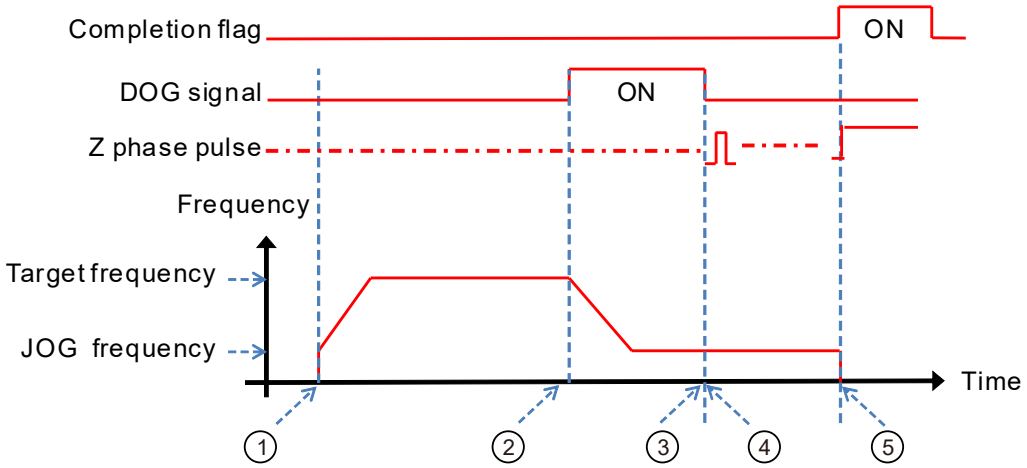
① DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the DOG signal is received, the output frequency decreases to the JOG frequency and the motion continues in the negative direction.

③ The motion goes on according to the number of Z phases when the signal changes from ON to OFF after the axis moves away from the DOG signal.

④ The first Z phase pulse

⑤ The motion stops after the 2nd Z phase completes and the Completion flag is ON.

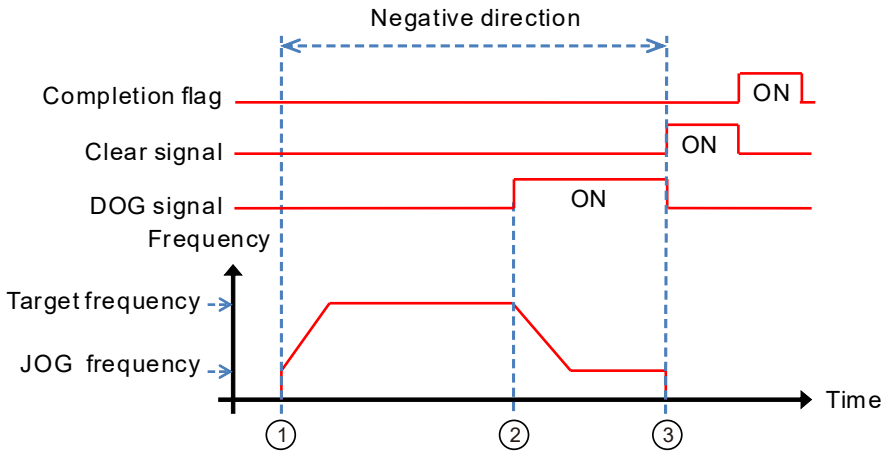


Function code 4:

① DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the DOG signal is received, the output frequency decreases to the JOG frequency and continues in the negative direction. The output does not stop until the zero point signal changes from ON to OFF.

③ After the axis moves away from the DOG signal, the output stops when the signal changes from ON to OFF and the clear signal is be ON for about 20 milliseconds.



Function code 0+ the negative limit function enabling:

You set the negative limit input point in HWCONFIG, and then download the setting to the PLC. The PLC automatically calculates the negative limit function when the instruction is executed.

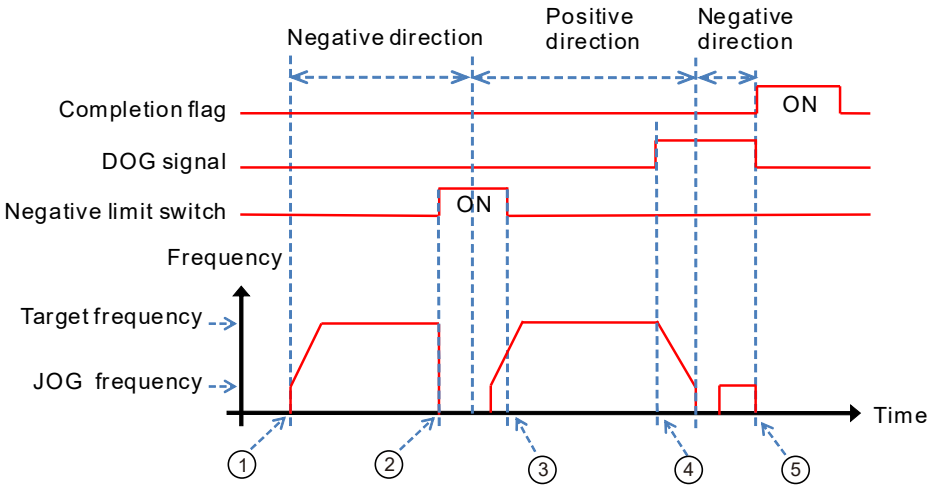
① DZRN function is executed and the search for the zero point is in the negative direction with the target frequency.

② After the negative limit switch is ON, the motion stops and then goes in the positive direction after reversing direction.

③ The motion continues in the positive direction after leaving the negative limit switch.

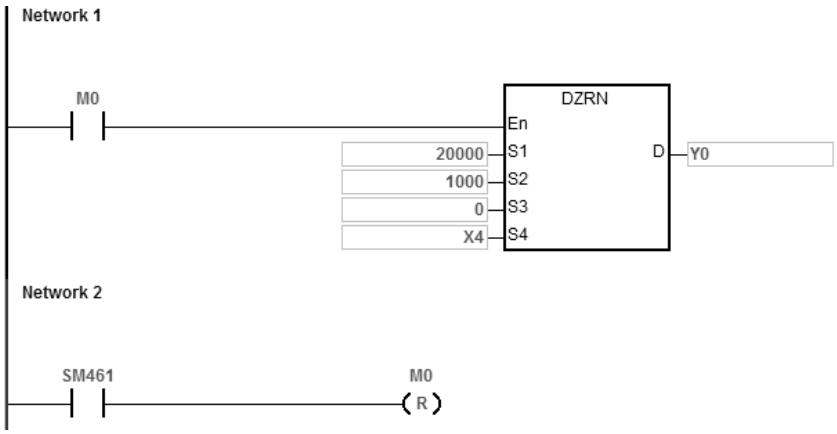
④ The output frequency ramps down after receiving the DOG signal. The reverse output is performed with the JOG frequency after reversing direction.

⑤ The output stops when the signal changes from ON to OFF after the axis moves away from the DOG signal.



Example 1

When M0 is ON, outputting the pulse from Y0 with a frequency of 20 kHz to search for the zero point in the negative direction. When the DOG signal is received and X4 is ON, it keeps moving in the negative directon with the JOG frequency of 1 kHz. The output stops immediately after X4 changes from ON to OFF.



Explanation

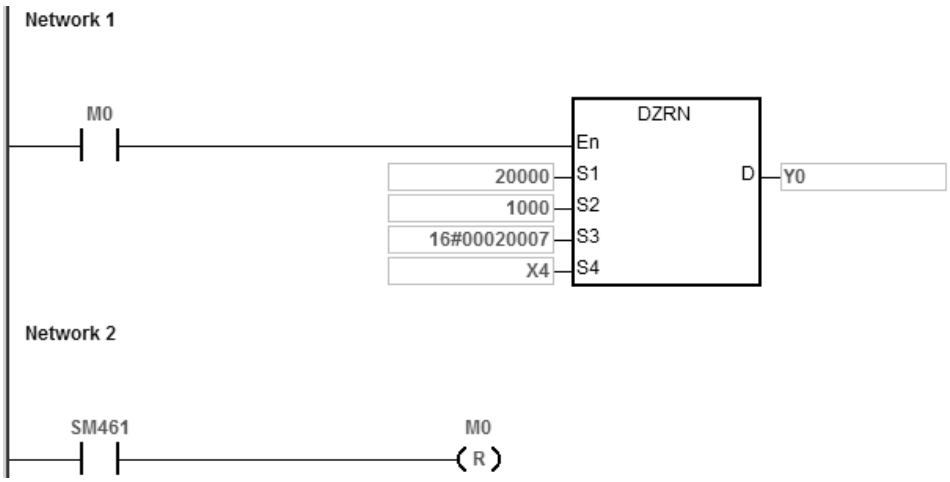
S_3 is set to 0.

High 16-bit [0000] is to disable the function to move a number of pulses or to search for Z phases.

Low 16-bit [0000] is Mode 0; when the DOG signal is received, the axis moves in the negative direction; after the axis moves away from the DOG signal, it stops immediately.

Example 2

When M0 is ON, outputting the pulse from Y0 with a frequency of 20 kHz to search for the zero point in the negative direction. When the DOG signal is received and X4 is ON, the PLC decreases the frequency to the JOG frequency of 1 kHz and keeps moving in the positive directon with the JOG frequency of 1 kHz. When X4 is OFF, the PLC starts to seek the Z phase pulse in the positive direction. When X5 receives two pulses, the PLC stops and Y14 outputs a 20ms-width pulse.



Explanation

1. If you specify the rising-edge trigger at X5 as the condition for Z phase input in HWCONFIG, the count is incremented once whenever the rising-edge trigger at X5 occurs.

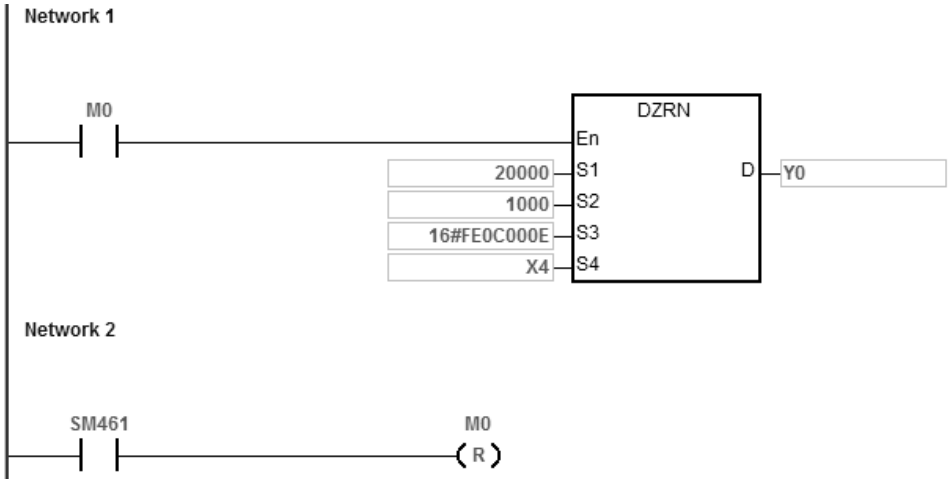
2. Y14 is specified as the output point for outputting the clear signal in HWCONFIG.

3. **S_3** is set as 16#00020007.
   High 16-bit [0002] is to search for the Z phase twice, once the axis moves away from the DOG signal.

   Low 16-bit [0007] is Mode 7; when the DOG signal is received, the axis moves in the negative direction; after the axis moves away from the DOG signal, and found the Z phase, a clear signal is outputted (20ms).

Example 3

When M0 is ON, outputting the pulse from Y0 with a frequency of 20 kHz to search for the zero point in the negative direction. When the DOG signal is received and X4 is ON, the PLC decreases the frequency to the JOG frequency of 1 kHz and keeps moving in the positive directon with the JOG frequency of 1 kHz. When X4 is OFF, the axis starts to move after 500 pluse output completes in the negative direction. Y14 outputs a 20ms-width pulse and then stops outputting.



Explanation

1. Y14 is specified as the output point for outputting the clear signal in HWCONFIG.

2. **S_3** is set as 16#FE0C000E.
   High 16-bit [FE0C=-500] once the axis moves away from the DOG signal. The axis starts to move after 500 pluse output completes in the negative direction.

   Low 16-bit [000E] is Mode 14; when the DOG signal is received, the axis moves in the positive direction; after the axis moves away from the DOG signal, a clear signal is outputted (20ms).

.

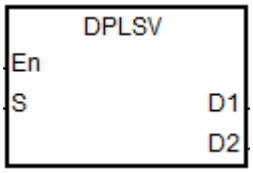
API	Instruction			Operand	Description
2705	D	PLSV		**S**，**D_1**，**D_2**	Adjustable pulse output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S ： Pulse output frequency

D_1 ： Pulse output device

D_2 ： Pulse direction output device

Explanation

1. This instruction adjusts the pulse output frequency and direction (**S**) for the devices specified in **D** and **D_2**. **S** specifies the pulse output frequency: for open collector output models, the range is between -200 kHz to +200 kHz. The minus sign and plus sign indicate the positive direction and negative pulse direction. You can change the pulse output frequency during the pulse output; but if the new frequency is different from the previous frequency in direction, the instruction stops the output, and one cycle later it outputs the target frequency again in a new direction.

2. Refer to the table below for pulse output devices for **D_1** and **D_2**. When the output mode for **D_1** and **D_2** is not the default Pulse+direction output (0 is the default value), set SR to 1, which changes the mode to A/B phase output.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Output point for **D_1**	Y0	Y2	Y4	Y6
Direction output point for **D_2**	Y1	Y3	Y5	Y7
Output mode	SR462	SR482	SR502	SR522
Busy flag	SM460	SM480	SM500	SM520
Output present register	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521

3. The output point for the pulse direction output device **D_2** is affected by the scan cycle if it is not the recommended output point in the table above.

4. The pulse direction output device **D_2** changes its own state according to the minus or plus sign in **S**. **D_2** is OFF if **S** is plus (+) and **D_2** is ON if **S** is minus (-).

5. There is no ramp-up or ramp-down setting in the instruction. The instruction does not perform a ramp-up operation at the beginning or ramp-down at stopping. If you want the ramp-up and ramp-down function, use the DRAMP instruction (API 0703) for increasing and decreasing the pulse output frequency.

6. While the instruction is executing the pulse output, the output immediately stops if the drive condition changes to OFF.

7. When the direction outputs are not using the default outputs to output, you can refer to the table below for setting up the time for directional outputting goes first before pulse outputting. So that you can be sure the outputting only happens when switching to the direction outputs. The unit for SR is millisecond and the range is 0-20 ms. Default is 0, indicating inactive.

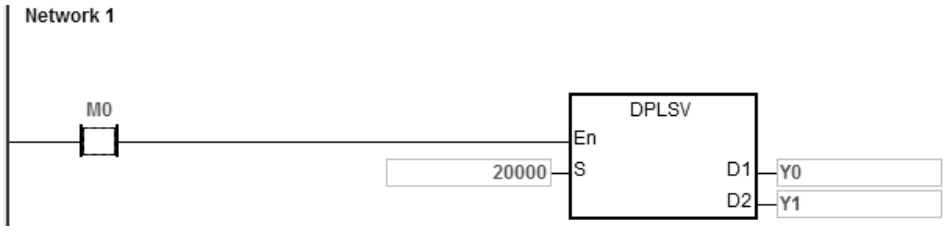
Output point for **D_1**	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
SR number	SR640	SR641	SR642	SR643	SR644	SR645	SR646	SR647

Note: the time source here is from PLC scan time.

8. Refer to Example 3 from the DJOG instruction (API 2703) for programming in ST language.

Example

When M0 is ON, Y0 outputs the pulse at 20 kHz. Y1 = OFF means the positive direction for the pulse output.



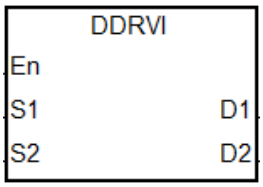
API	Instruction			Operand	Description
2706	D	DRVI		S_1, S_2, D_1, D_2	Relative Position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Number of output pulses (relative positioning)

S_2 : Pulse output frequency

D_1 : Pulse output device

D_2 : Pulse direction output device

Explanation

1. This instruction specifies the output pulse setting for relative positioning. S_1 is the number of output pulses (relative positioning). The range is between -2,147,483,648 to +2,147,483,647, and the +/- signs indicate the positive and negative pulse directions. When the instruction is executed and S_1 is 0, it indicates not outputting and output completion flag not be set to ON. If 0 is a possible output number in your program, it is suggested to add more conditions in your program to rule out this possibility.

2. S_2 is the pulse output frequency, and the range is between 0 Hz–200 kHz for open collector output models. If the value in S_2 is less than the Start/end frequency (Hz) set in SR (refer to the SR table below for more details), PLC operates according to the values set in SR.

3. Refer to the following table for the selection of pulse output devices for D_1 and D_2. When the output mode for D_1 and D_2 is not the default Pulse+direction output (0 is the default value), modify the mode by setting SR to 1, which changes the mode into A/B phase output.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Output point for **D_1**	Y0	Y2	Y4	Y6
Direction output point for **D_2**	Y1	Y3	Y5	Y7
SR number	SR462	SR482	SR502	SR522

DDRVI and DDRVA support the output devices for **D_1** which can be the odd points as shown in the following table. Only the Pulse+direction output mode is supported. It is not recommended that you use the input points Y0, Y2–Y10 as the high-speed output points.

Axis number	Axis 7	Axis 8	Axis 9	Axis 10
Output point for **D_1**	Y1	Y3	Y5	Y7
Direction output point for **D_2**	Y120 or above or any M device (BOOL)			

4. The output point for the pulse direction output device **D_2** is affected by the scan cycle if it is not the recommended output point in the table above.

5. When **D_2** uses the output mode Pulse+direction and **S_1** has a minus sign, **D_2** is ON. When **S_1** has a plus sign, **D_2** is OFF. **D_2** is not OFF immediately after the pulse output in the negative direction is completed. **D_2** switches to OFF when the contact for the execution of the instruction is OFF.

6. For the instruction execution, you must use the parameters such as the start/end frequency and ramp-up/down time. Refer to the following table for SM/SR corresponding to the axes.

   The following table applies when **D_1** selects the even output points as the output devices.

Output point for **D_1**	Y0	Y2	Y4	Y6
Busy flag	SM460	SM480	SM500	SM520
Completion flag	SM461	SM481	SM501	SM521
Reversing the output direction	SM462	SM482	SM502	SM522
Stop flag	SM463	SM483	SM503	SM523
Enabling S curve ramp-up/down	SM468	SM488	SM508	SM528
Enabling fixed slope ramp-up/down	SM469	SM489	SM509	SM529
Output completion auto-reset	SM470	SM490	SM510	SM530
Executing an interrupt program when pulse output ends	SM471	SM491	SM511	SM531

Corresponding interrupt I number	I500	I501	I502	I503
The output immediately stops when the instruction is disabled or stops	SM476	SM496	SM516	SM536
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521
Start/end frequency (Hz）	SR463	SR483	SR503	SR523
Ramp-up time (ms)	SR464	SR484	SR504	SR524
Ramp-down time (ms)	SR465	SR485	SR505	SR525
Target frequency of the fixed slope	SR472 SR473	SR492 SR493	SR512 SR513	SR532 SR533
S curve mode	SR604	SR605	SR606	SR607
Present output frequency	SR610 SR611	SR612 SR613	SR614 SR615	SR616 SR617

The following table applies when D_1 selects the odd output points as the output devices.

Output point for D_1	Y1	Y3	Y5	Y7
Busy flag	SM472	SM492	SM512	SM532
Completion flag	SM473	SM493	SM513	SM533
Output stop flag	SM474	SM494	SM514	SM534
Output completion auto-reset	SM475	SM495	SM515	SM535
The output immediately stops when the instruction is disabled or stops	SM477	SM497	SM517	SM537
Present output position	SR474 SR475	SR494 SR495	SR514 SR515	SR534 SR535
Start/end frequency(Hz）	SR476	SR496	SR516	SR536
Ramp-up/down time（ms）	SR477	SR497	SR517	SR537

Some flags, such as S curve ramp-up/down, fixed slope ramp-up/down, and executing an interrupt program when pulse output ends, are not supported by the odd output points.

If the output is ongoing, the preset output is the ramp-down stop when the instruction is disabled or stops temporarily. If the output must be stopped immediately, set the immediately stop flag to ON.

7. Added new SR652, SR653 for displaying number of pulse in ramp-up area and new SR654, SR655 for displaying number of pulse in ramp-down area.

Refer to the following table for Attribute, Factory default and latched values of SM/SR corresponding to the axes.

Item	Attribute	Stop→Run	Run→Stop	Power ON	Factory default	Latched
Busy flag	R	OFF	OFF	OFF	OFF	N
Completion flag	R / W	OFF	OFF	OFF	OFF	N
Stop flag	R / W	OFF	OFF	OFF	OFF	N
Output completion auto-reset	R / W	OFF	OFF	OFF	OFF	N
Enabling S curve ramp-up/down	R / W	OFF	OFF	OFF	OFF	N
Enabling fixed slope ramp-up/down	R / W	OFF	OFF	OFF	OFF	N
Executing an interrupt program when pulse output ends	R / W	OFF	OFF	OFF	OFF	N
The output immediately stops when the instruction is disabled or stops	R / W	OFF	OFF	OFF	OFF	N
Present output position	R / Wd	No change	No change	No change	0	Y
Start/end frequency(Hz）	R / Wd	No change	No change	No change	200	Y
Ramp-up time（ms）	R / Wd	No change	No change	No change	200	Y
Ramp-down time（ms）	R / Wd	No change	No change	No change	200	Y
Target frequency of the fixed slope	R / Wd	No change	No change	No change	0	Y
S curve mode	R / Wd	No change	No change	0	0	N
Present output frequency	R	0	0	0	0	N

Note: R mean the device is readable. W means the device is writable. Wd means the device is writable any time except when performing the high-speed output.

8. After output completion auto-reset is enabled, the flag is automatically reset to OFF when the output completes. You must set the flag to ON again if the auto-reset function is needed for the next output operation. It is suggested that you use the rising-edge or falling-edge to trigger the flag every time. The function is usually used when the output DDRVI instruction cannot be scanned and executed by PLC program after DDRVI is executed. For example, when the instruction is executed in an interrupt program.

9. If you use the Completion flag to set the I interrupt function, the flag is automatically reset to OFF when the output completes and the interrupt program starts. You must set the flag to ON again for the next output in which the interrupt occurs. It is suggested that you use the rising-edge or falling-edge to trigger the flag every time.

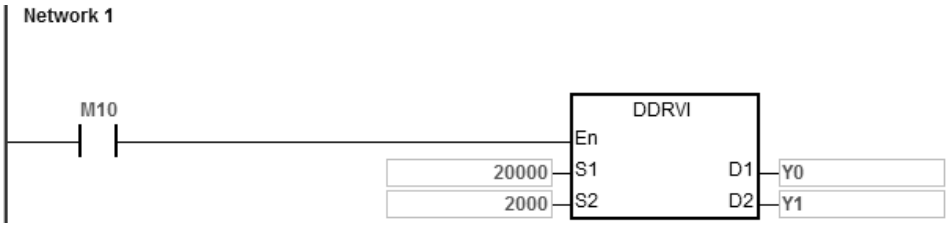
10. When the direction outputs are not using the default outputs to output, you can refer to the table below for setting up the time for directional outputting goes first before pulse outputting. So that you can be sure the outputting only happens when switching to the direction outputs. The unit for SR is millisecond and the range is 0-20 ms. Default is 0, indicating inactive.

Output point for **D_1**	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
SR number	SR640	SR641	SR642	SR643	SR644	SR645	SR646	SR647

Note: the time source here is from PLC scan time.

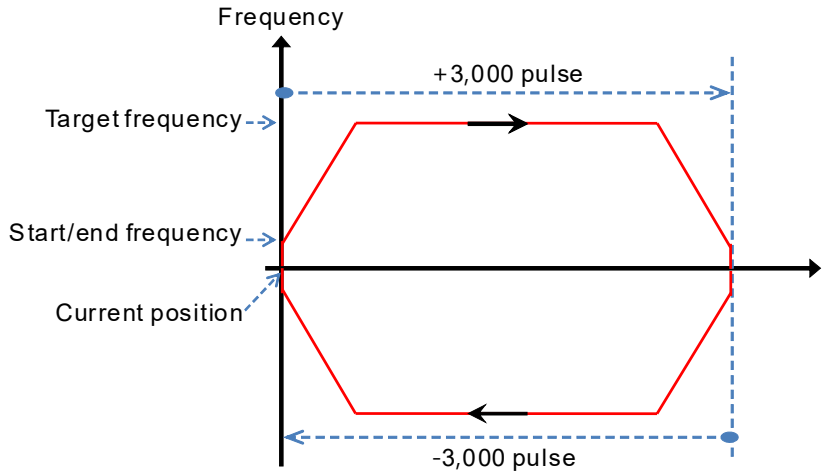
Example 1

When M10 is ON, Y0 outputs 20,000 pulses at 2 kHz (relative positioning). Y1 = OFF means the positive direction.



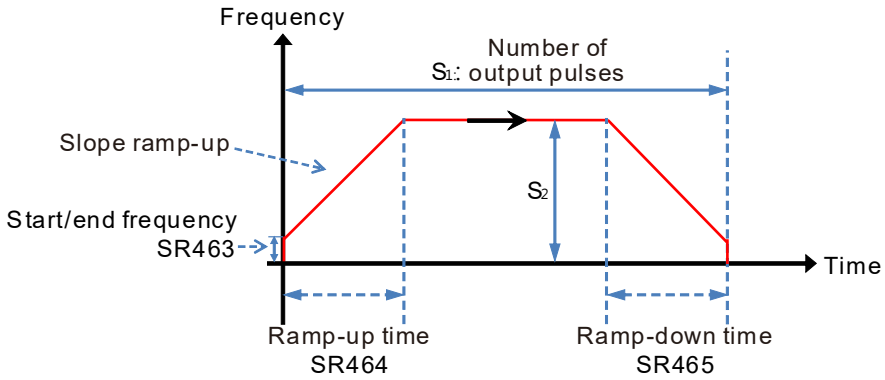
Additional remarks

1. The following graph explains the relative positioning by specifying the travel distance from the current position with the +/- signs.



2. Setting the ramp up/down and the items for relative positioning

   a) Y0 output curve diagram

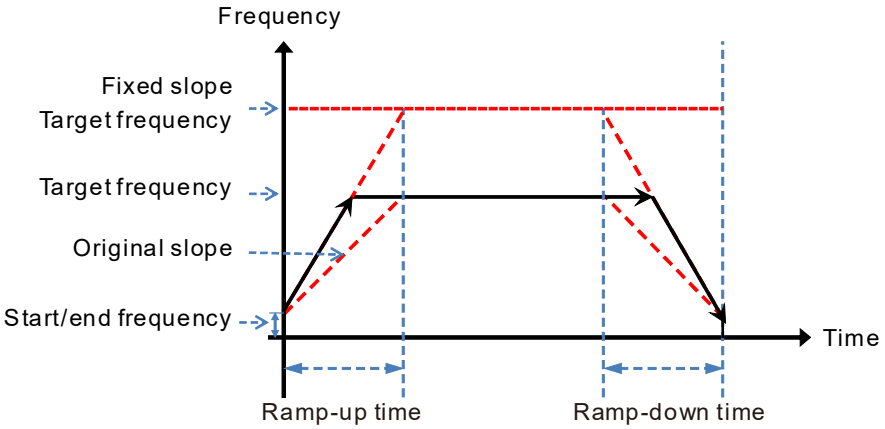


   b) You can create several DRVI instructions using the same output point in the program. The PLC can start only one DRVI instruction when executing the program. For example, when one instruction starts Y0 for output, then other instructions that also use Y0 for output are not executed. The instruction that starts the output point first uses the point first.

   c) After the instruction is executed, any modification of the parameters is not accepted until the instruction is disabled.

3. Ramp-up and ramp-down for fixed slopes

a) The following graph shows the Y0 output curve diagram.



b) Start the fixed slope flag SM469 and write the target frequency of the fixed slope into SR472.

c) The new slope takes the place of the original slope and the DDRVI instruction performs the positioning as shown above.

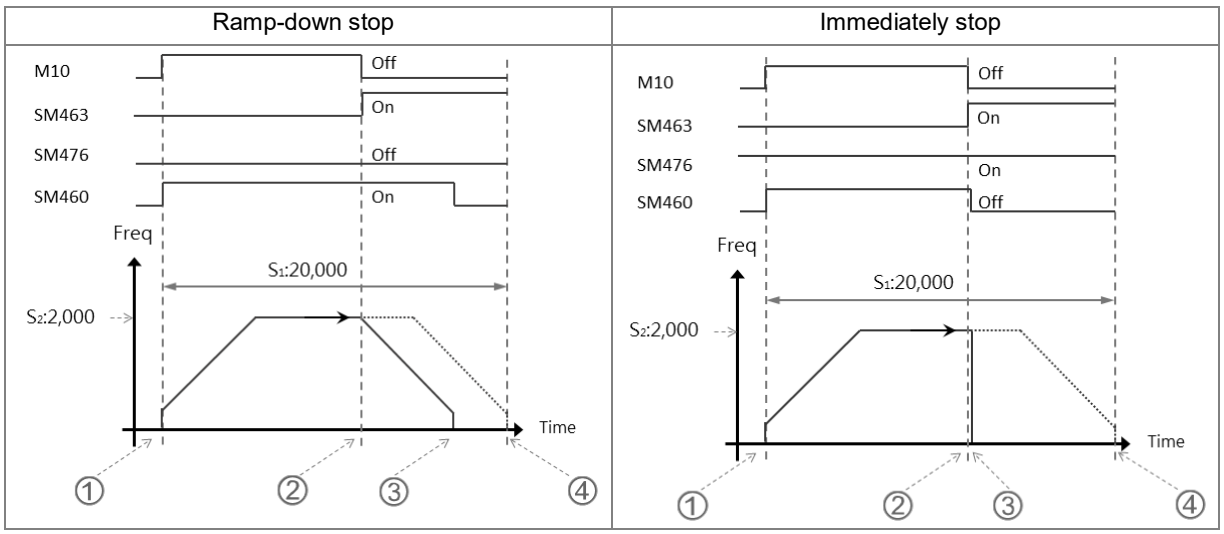
d) The general slope is determined by the start/end frequency, target frequency, and ramp up and down time.

The fixed slope is determined by the start/end frequency, target frequency of the fixed slope, and ramp up and down time.

e) When the target frequency of the fixed slope is less than the target frequency, the fixed slope function is not started.

4. Stop flag and immediately stop flag

a) The following graphic shows the output diagram based on example 1.



b) Ramp-down stop

① When M10 is ON, the DDRVI instruction is executed and starts to output pulses, and SM460, the Busy flag is ON.

② When the instruction is disabled (M10 is OFF) or the stop flag SM463 is ON, the state of the immediately stop flag SM476 is confirmed.

③ If the immediately stop flag SM476 is OFF, the output ramps down to stop and then the Busy flag SM460 is OFF.

④ If M10 is not OFF and the stop flag SM463 is not ON after the instruction is executed, Y0 stops the output after 20,000 pulses.

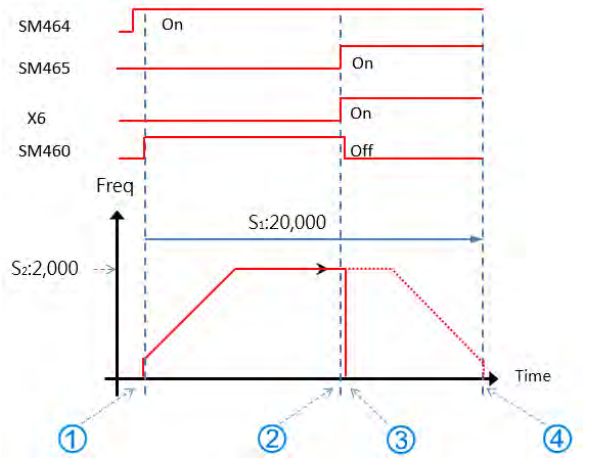
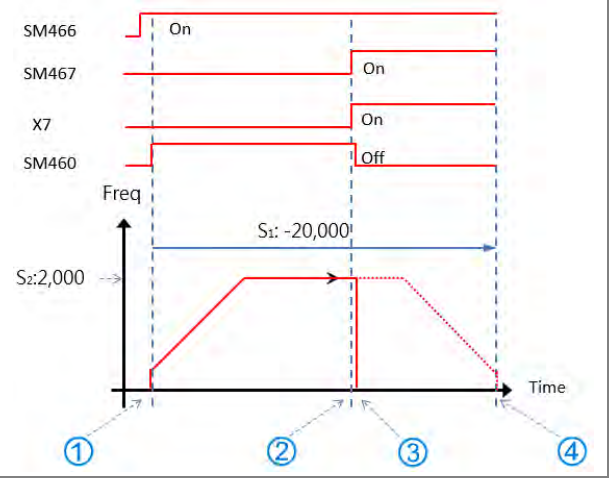
c) Immediately stop

② If the instruction is disabled (M10 is OFF) or the stop flag SM463 is ON, the state of the immediately stop flag SM476 is confirmed.

③ If the immediately stop flag SM476 is ON, the output stops immediately and the Busy flag SM460 is OFF.

5. Hardware limit function

a) The following graphic shows the output curve diagram of axis 1 (Y0/Y1).

Positive limit function	Negative limit function
	

b) Operation of the positive limit

⓪ Positive limit input point X6 is set in HWCONFIG and downloaded to the PLC. When SM464 is set to ON, the positive limit function starts.

① DDRVI outputs 20,000 pulses in the positive direction. **Note:** the instruction does not output in the positive direction if the positive limit point is ON when the function starts.

② When the limit input point X6 is triggered by the external mechanism and is ON, the condition for the hardware positive limit function is met.

③ DDRVI instruction stops the output immediately and the positive limit alarm SM465 is ON.

④ If the condition for the hardware limit function is not met, Y0 stops the output after 20,000 pulses.

c) Operation of the negative limit

⓪ Negative limit input point X7 is set in HWCONFIG and is downloaded to PLC. When SM466 is set to ON, the negative limit function starts.

① DDRVI outputs 20,000 pulses in the negative direction. **Note:** the instruction does not output in the negative direction if the negative limit point is ON when the function starts.

② When the limit input point X7 is triggered by the external mechanism and is ON, the condition for the hardware negative limit function is met.

③ DDRVI instruction stops the output immediately and the negative limit alarm SM467 is ON.

④ If the condition for the hardware limit function is not met, Y0 stops the output after -20,000 pulses.

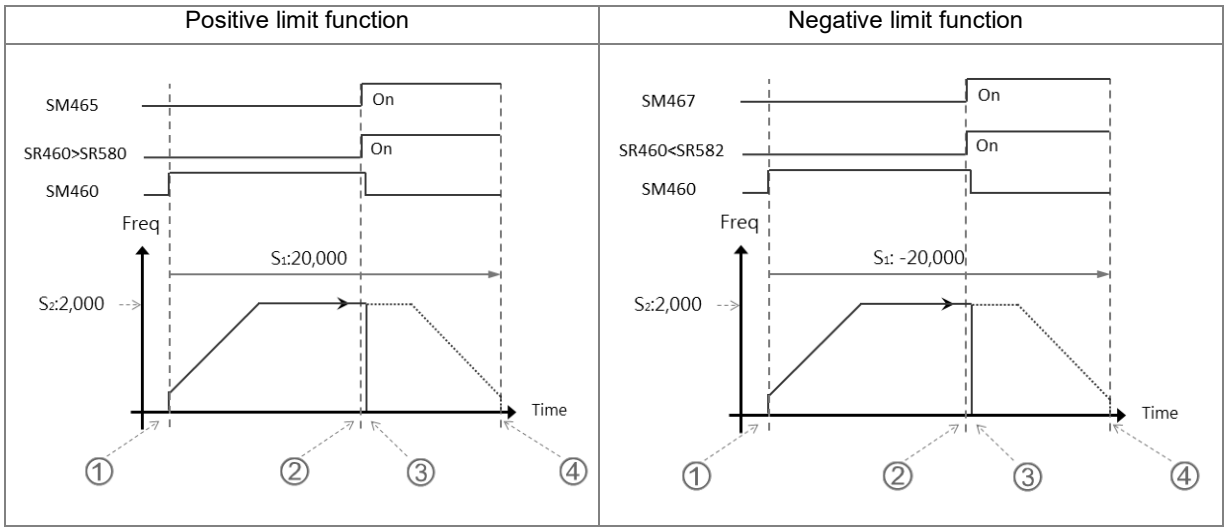
d) SM flags related to the limit function

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Enabling the hardware positive limit	SM464	SM484	SM504	SM524
Positive limit alarm flag	SM465	SM485	SM505	SM525
Enabling the hardware negative limit	SM466	SM486	SM506	SM526
Negative limit alarm flag	SM467	SM487	SM507	SM527

Note: The limit point alarm flag is read-only and is set or reset automatically by the PLC according to the state of the start flag.

6. Software limit function

a) The following diagram shows the output curve diagram of axis 1 (Y0/Y1).



b) Operation of the positive limit

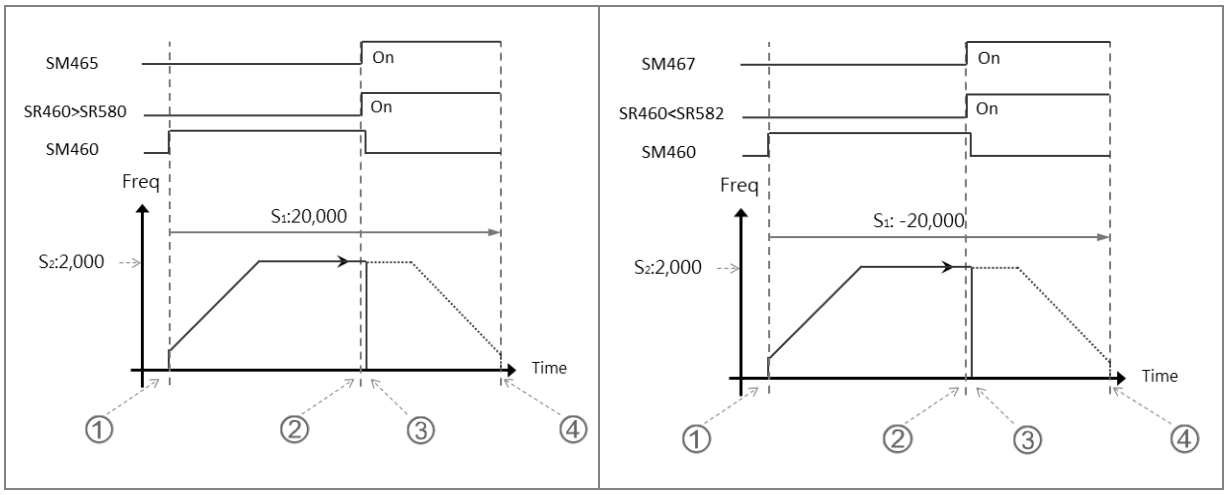
⓪ Positive software limit position is set in HWCONFIG (by setting SR580=11,000 for example). After downloading the setting to the PLC, it means the positive limit function starts.

① DDRVI starts to output 20,000 pulses in the positive direction. **Note:** the instruction does not output in the positive direction if the positive limit is exceeded when the function starts.

② When the current position SR460 > the limit position SR580, it means that the condition for the software limit function is met.

③ DDRVI instruction stops the output immediately and the positive limit alarm SM465 is ON.

④ If the condition for the limit function is not met, Y0 stops the output after 20,000 pulses.



c) Operation of the negative limit

⓪ Negative software limit position is set in HWCONFIG (by setting SR582 = –11,000 for example). After downloading the setting to the PLC, the negative limit function starts.

① DDRVI starts to output 20,000 pulses in the negative direction. **Note:** the instruction does not output in the negative direction if the negative limit is exceeded when the function starts.

② When the current position SR460 < the limit position SR582, the condition for the software negative limit function is met.

③ DDRVI instruction stops the output immediately and the negative limit alarm SM467 is ON.

④ If the condition for the software limit function is not met, Y0 stops the output after -20,000 pulses.

d) Rising-edge trigger

Limit function	Contact A	Contact B
Types	Rising-edge trigger	Falling-edgetrigger
Descriptions	 When started: from low to high and maintains at high	 When started: from high to low and maintains at low

e) SR related to the limit function

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Positive software limit position	SR580 SR581	SR584 SR585	SR588 SR589	SR592 SR593
Negative software limit position	SR582 SR583	SR586 SR587	SR590 SR591	SR594 SR595

Positive output limit: the output stops immediately if the current position is greater than the positive limit position.

Negative output limit: the output stops immediately if the current position is less than the negative limit position.

When the positive and negative limits are both 0, the software limit function cannot start. Since the software limit function checks the output position when the output instruction is scanned, stopping the output is affected by the PLC scan. To quickly stop the output in real time, use the external input point as the limit point.

7. **ST Example**

```
0001 IF  M0  THEN
0002     DDRVI(1000,1000,Y0,Y1);
0003     M1  :=TRUE;
0004 ELSIF  M1 AND  SM461  THEN
0005     SM470:=TRUE;
0006     M1:=FALSE;
0007 END_IF;
```

Explanation

7.1 When M0 is ON, Y0 outputs 1000 pulses at 1 kHz.

7.2 When the pulse output completes, SM461 is ON and SM470=ON is triggered.

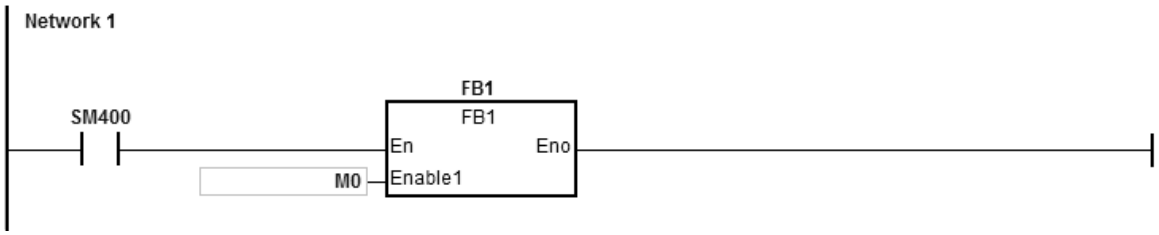
7.3 The pulse output restarts when M0 changes from OFF to ON again

7.4 When there are more than two high-speed output instructions in a program, do not use the variable name, M1, repeatedly.

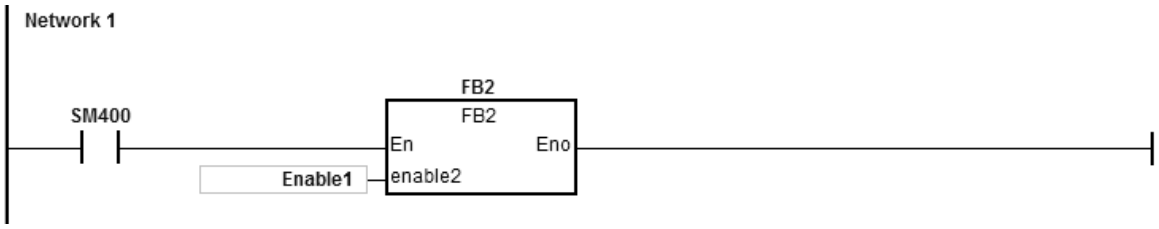
7.5 It is suggested not to use ST language in a program if there is any chance the outputting needed to be stopped.

8. **Function block example**

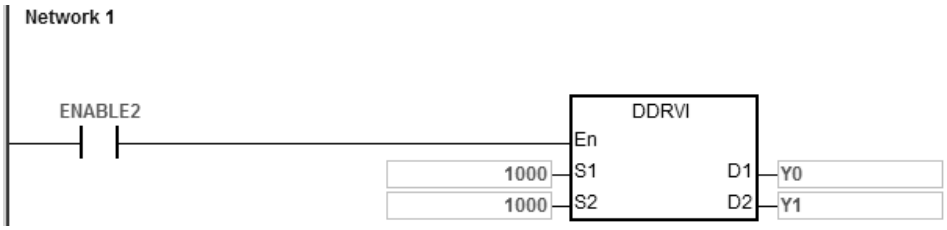
Main program



Function block 1



Function block 2



Explanation

When M0 is ON, Y0 outputs 10000 pulse at 1kHz. Y1 = OFF means the pulse output direction is positive.

9. Refer to PLSR instruciton (API2701) for more details on changing target frequency during outputting.
10. Changing the target position during outputting:

10.1 You cannot change outputting direction

even after the target position is changed. For example, if the original output target position is in the positive direction, after the target positon is changed, it still outputs in the positive positive.

10.2 After you set the change flag, the changing happens when the instruction is being scanned. And after it is changed, the change flag is cleared off. But if the target position is set incorrectly, the change flag keeps ON.

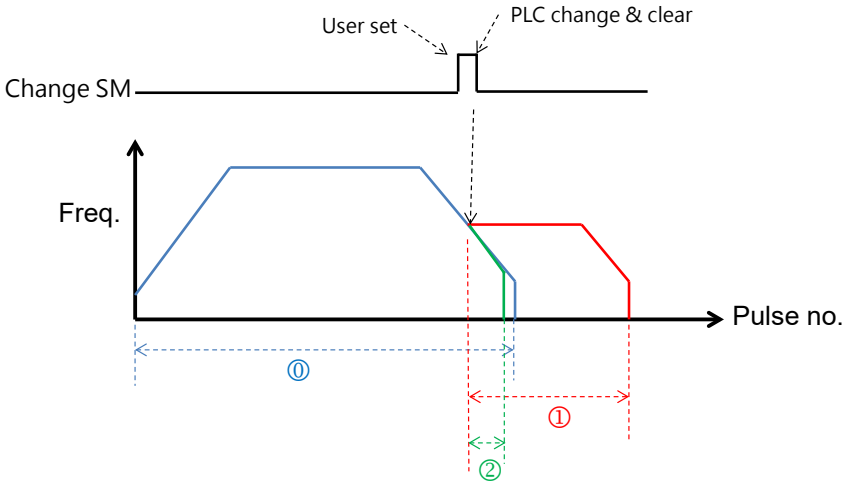
10.3 Refer to the following table for the corresponding outputs and change flag SM.

Output point	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
SR number	SM478	SM479	SM498	SM499	SM518	SM519	SM538	SM539

10.4 This changing target posiiton funciton is available for DDRVI and DDRVA insturcitos. When the instruction DDRVI is used and a change flag is set, the change will be executed. Even if the output number is 0, the operation will be carried out and stop outputting. When the instruction DDRVA is used, if the changed target position is the same as the previous one, this change will not be performed.

10.5 Three situations may occur, when changing target position during output.

Situation A: changing timing occurs during outputting in the ramp-down area.
⓪ oringinal target position when started (blue)
① new number of output pulse >= remaining number of output pulse, its outputting curve (red).
② new number of output pulse < remaining number of output pulse, its outputting curve (green).

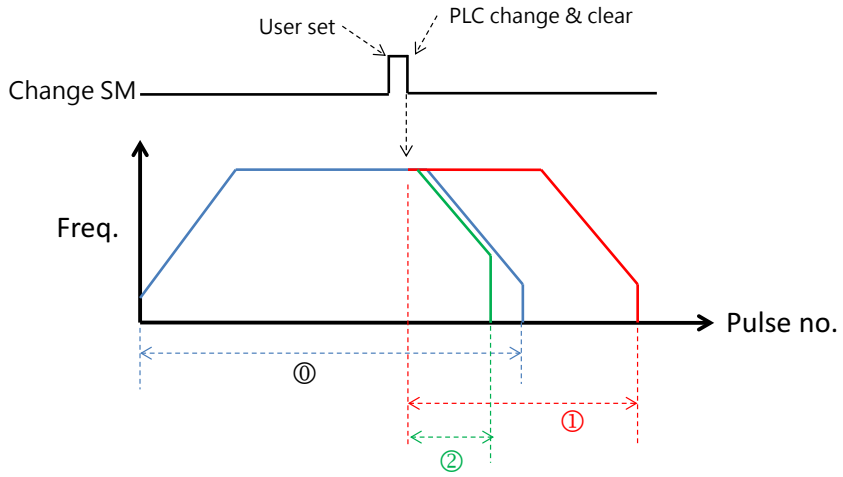


Situation B: changing timing occurs during outputting in the target frequency area.

⓪ oringinal target position during initial start (blue)

① new number of output pulse >= number of output pulse in ramp-down area, its outputting curve (red).

② new number of output pulse < number of output pulse in ramp-down area, its outputting curve(green)



Situation C: changing timing occurs during outputting in the ramp-up area.

⓪ oringinal target position during initial start (blue)

① new number of output pulse >= number of output pulse in ramp-down area, its outputting curve (red).

② new number of output pulse < number of output pulse in ramp-down area, its outputting curve(green)

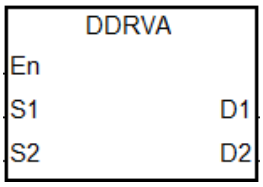
API	Instruction			Operand	Description
2707	D	DRVA		**S_1, S_2, D_1, D_2**	Absolute position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Number of output pulses (absolute positioning)

S_2 : Pulse output frequency

D_1 : Pulse output device

D_2 : Pulse direction output device

Explanation

1. This instruction specifies the output pulse setting for absolute positioning. **S_1** is the number of output pulses (absolute positioning). The range is between -2,147,483,648 to +2,147,483,647, and +/- signs indicate the positive and negative pulse directions. When the instruction is executed, it checks the current position with **S_1**. If **S_1** is 0, it indicates not outputting and output completion flag not be set to ON. If 0 is a possible output number in your program, it is suggested to add more conditions in your program to rule out this possibility.

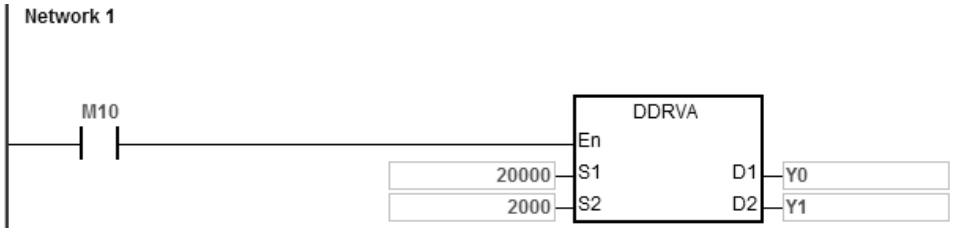
2. **S_2** is the pulse output frequency and the range is 0 Hz–200 kHz for the open collector output models. If the value in **S_2** is less than the Start/end frequency (Hz) set in SR (refer to the SR table of DRVI instruction for more details), PLC operates according to the values set in SR.

3. Absolute positioning means that the instruction outputs pulses to move from current position until the specified target position is reached. For example, the number of output pulses at current position is 100 and the number of pulses at the target position **S_1** is set to 1000. So the number of the actual output pulses is 1000-100=900.

4. Refer to the DDRVI instruction (API 2706) for more explanation.

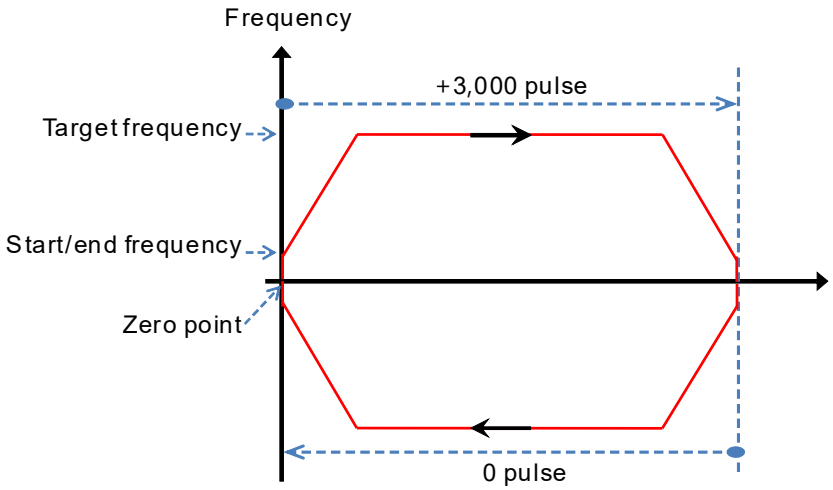
Example

If the value of the present output position SR460 (32-bit) is 100 and M10 is ON, then the DDRVA instruction causes Y0 to output pulses at 2 kHz until the value in SR460 becomes 20,000 (absolute positioning). Y1 = OFF means the positive direction.



Additional remarks

1. The following graph shows absolute positioning: the way of specifying the distance from the center (zero point).



API	Instruction	Operand	Description
2708	CSFO	**S_1, S_2, S_3, S_4, S_5, D_1, D_2**	Catch speed and proportional output

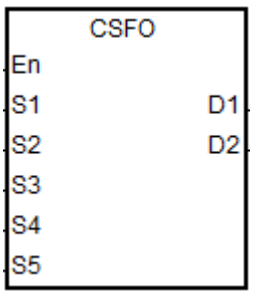
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	○															
S_2								●								
S_3								●								
S_4								●								
S_5								●								
D_1		○														
D_2								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1	●												
S_2			●				●						
S_3			●				●						
S_4		●											
S_5		●											
D_1	●												
D_2			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	-

6

Symbol



S_1 : Signal input device

S_2 : Number of input pulses

S_3 : Pulse input frequency

S_4 : Denominator (output frequency) for the proportion of the input frequency and output frequency

S_5 : Numerator (input frequency) for the proportion of the input frequency and output frequency

D_1 : Pulse output device

D_2 : Pulse output frequency

Explanation

1. This instruction specifies the catch speed and proportional output. **S_1** can only specify X0, X2, X4, X6, X10 and X12 as the input points, and the operand occupies two consecutive input points. You cannot execute the instruction if the input points are the points specified for **S_1** above. After you select the input points, the high-speed counter is automatically specified. If there is a DCNT instruction (API 1004) with the same high-speed counter in the program,

the PLC first executes the instruction that starts the counter first. The input points and corresponding high-speed counters are shown in the following table.

Group number	1	2	3	4	5	6
S_1+0 input point (Phase A)	X0	X2	X4	X6	X10	X12
S_1+1 input point (Phase B)	X1	X3	X5	X7	X11	X13
High-speed counter number	HC202	HC206	HC210	HC214	HC218	HC222
Flag for reversing input direction	SM270	SM271	SM272	SM273	SM274	SM275

2. If the high-speed counters for the instruction can use only the phase A/B input mode, set the flag for reversing the input direction to ON when MPG is connected but has not rotated yet, and the PLC input point is ON. Set the function to detect the state of PLC run cycle to OFF.

3. **S_2** is the number of input pulses. Use a 32-bit variable to declare the parameter.

4. **S_3** is the frequency of input pulses. Use a 32-bit variable to declare the parameter with the unit of 1 Hz.

5. **S_4** is the denominator (output frequency) for the proportion of the input frequency and output frequency. **S_5** is the numerator (input frequency) for the proportion of the input frequency and output frequency. The range of **S_4** and **S_5** is between 1–255. If the setting value exceeds the range, the instruction is processed as the maximum or minimum value. For example, if input frequency: output frequency= 5:3, the denominator output frequency is K3 and the numerator input frequency is K5. If the input frequency: output frequency=1:2, the denominator output frequency is K2 and the numerator input frequency is K1.

6. **D_1** can only specify Y0, Y2, Y4, and Y6 as the output points and occupies two consecutive output points.

   The output points and corresponding output mode SR are shown in the following table.

Output axis number	1	2	3	4
D_1+0 output point	Y0	Y2	Y4	Y6
D_1+1 output point	Y1	Y3	Y5	Y7
Output mode	SR462	SR482	SR502	SR522

7. **D_2** is the frequency of the output pulses. Use a 32-bit variable to declare the parameter with the unit of 1 Hz.

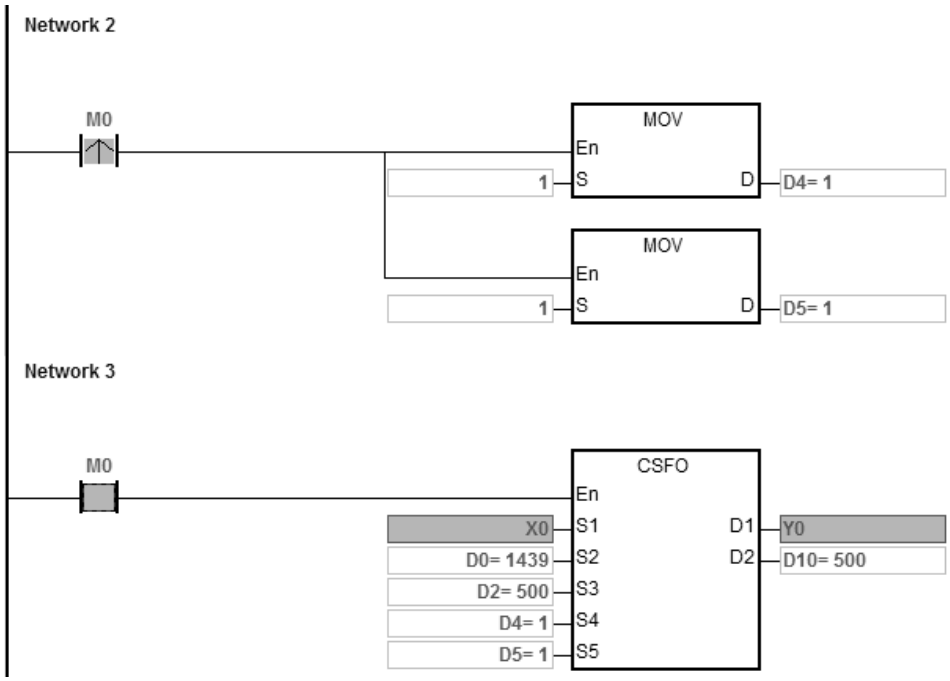
8. There is no limit to the number of times you can use this instruction; but the high-speed input and output points cannot be used by other instructions when this instruction is executed. Otherwise, the instruction cannot be executed.
9. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Notes

1. The PLC calculates the input pulse frequency based on the input pulse width (ON) in the positive half cycle. If the proportion of the pulse width for ON:pulse width for OFF is not 1:1, the PLC takes the ON width as the standard for conversion by default.
2. The input pulse=ON means the input point LED is on. When using the MPG but it has not rotated yet, you can check if the input point LED is OFF and the flag for reversing the input direction is activated .
3. The minimum pluse frequence for input and output is 50 Hz. If the actual input pulse is 10 Hz and the value in **S_3** shows 10 Hz. But since the minimum pulse frequence for output is 50 Hz, the instruction uses 50 Hz instead of 10 Hz to calculate. For the ratio 1:2, after conversion, the output pulse is 100 Hz. For the ratio 2:1, after conversion, the output pulse is 50 Hz instead of 25 or 5 Hz.
4. You can modify values in **S_4** or **S_5** during the execution of this instruction. But you need to re-execute this instruction to actualize this modification. Otherwise the value in **S_2** is not accurate enough to be used for calculating the current output position of SR.
5. After conversion, if the output pulse frequecne exceeds the maximum limit, the error code is recorded as 16#2030 in Error Log File in ISPSoft and SM29 is ON. SR29 is the axis number.
6. If the input pulse is 0 Hz and lasts for more than 2 seconds, PLC stops outputting automatically and the error code is recorded as 16#2031 in Error Log File in ISPSoft and SM29 is ON. SR29 is the axis number.

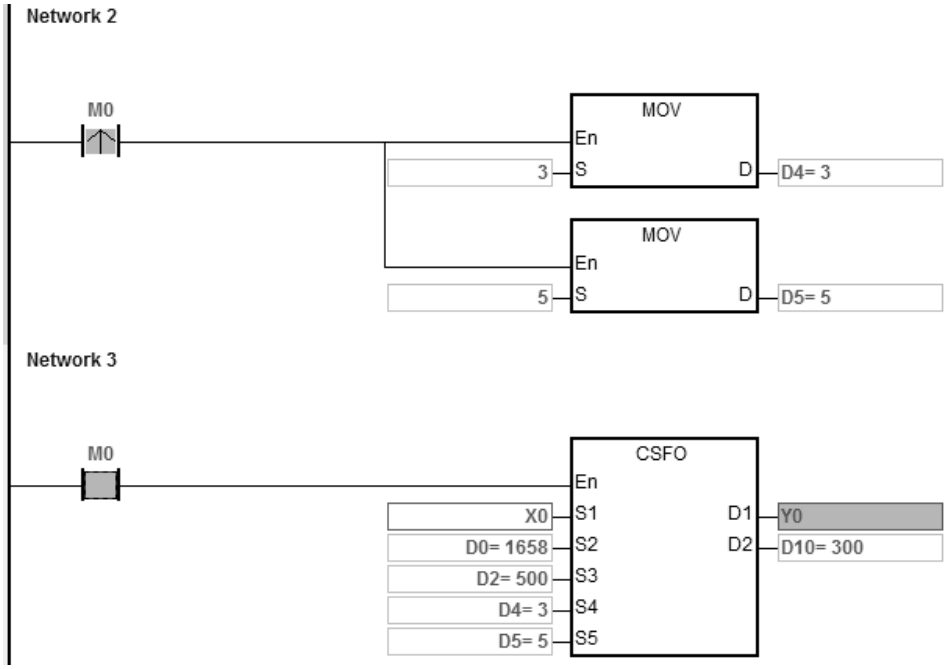
Example 1

X0/X1 input pulses for detecting the MPG. When M0 is ON, the setting values of **S_4** and **S_5** are both 1 (D4=1 and D5=1). When the input frequency is 500 Hz (D2, D3=500) and the number of pulses is 2500 (D0, D2=2500), the output axis (Y0/Y1) outputs 2500 pulses (SR460, SR461=2500) at 500Hz (D10, D11=500).



Example 2

When M0 is ON, the setting values of S4 and S5 are 3 and 5 (D4=3, D5=5) respectively. When the input frequency of the MPG is 500 Hz (D2, D3=500) and the number of pulses is 2500 (D2, D3=2500), the output axis (Y0/Y1) outputs 1500 pulses (SR460, SR461=1500) at 300 Hz (D10, D11=300).



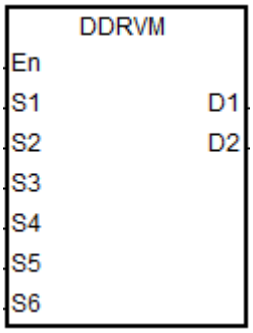
API	Instruction			Operand	Description
2709	D	DRVM		**S_1, S_2, S_3, S_4, S_5, S_6, D_1, D_2**	Mark alignment positioning

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3	○		○													
S_4								●	●				○	○		
S_5								●	●				○	○		
S_6								●	●				○	○		
D_1		○														
D_2		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3	●												
S_4			●				●						
S_5			●				●						
S_6			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Target number of output pulses

S_2 : Target output frequency

S_3 : Input point number for receiving the external interrupt signal

S_4 : The number of output pulses at the end for the fore mask section

S_5 : The number of output pulses at the beginning for the rear mask section

S_6 : The number of output pulses in the ramp-down process after the interrupt signal is received

D_1 : Pulse output device

D_2 : Direction pulse output device

Explanation

1. This instruction performs mark alignment positioning. When the value of **S_1** is 0, it indicates the output is a maximum 32-bit value (with a sign) and the output does not perform the ramp-down stop until receiving the mark signal.

If the value of **S_1** is not 0 and the external interrupt does not occur, the target number output pulses are output. The range of **S_1** is between -2,147,483,648 to +2,147,483,647, and the +/- sign means the positive or negative direction.

2. When the value of **S_2** is less than or equal to 0, the output is not enabled. If the output frequency is greater than the maximum frequency, the PLC processes it as the maximum output frequency.

3. **S_3** can specify the X0~X7 and X10~X17 device as the input point. The input source signal is affected by the scan cycle if the selected input point is not in M0–X15 in the PLC.

4. You must use the instruction by combining the input point number and the external interrupt program to achieve the real-time ramp-down output. For the rising-edge or falling-edge trigger for the external interrupt, select the triggers in the external interrupt program in ISPSoft.

   A. Using the external interrupt (e.g. I0xx, I1xx): when the external interrupt occurs, the ramp-down stop is performed after the mark signal is received. No operation is performed if the external interrupt is not enabled.

   B. Without using the external interrupt (I0xx, I1xx): When the instruction is executed, the ramp-down stop is performed after the mark signal is received if the rising edge occurs at X point or M device. It is affected by the scan time.

5. The number of output pulses in the fore mask section is between 1–**S_4**. When the setting value for **S_4** is 0, it means the fore- mask section function is disabled. When the number of pulses to be masked exceeds the value of **S_1**, the instruction takes the target number of output pulses **S_1** as the number to be masked. If the external input trigger occurs within the number of output pulses to be masked, the external input interrupt is automatically invalid.

6. The number of output pulses in the rear mask section is between **S_5** and **S_1**. When **S_5** =0 or **S_5** >=**S_1**, the rear mask section function is disabled. When the external input trigger occurs in the masked sections, the external input interrupt is automatically invalid. When **S_4** >**S_5**, it indicates the external input interrupt is invalid in the output process.

7. If the fore and rear mask sections are both set and **S_4** < **S_5**, the valid input interrupt occurs in the section between **S_4**+1 and **S_5**-1.

8.

S_6: the number of output pulses in the ramp-down	Description
If **S_6** is set to 0 after the mark signal is received	the ramp-down stop is performed based on the ramp-down time
If **S_6** =-1 or < 0 after the mark signal is received	the output immediately stops
If the setting value of **S_6** is not enough to achieve the ramp-down stop within the ramp-down time,	The instruction limits the target frequency and performs the ramp-down stop in accordance with the set

	ramp-down time
If **S_6** is greater than the number of pulses output within the ramp-down time,	The instruction outputs the number of redundant pulses with the same output frequency when the interrupt is triggered, and then performs the ramp-down process

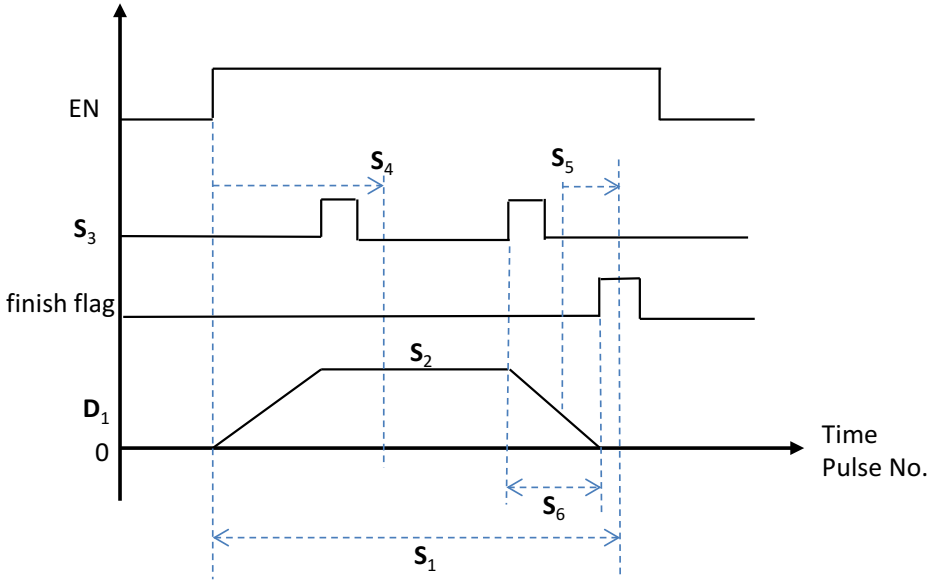
9. For PLC with FW V1.06.00 (V1.06.00 included) or later: If **S_6** is set to 0 after the mark signal is received, it means that the ramp-down stop is performed based on the ramp-down time. If **S_6** is set to -1 after the mark signal is received, it means that the output immediately stops. If **S_6** is set to >0 after the mark signal is received and if the setting value of **S_6** is not enough to achieve the ramp-down stop within the ramp-down time, the instruction limits the target frequency and performs the ramp-down stop in accordance with the set ramp-down time. If **S_6** is greater than the number of pulses output within the ramp-down time, the instruction outputs the number of redundant pulses with the same output frequency when the interrupt is triggered, and then performs the ramp-down process.

S_6: the number of output pulses in the ramp-down	Description
If **S_6** = > 0 after the mark signal is received If the setting value of **S_6** is not enough to achieve the ramp-down stop within the ramp-down time,	The instruction limits the target frequency and performs the ramp-down stop in accordance with the set ramp-down time
If **S_6** = > 0 after the mark signal is received If **S_6** is greater than the number of pulses output within the ramp-down time,	The instruction outputs the number of redundant pulses with the same output frequency when the interrupt is triggered, and then performs the ramp-down process
If **S_6** is set to 0 after the mark signal is received	the ramp-down stop is performed based on the ramp-down time
If **S_6** =-1 after the mark signal is received	the output immediately stops
If **S_6** =< -1 after the mark signal is received If the setting value of **S_6** is not enough to achieve the ramp-down stop within the ramp-down time,	The instruction does NOT limit the target frequency and after outputting the number of output pulses, the output stops immediately
If **S_6** =< -1 after the mark signal is received If **S_6** is greater than the number of pulses output within the ramp-down time,	The instruction outputs the number of redundant pulses with the same output frequency when the interrupt is triggered, and then performs the ramp-down process

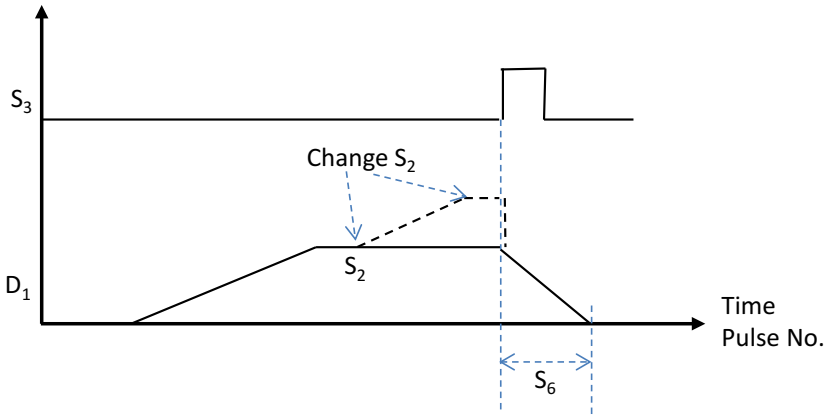
- Added new SR652, SR653 for displaying number of pulse in ramp-up area and new SR654, SR655 for displaying number of pulse in ramp-down area.

10. If the output has entered the ramp-down process when the external input interrupt trigger occurs, the instruction completes the output of the number of pulses specified by **S_6**.

11. See the following timing diagram about the output, interrupt trigger, mask and relevant flags. For the completion flag, refer to the Completion flag SM for the axes in DDRVI instruction (API 2706).



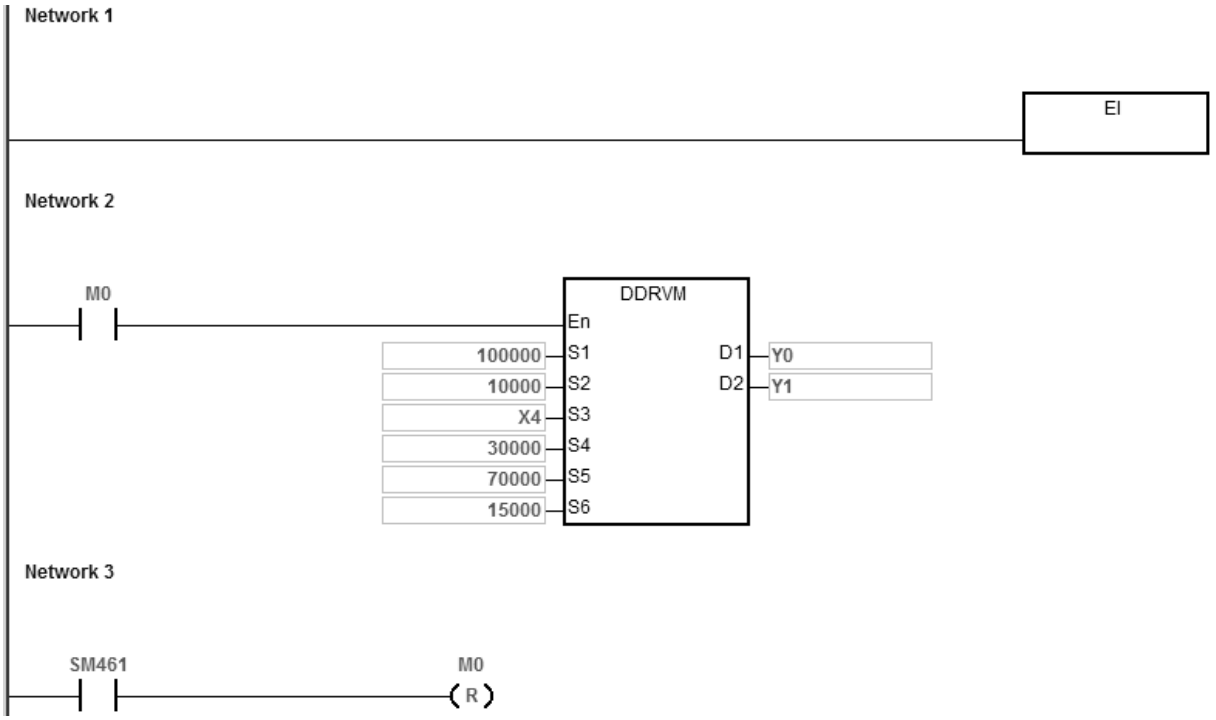
12. The target frequency **S_2** of the instruction can be modified in the process of the pulse output. Since the number of output pulses in the ramp-down section has been set when the instruction is executed, the timing diagram for changing the frequency is shown as the following dashed line.



13. Refer to the DDRVI instruction (API 2706) for the selection of pulse output devices for **D_1** and **D_2**. The output points for **D_1** can only be even-numberd such as Y0, Y2 or Y6. If the output points are not the recommended ones, the direction output **D_2** is affected by the scan cycle.

14. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example 1



X4 external interrupt program

Notes

1. When M0 changes from OFF to ON, Y0 starts to output pulses. After outputting more than 30,000 pulses, the external interrupt is detected on X4. The value in D0 increases by 1 and the pulse output stops after 15,000 pulses (S_6 = 15,000). If the interrupt does not occur, the output does not stop until 100,000 pulses are output.

2. If the number of output pulses is between 1–30,000, they are in the fore mask section. The external interrupt occurs at X4 at the moment and the instruction does not perform the ramp-down stop operation.

3. If the number of output pulses is between 70,000–100,000, they are in the rear mask section. The external interrupt occurs at X4 at the moment and the instruction still does not perform the ramp-down stop operation.

4. When the pulse output completes, SM461 is ON and M0 is reset.

Additional remarks

1. The alignment mark function and mask function:

① When DDRVM is executed, the pulse output starts.

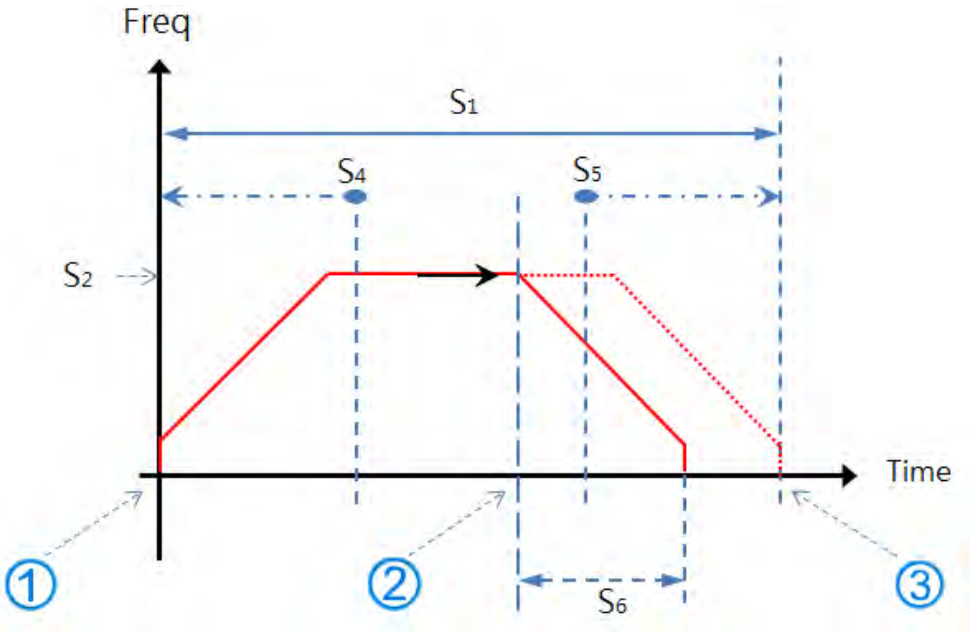
② When the interrupt occurs for the mark alignment, the ramp-down process starts and the output stops after the number of the pulses specified by **S_6**.

③ When the interrupt for the mark alignment does not occur or has no effect, DDRVM stops outputting the pulses after the target number of pulses specified by **S_1**.

S_4: The number of output pulses in the fore mask section. If the interrupt occurs in this section, the interrupt is ineffective for the mark alignment.

S_5: The number of output pulses in the rear mask section. If the interrupt occurs in this section, the interrupt is ineffective for the mark alignment.

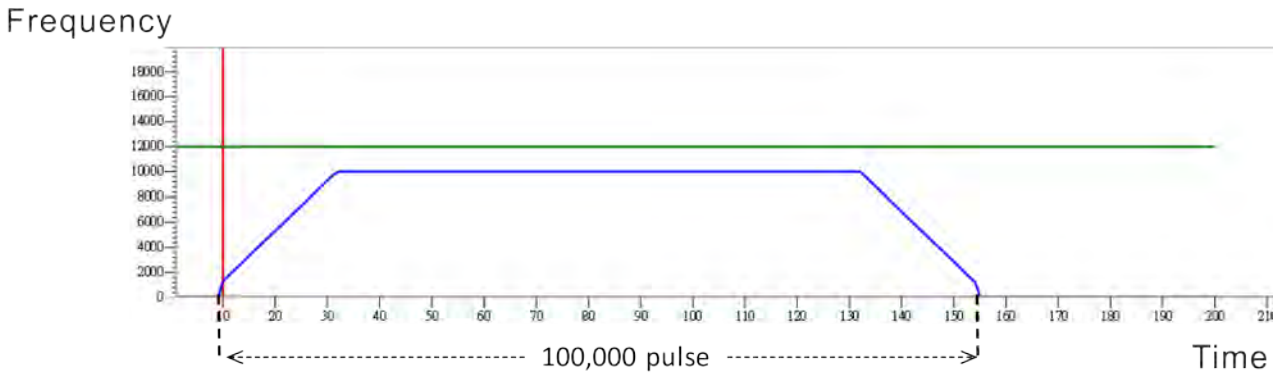
S_2: The target output frequency



2. The mark alignment function is enabled in the ramp-up and target frequency and ramp-down sections.

When the mark alignment function is not enabled, set **S_1** = 100,000; the output stops after 100,000 pulses.

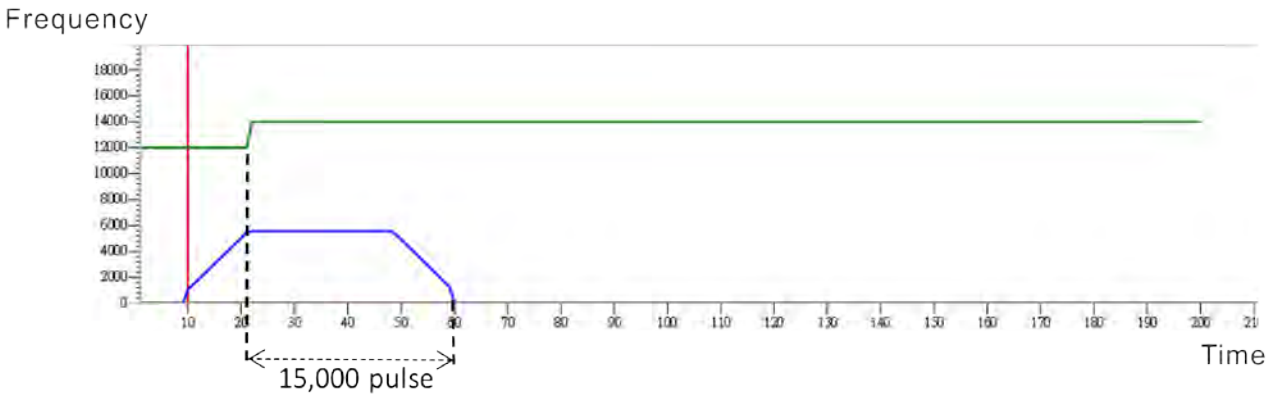
- Green line: signal of mark aligment
- Blue line: the actual output speed



Case 1: when the mark alignment occurs in the ramp-up section

- Green line: signal of mark aligment
- Blue line: the actual output speed

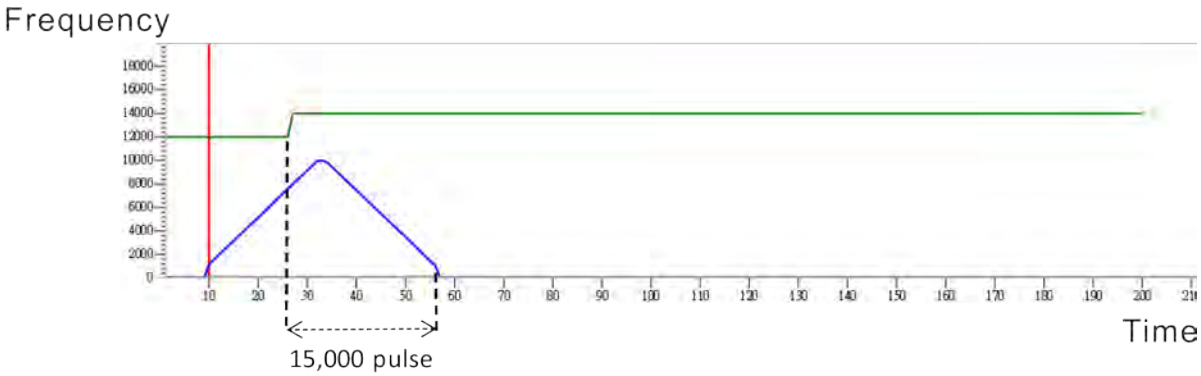
The operation of the mark alignment (green line: OFF -> ON) begins when the number of pulses is 3000. The target frequency of 10 kHz cannot be reached even if **S_6** is 15,000.



Case 2: when the mark alignment occurs in the ramp-up section

- Green line: signal of mark aligment
- Blue line: the actual output speed

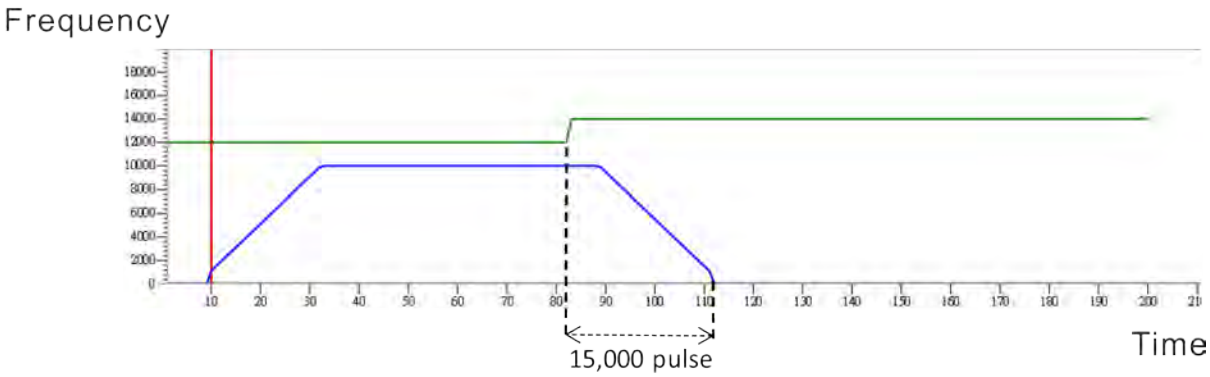
The operation of the mark alignment (green line: OFF -> ON) begins when the number of pulses is 6000. The target frequency of 10 kHz can be reached if **S_6** is 15,000.



When the mark alignment occurs in the target frequency section

- Green line: signal of mark aligment
- Blue line: the actual output speed

The operation of the mark alignment (green line: OFF -> ON) begins when the number of pulses is 50,000. Set **S_6** = 15,000; the output stops after 15,000 pulses.

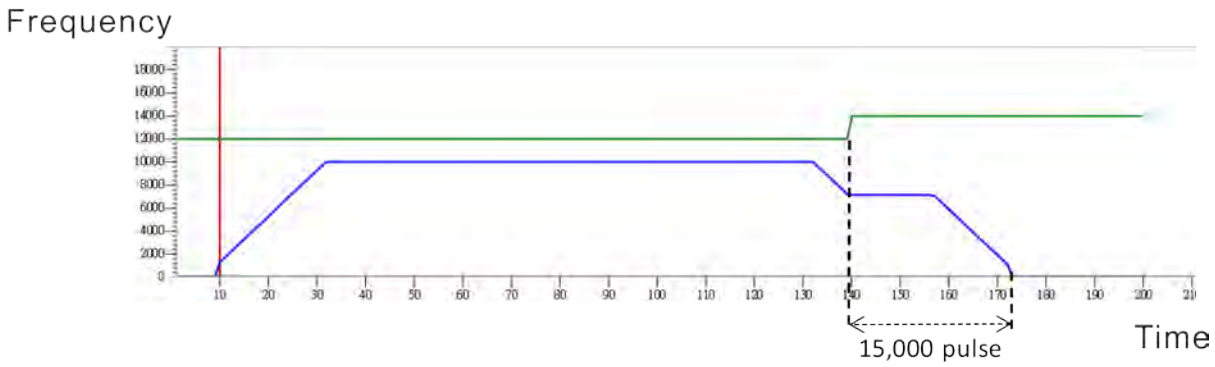


Case 1: when the mark alignment occurs in the ramp-down section

- Green line: signal of mark aligment
- Blue line: the actual output speed

The operation of the mark alignment (green line: OFF -> ON) begins when the number of pulses is 95,000.

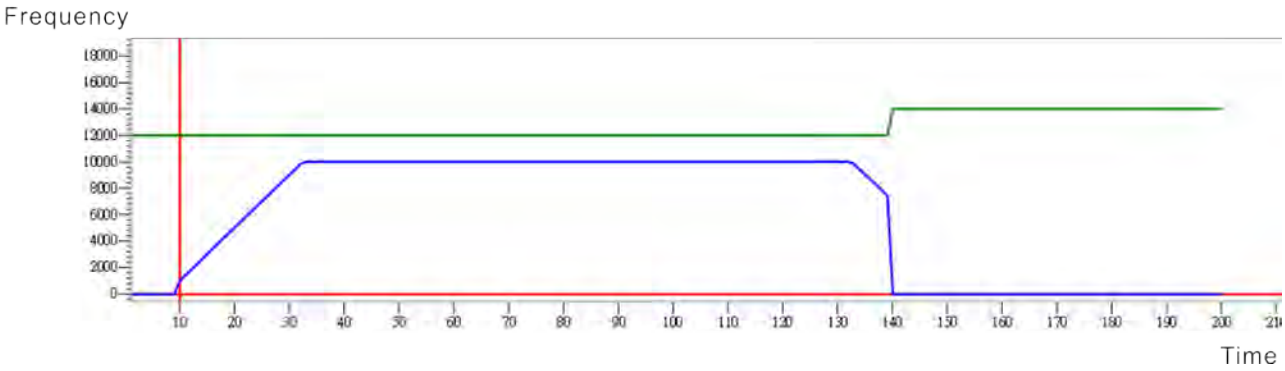
Set **S6** = 15,000; the output stops after 15,000 pulses at the current speed.



Case 2: when the mark alignment occurs in the ramp-down section

- Green line: signal of mark aligment
- Blue line: the actual output speed

The operation of the mark alignment (green line: OFF -> ON) begins when the number of pulses is 95,000. After the mark alignment, set **S6** to -1, the output stops immediately.



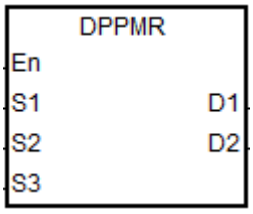
API	Instruction			Operand	Description
2710	D	PPMR		S_1, S_2, S_3, D_1, D_2	2-Axis relative-coordinate point-to-point synchronized motion

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Number of output pulses for the X axis

S_2 : Number of output pulses for the Y axis

S_3 : Maximum point-to-point output frequency

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. This instruction performs 2-axis relative-coordinate point-to-point synchronized motion. **S1** and **S2** specify the number of output pulses (relative positioning) for the X and Y axes. The range is between: -2,147,483,648 to +2,147,483,647, and the +/- sign indicates the positive/negative direction.

2. S_3 specifies the maximum point to point output frequency. The range is between 1 Hz–200 kHz.

3. D_1 and D_2 are the output devices for the X and Y axes respectively. You can designate the following six axes for output but you cannot change the fixed direction output points. The direction signal: OFF means in positive direction and ON is in negative direction. If you do not use the preset Pulse+direction output mode (default: 0), change the output mode by setting SR to 1.

Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Output point for **D_1**	Y0	Y2	Y4	Y6
Direction output point for **D_2**	Y1	Y3	Y5	Y7
SR number	SR462	SR482	SR502	SR522

4. The PLC assigns the value of **S_3** to the X or Y axis depending on which outputs the largest number of pulses. If the numbers of pulses output for the two axes are excessively different, so that no proper output frequency can be calculated through the numbers of their output pulses, the PLC automatically decreases the maximum point to point frequency and without any error message appearing.

5. The instruction uses the parameters such as the start and end frequencies and ramp-up and ramp-down times for the axes the instruction executes. Instead of the originally configured parameters for the axes, the X axis parameters are taken as the reference source. For example, when the X axis selects axis 1 and the Y axis selects axis 3 for the output, axis 1 parameters become the parameter sources for the start frequency and ramp up and ramp-down time.

6. The following table shows the flags for the axes and the corresponding SM/SR.

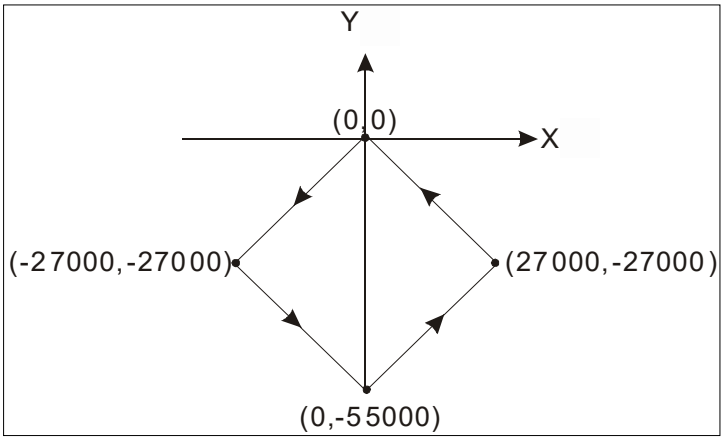
Axis number	Axis 1	Axis 2	Axis 3	Axis 4
Busy flag	SM460	SM480	SM500	SM520
Completion flag	SM461	SM481	SM501	SM521
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521
Start/end frequency	SR463	SR483	SR503	SR523
Ramp-up time	SR464	SR484	SR504	SR524
Ramp-down time	SR465	SR485	SR505	SR525

7. There is no limit to how many times you can use the instruction in a program. But during the execution of the instruction, it cannot execute the output (if the Y axis output is being used) until the Y axis completes its output and is released.

8. When the 2-axis synchronized pulse output completes, the corresponding Completion flags for the two axes is set. Since the same timing for the completion of the 2-axis synchronized pulse output cannot be ensured every time, the program should check the Completion flags of both axes before executing the next user program.

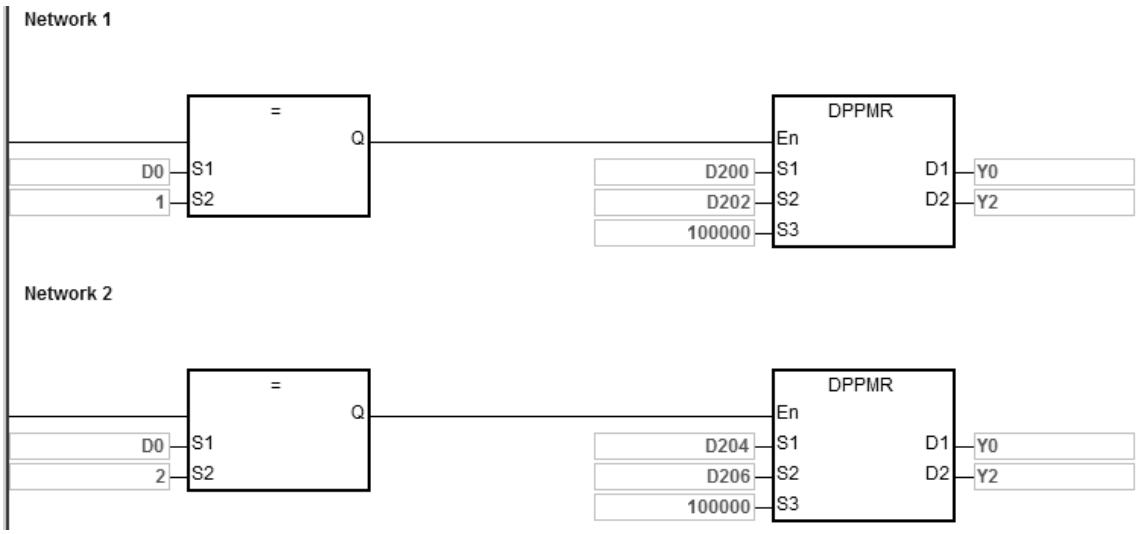
Example (Ladder diagrams)

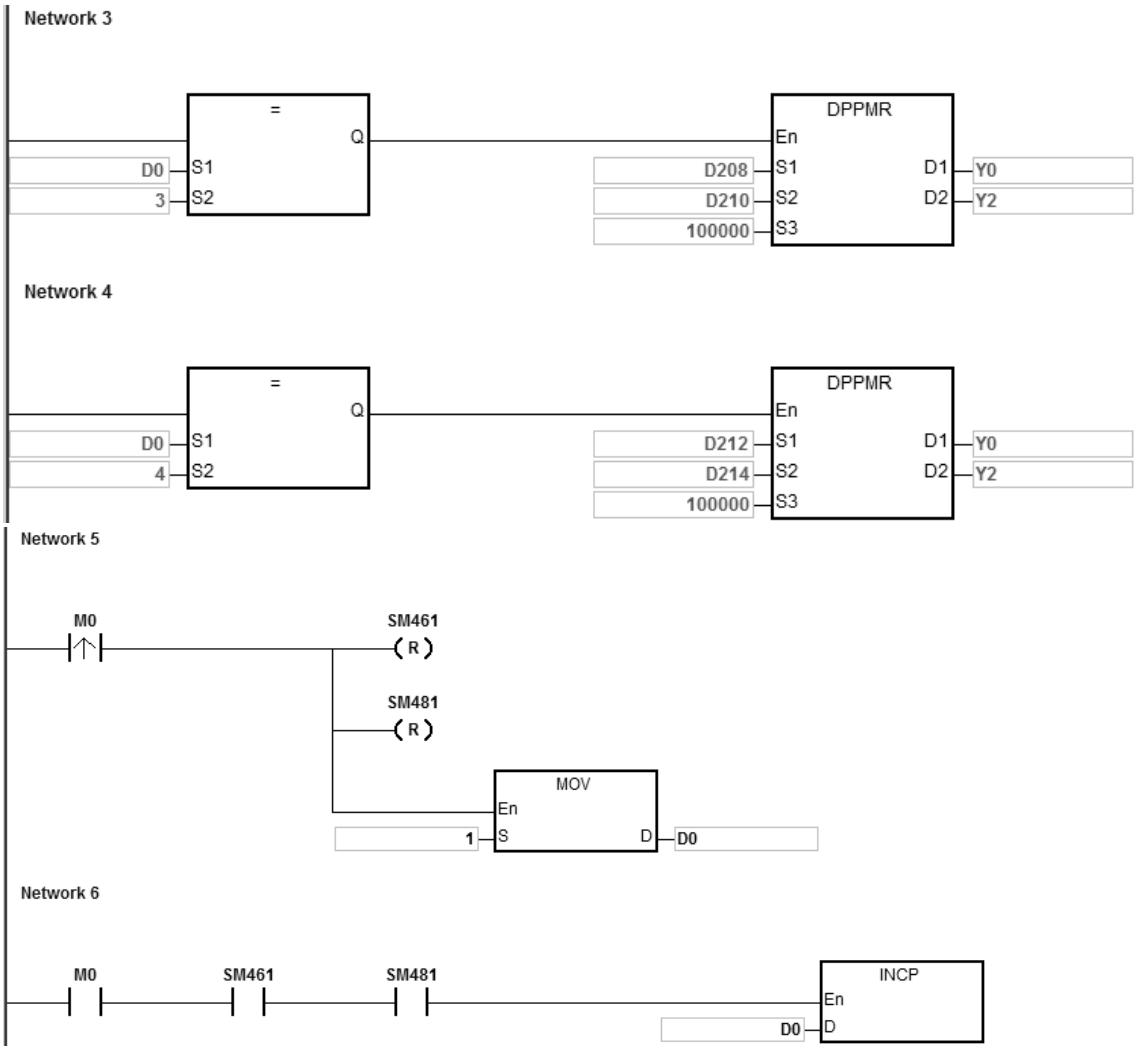
1. Draw a rhombus as shown below.



2. Steps:

- Set the coordinates of four points (0, 0), (-27000, -27000), (0, -55000), (27000, -27000) as in the figure above. Calculate the relative coordinates of the four points and obtain (-27000, -27000), (27000, -28000), (27000, 28000), and (-27000, 27000). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
- RUN the program in the PLC. Set M0 to ON to start the 2-axis line drawing.





3. Operation:

When PLC runs the program and M0 is ON, the PLC starts the first point-to-point motion at 100 kHz. D0 increments by 1 whenever a point-to-point motion is completed and the second point-to-point motion starts to execute automatically. The operation pattern repeats until the fourth point-to-point motion completes.

Example 2 (ST program)

```
0001 IF  M0  THEN
0002     DPPMR(1000,1000,1000,Y0,Y2);
0003     M1  :=TRUE;
0004 ELSIF  M1 AND  SM461  AMD  SM481  THEN
0005     SM470:=TRUE;
0006     SM490:=TRUE;
0007 M1:=FALSE;
0008 END_IF;
```

Explanation

1. When M0 is ON, Y0 and Y2 outputs 1000 pulses at 1kHz simultaneously. When M1 is ON, it indicates Y0 and Y2 are used for high-speed pulse output.

2. When the Y0 pulse output is completed, SM461 is ON. When the Y2 pulse output is completed, SM481 is ON. And if M1 is also ON, it triggers SM470 (output completion auto-reset for Y0) and SM490 (output completion auto-reset for Y2) to be ON. Y0 and Y2 are free and ready to be used again and M1 is cleared to OFF.

3. When M0 switches from OFF to ON, the pulse output is restarted.

4. When there are more than two high-speed output instructions in a program, do not use the variable name, M1, repeatedly.

5. It is suggested not to use ST language in a program if there is any chance the outputting needed to be stopped.

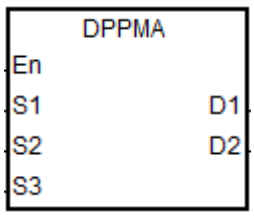
API	Instruction			Operand	Description
2711	D	PPMA		S_1, S_2, S_3, D_1, D_2	2-Axis absolute-coordinate point-to-point synchronized motion

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Number of output pulses for the X axis

S_2 : Number of output pulses for the Y axis

S_3 : Maximum point-to-point output frequency

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

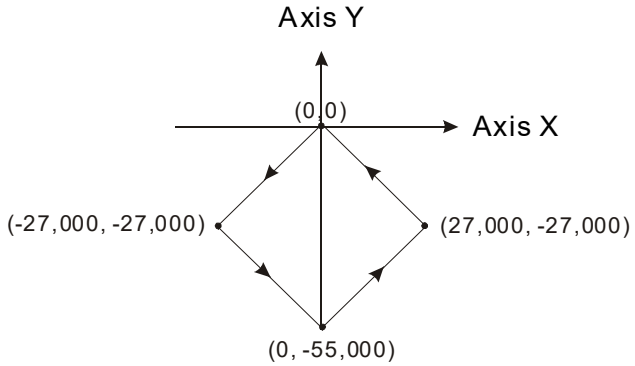
Explanation

1. This instruction performs 2-axis absolute-coordinate point-to-point synchronized motion. **S1** and **S2** specify the number of output pulses (absolute positioning) for the X and Y axes.

2. For more information, refer to the DPPMR instruction (API 2710).

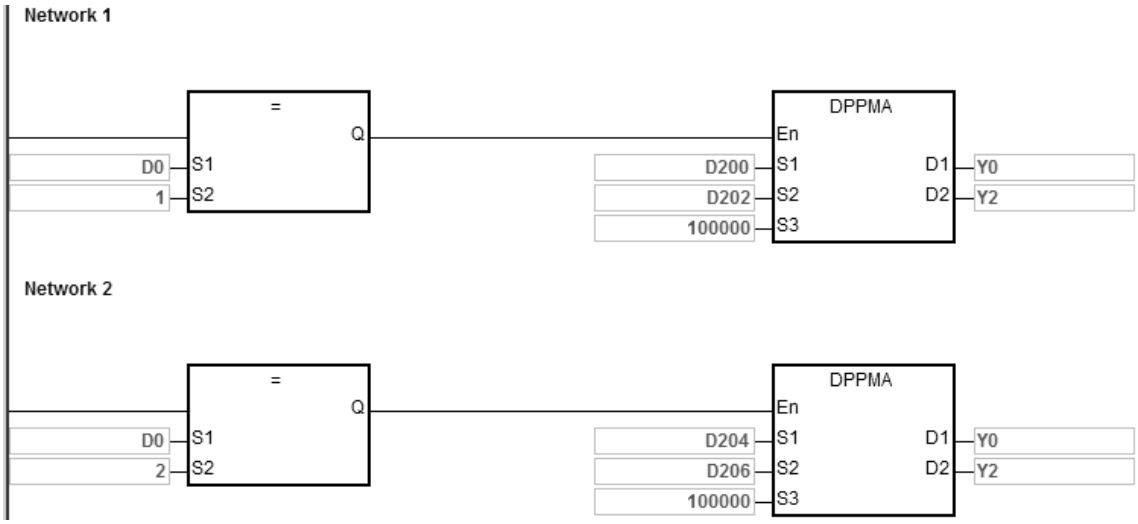
Example

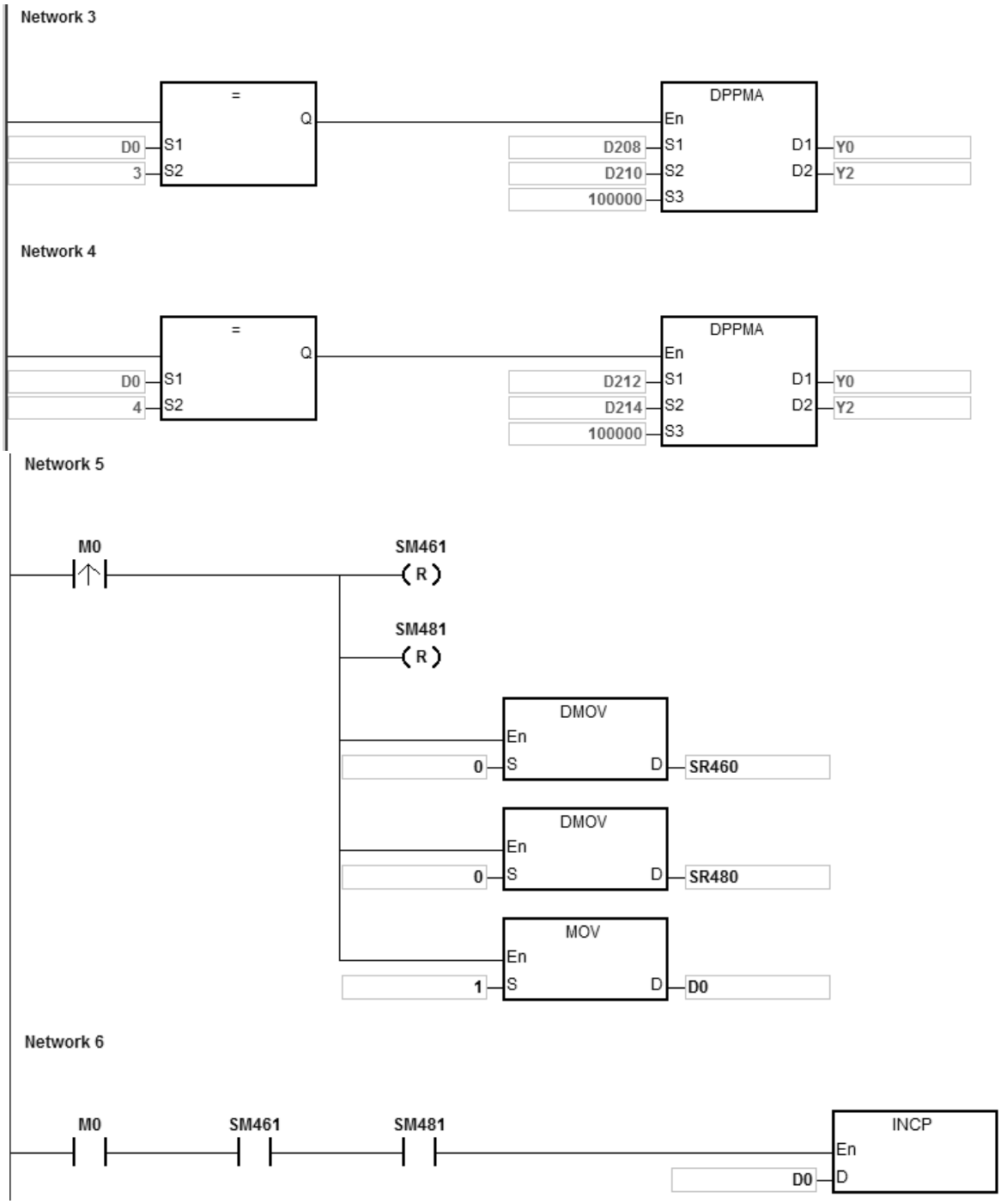
1. Draw a rhombus as in the figure below.



2. Steps:

- Set the four coordinates (-27000, -27000), (0, -55000), (27000, -27000) and (0, 0) as in the figure above. Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
- RUN the program in the PLC. Set M0 to ON to start the 2-axis line drawing.





3. Operation:

When PLC runs the program and M0 is ON, the PLC starts the first point-to-point motion at 100 kHz. D0 increments by 1 whenever a point-to-point motion completes and the second point-to-point motion starts to execute automatically. The operation pattern repeats until the fourth point-to-point motion completes.

API	Instruction			Operand	Description
2712	D	CICR		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis relative-position clockwise arc interpolation

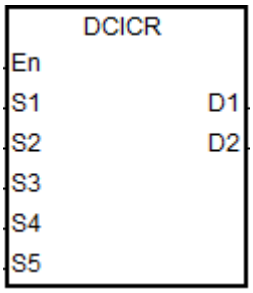
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		●
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●		●				
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



S_1 : X axis target coordinate (relative positioning)

S_2 : Y axis target coordinate (relative positioning)

S_3 : The shift of the center or the central angle

S_4 : Target reference frequency

S_5 : Function selection

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. S_1, S_2 and S_3 respectively designate the X axis target coordinate, Y axis target coordinate (relative position) and the shift of the center or a central angle of a circle. Refer to the following clockwise arc operation for details.

2. D_1 and D_2 are the pulse output devices for the X and Y axes respectively. Refer to the DPPMR instruction (API 2710) for the selection of output points and output modes for the axes.

3. **S_4** sets the target frequency for the reference. The target reference frequency is used for the prior calculation when the PLC plans the travel path for the arc after the instruction is executed. But if the estimated calculation process cannot achieve the arc travel path, the output frequency is automatically decreased to fulfill the synchronized arc drawing function.

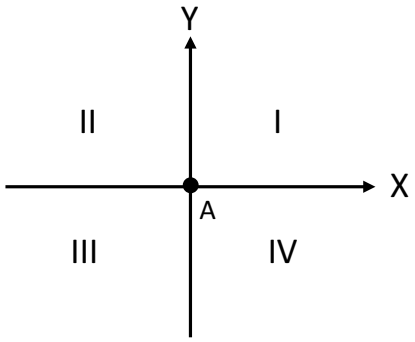
4. **S_5** is the setting value for function selection. See the descriptions below.

Setting value in **S_5**	**S_3** : The shift of the center or central angle	Description
0	Shift of the center (32-bit integer type)	The arc resolution uses a 10° arc as the basic angle for motion.
1	Shift of the center (32-bit integer type)	The arc resolution uses a 5° arc as the basic angle for motion.
2	Shift of the center (32-bit integer type)	The arc resolution uses a 1° arc as the basic angle for motion.
10	Shift of central angle (floating point type)	The arc resolution uses a 10° arc as the basic angle for motion.
11	Shift of central angle (floating point type)	The arc resolution uses a 5° arc as the basic angle for motion.
12	Shift of central angle (floating point type)	The arc resolution uses a 1° arc as the basic angle for motion.
Other	Shift of the center (32-bit integer type)	Seen as setting value in **S_5** is 0 The arc resolution uses a 10° arc as the basic angle for motion.

5. Refer to Example 2 of the DPPMR instruction for more details on ST language programming.

Drawing a clockwise arc

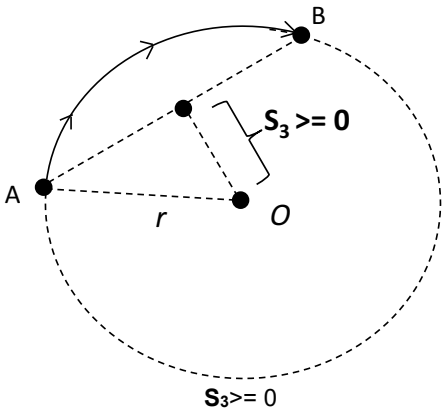
1. Define the four quadrants (QI, QII, QIII and QIV) of a coordinate system as in the graph below. Point A indicates the current coordinates on the X and Y axes.



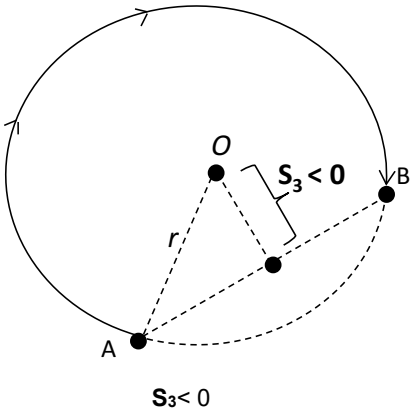
2. Point B is the target coordinates specified by **S1** and **S2**. Point O is the center of a circle enclosing both point A and point B.

3. **S3** is the distance to shift the center point O. **S3** can also be the distance to shift the center point or central angle, if it is defined as the central angle, it is the included angle of AOB.

4. The point A represents the origin of the axes X and Y in a plane. The target point B can be in Quadrant I, II, III, IV of the plane. Use the target point B to connect to the point A to draw a line and then use the setting values (shift the center point or central angle) in **S3** to define an arc travel path.
   The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant I and IV with different setting values in **S3** (shift the center point). (r: radius)

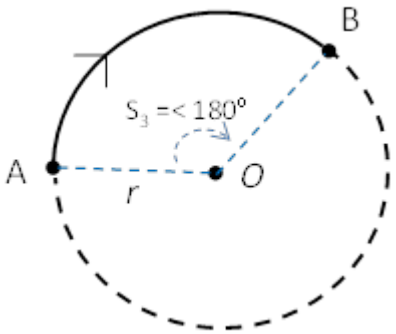


S3>= 0

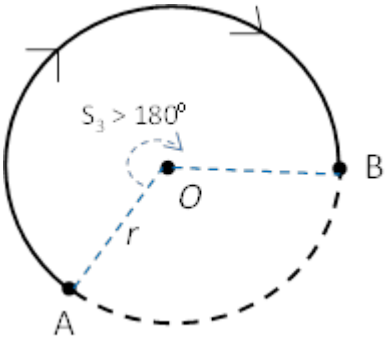


S3< 0

5. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant I and IV with different setting values in **S3** (shift the central angle). (r: radius)

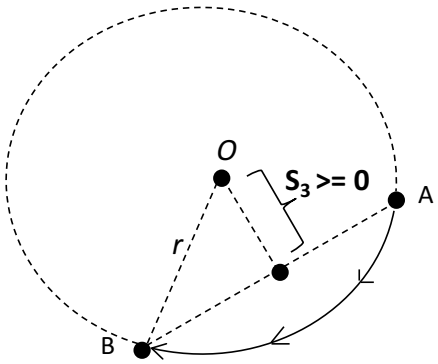


S3<= 180.0

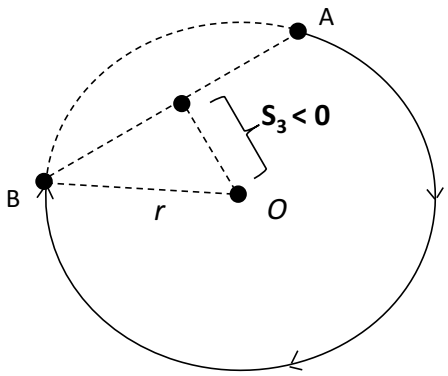


S3> 180.0

6. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant II and III with different setting values in **S3** (shift the center point). (r: radius)

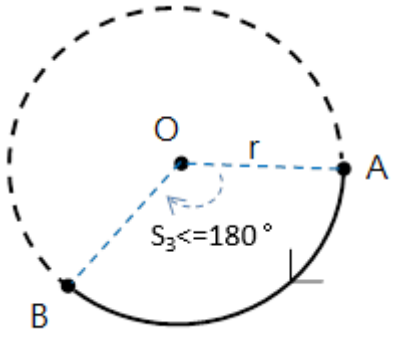


S3>= 0

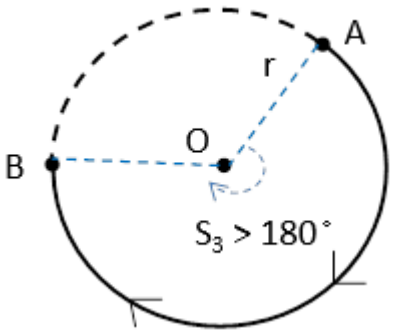


S3< 0

7. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant II and III with different setting values in **S3** (shift the central angle). (r: radius)



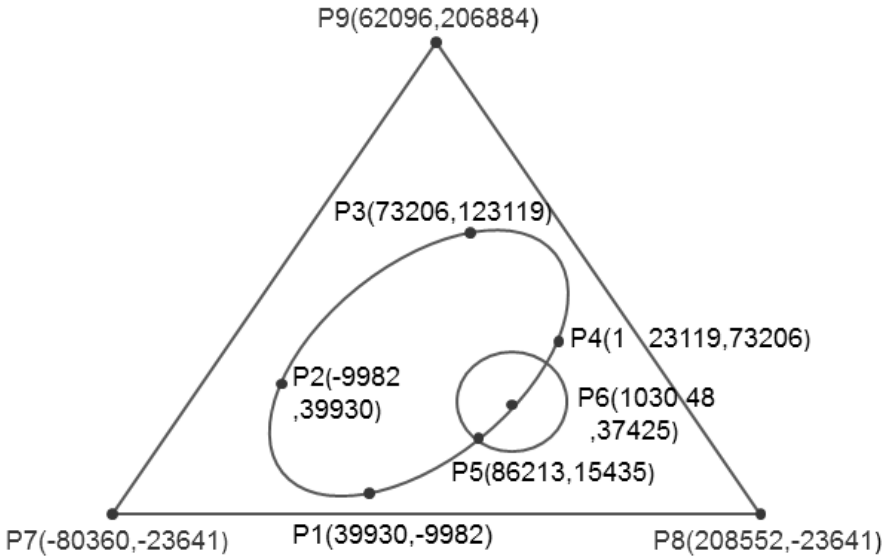
S3<= 180.0



S3> 180.0

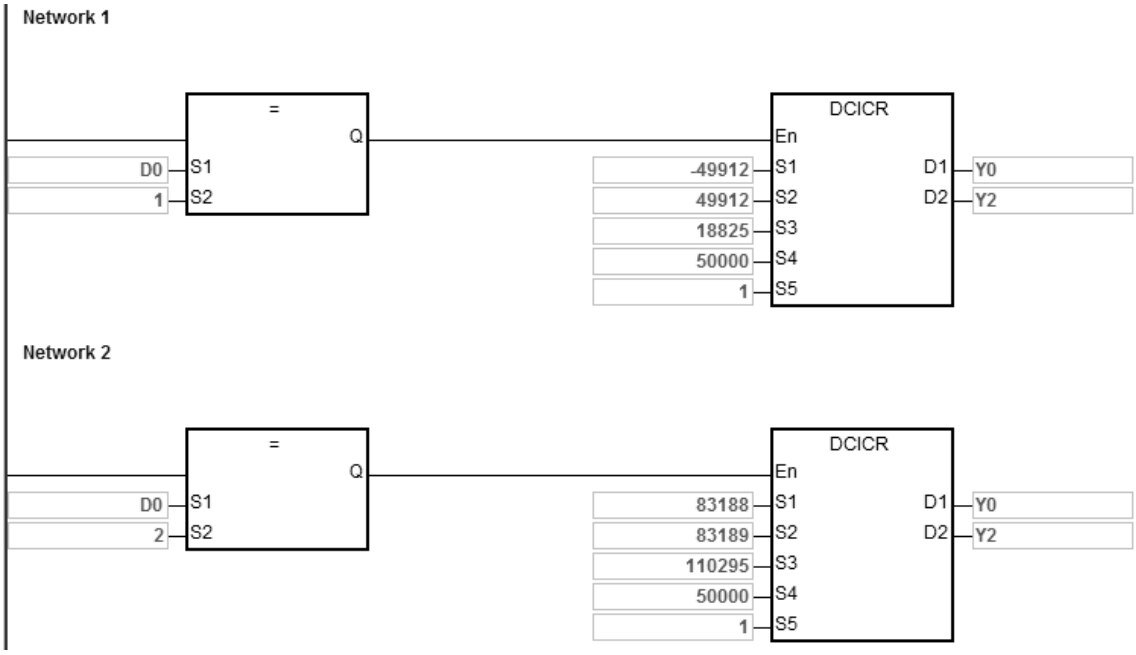
Example

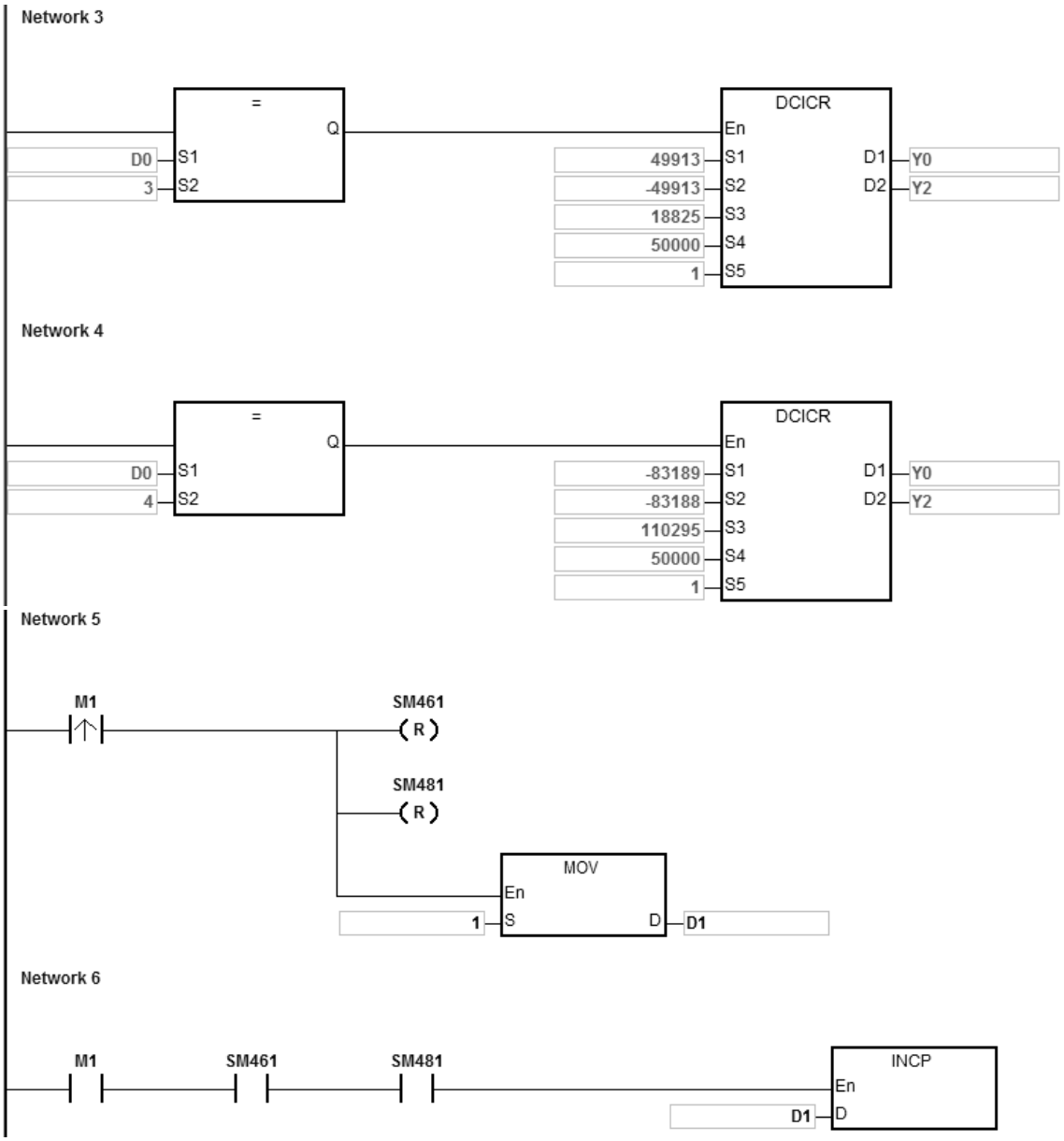
1. Draw a DELTA LOGO as the figure below.



2. Steps: divide the logo into three parts.

- For an ellipse: use the DCICR instruction for the relative-position clockwise arc interpolation.
- The absolute coordinates of the ellipse: P1（39930，-9982）, P2（-9982，39930）, P3（73206，123119） and P4（123119，73206）.
- Taking （39930，-9982）as the starting point, yields these relative coordinates: （-49912，49912），（83188，83189），（49913，-49913） and （-83189，-83188）.





When the PLC runs the program and M1 is ON, the PLC starts drawing the first segment of the arc. D1 increments by one when a segment of arc is completed and the second segment of the arc starts to automatically execute. This operation pattern repeats until the fourth segment of arc is completed.

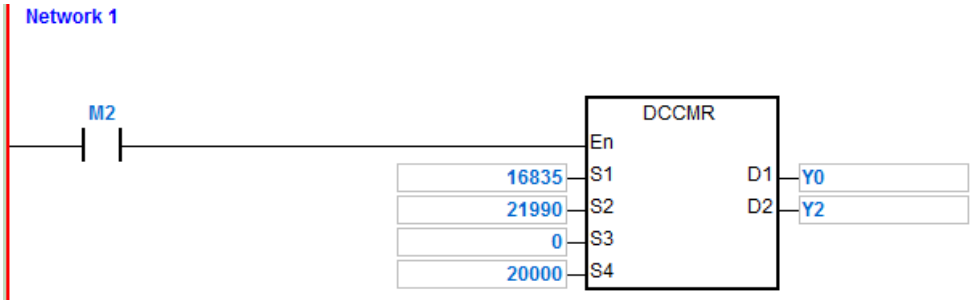
When D1=1, using DCICR, the PLC draws the first-segment arc from P1 to P2 with the shift of the center of the circle: 18825 and arc resolution: 5.

When D1=2, using DCICR, the PLC draws the second-segment arc from P2 to P3 with the shift of the center of the circle: 110295 and arc resolution: 5.

When D1=3, using DCICR, the PLC draws the third-segment arc from P3 to P4 with the shift of the center of the circle: 18825 and arc resolution: 5.

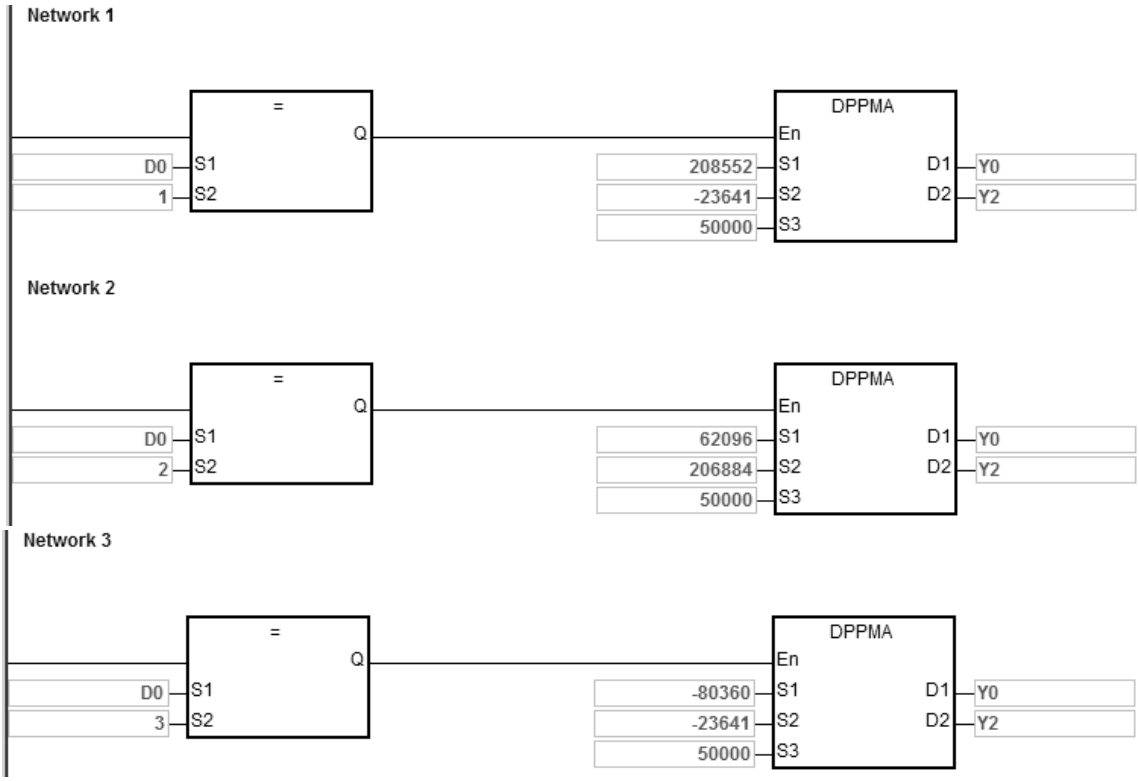
When D1=4, using DCICR, PLC draws the fourth-segment arc from P4 to P1 with the shift of the center of the circle: 110295 and arc resolution: 5.

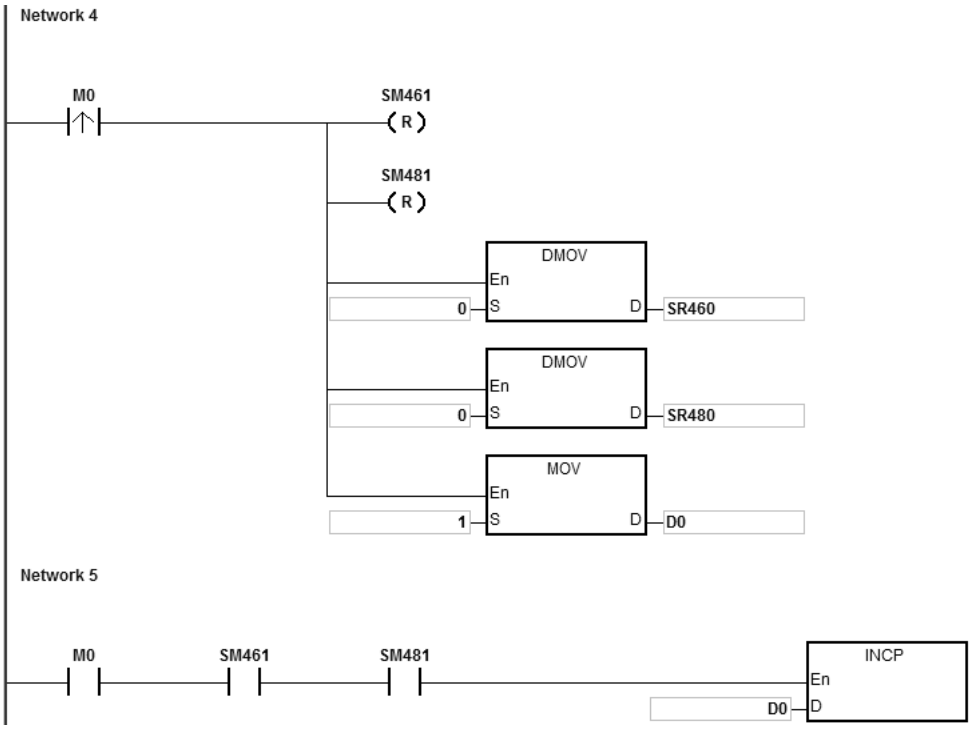
- For a circle = use DCCMR instruction (API 2716) for the relative-position circle drawing.
- The absolute coordinates of the circle: P5（86213，15435） and P6（103048，37425）. Taking （86213，15435） as the starting point, yields the relative coordinates for the center of the circle as （16835，21990）.



When the PLC runs the program and M2 is ON, the PLC starts to draw the relative-position circle with the target reference frequency of 20 kHz.

- For a delta, use DPPMA (API 2711) for the absolute-position 2-axis synchronized motion.
- Absolute coordinates of the delta= starting point P7 (-80360，23641), P8 (208552，-23641) and P9 (62096，206884).





When the PLC runs the program and M0 is ON, the PLC starts drawing the first segment of the 2-axis synchronized motion at 50 kHz. D1 increments by one when a segment of the 2-axis synchronized motion completes and the second segment of 2-axis synchronized motion starts to automatically execute. The operation pattern repeats until the third segment of 2-axis synchronized motion completes.

When D1=1, the PLC starts to draw the line from P7 to P8 with the DPPMA instruction.

When D1=2, the PLC starts to draw the line from P8 to P9 with the DPPMA instruction.

When D1=3, the PLC starts to draw the line from P9 to P7 with the DPPMA instruction.

API	Instruction			Operand	Description
2713	D	CICA		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis absolute-position clockwise arc interpolation

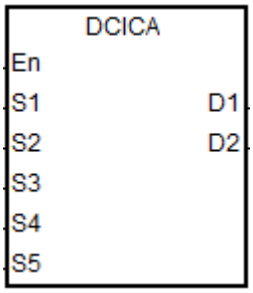
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		○
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●		●				
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



S_1 : X axis target coordinate (absolute positioning)

S_2 : Y axis target coordinate (absolute positioning)

S_3 : The shift of the center or the central angle

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. This instruction performs 2-axis absolute-position clockwise arc interpolation. **S_1** and **S_2** respectively designate the X- and Y-axis target coordinates. For more explanation, refer to DCICR instruction (API 2712).

2. Refer to Example 2 from the DPPMR instruction (API 2710) for programming in ST language.

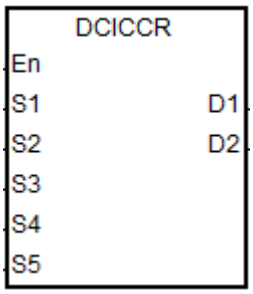
API	Instruction			Operand	Description
2714	D	CICCR		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis relative-position counterclockwise arc interpolation

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		○
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●		●				
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : X axis target coordinate (relative positioning)

S_2 : Y axis target coordinate (relative positioning)

S_3 : The shift of the center or the central angle

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. This instruction performs 2-axis relative-position clockwise arc interpolation. **S_1**, **S_2** and **S_3** respectively designate the target coordinate on X axis, target coordinate on Y axis, and the shift of the center of a circle. Refer to the following operation for drawing the counterclockwise arc for details.

2. **D_1** and **D_2** are the output devices for the X and Y axes respectively. Refer to the DPPMR instruction (API 2710) for the selection of output points and output mode for the axes.

3. **S_4** sets the target frequency for the reference. The target reference frequency is used for the prior calculation when the PLC plans the travel path for the arc, after the instruction is executed. If the estimated calculation process cannot achieve the planned arc travel path, the output frequency is automatically decreased so as to fulfill the synchronized arc drawing function.

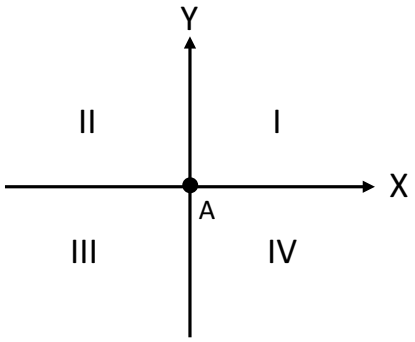
4. **S_5** is the setting value for function selection. See the descriptions below.

Setting value in **S_5**	**S_3** : The shift of the center or central angle	Description
0	Shift of the center (32-bit integer type)	The arc resolution uses a 10° arc as the basic angle for motion.
1	Shift of the center (32-bit integer type)	The arc resolution uses a 5° arc as the basic angle for motion.
2	Shift of the center (32-bit integer type)	The arc resolution uses a 1° arc as the basic angle for motion.
10	Shift of central angle (floating point type)	The arc resolution uses a 10° arc as the basic angle for motion.
11	Shift of central angle (floating point type)	The arc resolution uses a 5° arc as the basic angle for motion.
12	Shift of central angle (floating point type)	The arc resolution uses a 1° arc as the basic angle for motion.
Other	Shift of the center (32-bit integer type)	Seen as setting value in **S_5** is 0 The arc resolution uses a 10° arc as the basic angle for motion.

5. Refer to Example 2 from the DPPMR instruction (API 2710) for programming in ST language.

Drawing the counterclockwise arc

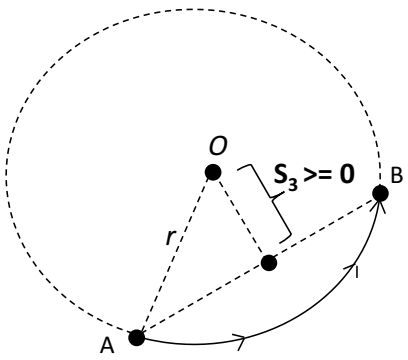
1. Define the four quadrants (QI, QII, QIII and QIV) of a coordinate system as below. Point A indicates the current coordinates on X and Y axes.



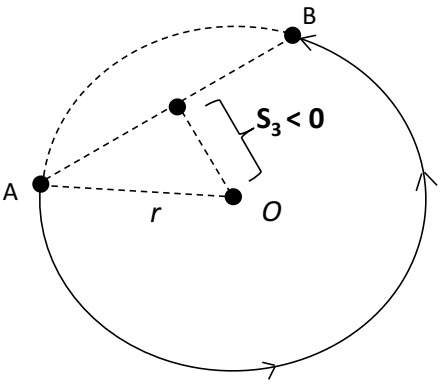
2. Point B is the target coordinates specified by **S1** and **S2**. Point O is the center of a circle containing points A and B.

3. **S3** is the distance to shift the center point O. **S3** can also be the distance to shift the center point or central angle, if it is defined as the central angle, it is the included angle of AOB.

4. The point A represents the origin of the axes X and Y in a plane. The target point B can be in Quadrant I, II, III, IV of the plane. Use the target point B to connect to the point A to draw a line and then use the setting values (shift the center point or the central angle) in **S3** to define an arc travel path.
   The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant I and IV with different setting values in **S3** (shift the center point). (r: radius)

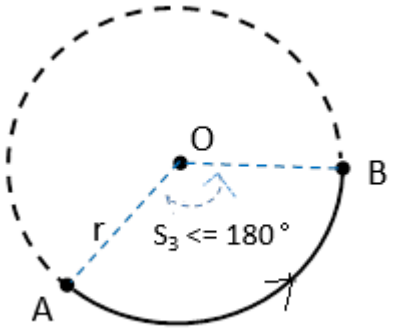


S3>= 0

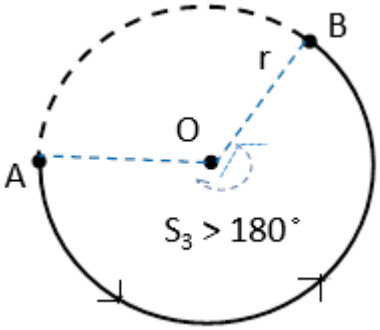


S3< 0

5. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant I and IV with different setting values in **S3** (shift the central angle). (r: radius)

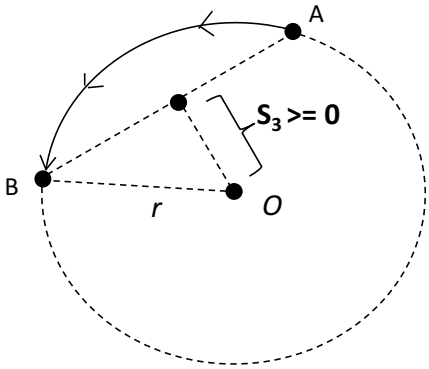


S3<= 180.0

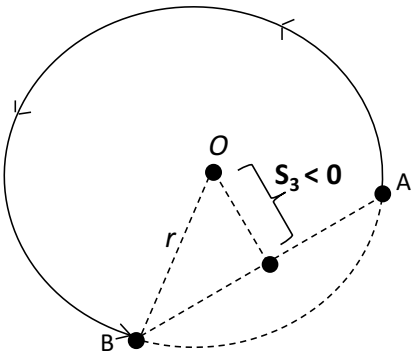


S3> 180.0

6. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant II and III with different setting values in **S3** (shift the center point). (r: radius)

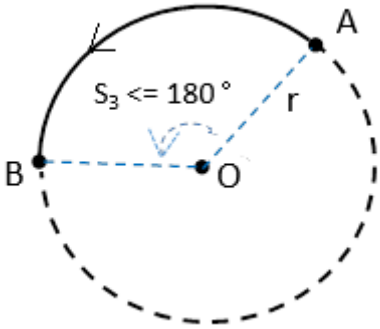


S3>= 0

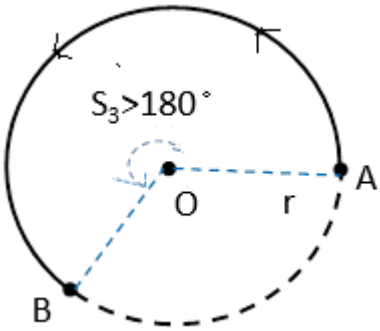


S3< 0

7. The figures below show 2 arc-travel path examples (solid lines) when the point B is in Quadrant II and III with different setting values in **S3** (shift the central angle). (r: radius)



S3<= 180.0

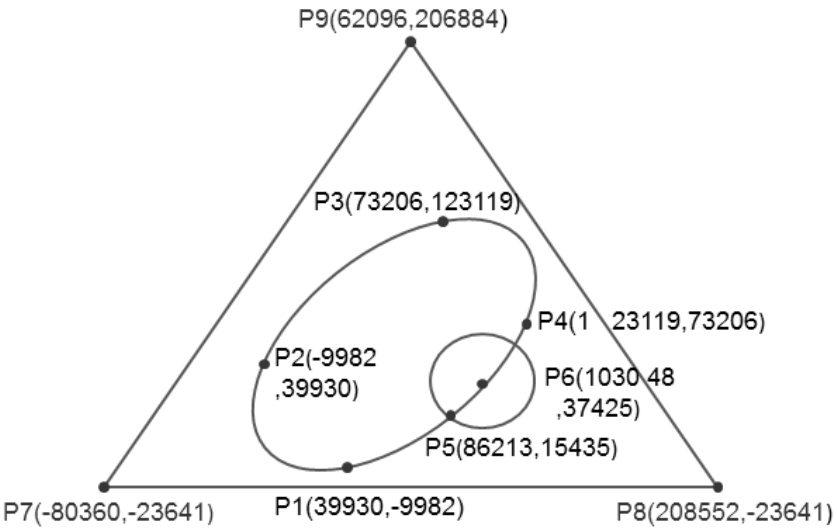


S3> 180.0

8. When the X axis coordinate = 0 and the Y axis coordinate >= 0 for target point B, then point B is defined as the point staying in QI of point A. Whereas, if Y axis coordinate < 0 for target point B, then point B is defined as the point staying in QIII of point A.

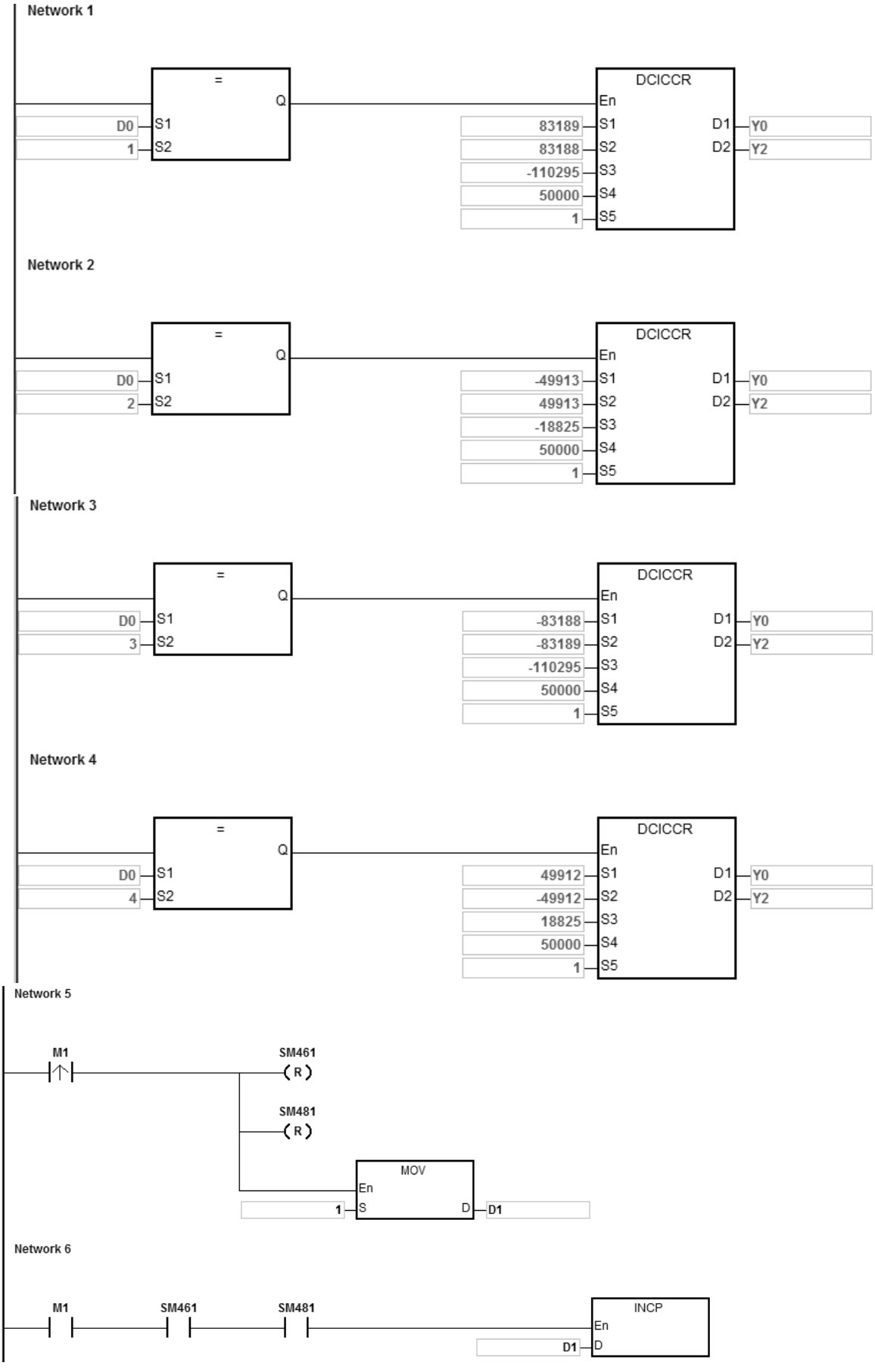
Example

1. Draw a DELTA LOGO as the figure below.



2. Steps: divide the logo into three parts.

- For an ellipse: use the DCICR instruction (API 2712) for the relative-position counterclockwise arc interpolation.
- The absolute coordinates of the ellipse: P1(39930，-9982), P4 (123119，73206), P3 (73206，123119) and P2 (-9982，39930)
- Taking (39930，-9982) as the start point, the obtained relative coordinates: (83189, 83188), (-49913, 49913), (-83188，-83189) and (49912，-49912).


Network 1
=
Q
D0
S1
1
S2
DCICCR
En
83189
S1
D1
Y0
83188
S2
D2
Y2
-110295
S3
50000
S4
1
S5
Network 2
=
Q
D0
S1
2
S2
DCICCR
En
-49913
S1
D1
Y0
49913
S2
D2
Y2
-18825
S3
50000
S4
1
S5
Network 3
=
Q
D0
S1
3
S2
DCICCR
En
-83188
S1
D1
Y0
-83189
S2
D2
Y2
-110295
S3
50000
S4
1
S5
Network 4
=
Q
D0
S1
4
S2
DCICCR
En
49912
S1
D1
Y0
-49912
S2
D2
Y2
18825
S3
50000
S4
1
S5
Network 5
M1
SM461
R
SM481
R
MOV
En
1
S
D
D1
Network 6
M1
SM461
SM481
INCP
En
D1
D

When the PLC runs the program and M1 is ON, the PLC starts drawing the first segment of the clockwise arc at 50kHz. D1 increments by 1 when a segment of arc is completed and the second segment of arc starts to automatically execute. The operation pattern repeats until the fourth segment of arc is completed.

When D1=1, using DCICCR, the PLC draws the first-segment arc from P1 to P4 with the shift of the center of the circle: -110295 and arc resolution: 5.

When D1=2, using DCICCR, the PLC draws the second-segment arc from P4 to P3 with the shift of the center of the circle: -18825 and arc resolution: 5.

When D1=3, using DCICCR, the PLC draws the third-segment arc from P3 to P2 with the shift of the center of the circle: -110295 and arc resolution: 5.

When D1=4, using DCICCR, the PLC draws the fourth-segment arc from P2 to P1 with the shift of the center of the circle: -18825 and arc resolution: 5.

- A circle = use DCCMR instruction (API 2716) for the relative-position circle drawing. Refer to the example in API 2712 DCICR for detailed operation.
- A delta = use DPPMA (APIU 2711) for the absolute-position 2-axis synchronized motion. Refer to the example in DCICR (API 2712) for more information.

API	Instruction			Operand	Description
2715	D	CICCA		S_1, S_2, S_3, S_4, S_5, D_1, D_2	2-Axis absolute-position counterclockwise arc interpolation

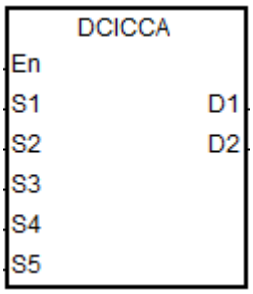
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		○
S_4							●	●	●		○		○	○		
S_5							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●		●				
S_4			●				●						
S_5			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



S_1 : X axis target coordinate (absolute positioning)

S_2 : Y axis target coordinate (absolute positioning)

S_3 : The shift of the center or the central angle

S_4 : Target reference frequency

S_5 : Function

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. S_1 and S_2 respectively designate the target coordinate on the X axis and the target coordinate on the Y axis. For an explanation of the instruction, refer to DCICCR instruction (API 2714).

2. Refer to Example 2 from the DPPMR instruction (API 2710) for programming in ST language.

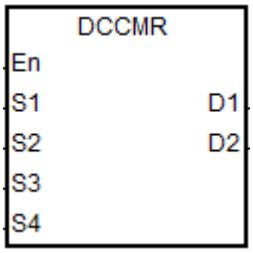
API	Instruction			Operand	Description
2716	D	CCMR		**S_1, S_2, S_3, S_4, D_1, D_2**	Relative-position circle drawing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : X axis coordinate of the center (relative positioning)

S_2 : Y axis coordinate of the center (elative positioning)

S_3 : Function selection

S_4 : Target reference frequency

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

Explanation

1. This instruction draws a circle based on relative positioning. **S_1** and **S_2** are respectively the X and Y axis coordinates of the center of a circle (relative position).

2. **S_3** is the setting value for function selection. See the descriptions below.

S_3 : Function Selection	Descriptions
0	The arc resolution uses a 10° arc as the basic angle for clockwise motion.
1	The arc resolution uses a 5° arc as the basic angle for clockwise motion.
2	The arc resolution uses a 10° arc as the basic angle for counterclockwise motion.
3	The arc resolution uses a 5° arc as the basic angle for counterclockwise motion.
4	The arc resolution uses a 1° arc as the basic angle for clockwise motion.
5	The arc resolution uses a 1° arc as the basic angle for counterclockwise motion.
Other	Seen as setting value in **S_5** is 0 The arc resolution uses a 1° arc as the basic angle for clockwise motion.

3. **D_1** and **D_2** are the pulse output devices for the X and Y axes respectively. Refer to DPPMR instruction (API 2710) for the selection of output points and output mode for the axes.

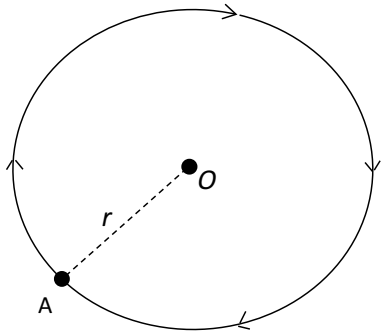
4. **S_4** sets the target frequency for the reference. The target reference frequency is used for the prior calculation when the PLC plans the travel path for the arc after the instruction is executed. But if the calculation process which is estimated cannot achieve the planned arc travel path, the output frequency is automatically decreased to fulfill the synchronized arc drawing function.

5. When the X axis coordinate of the center point O=0, point O is defined as the point staying in QI of point A if Y axis target coordinate >= 0. Whereas, if Y axis target coordinate < 0, point B is defined as the point staying in QIII of point A.

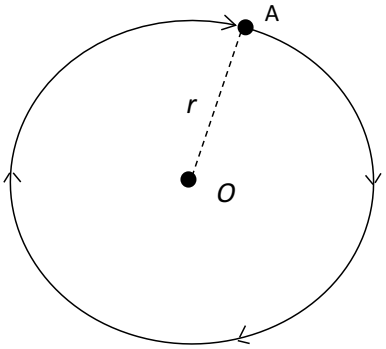
6. Refer to Example 2 from the DPPMR instruction (API 2710) for programming in ST language.

The clockwise circle drawing:

Point A is the present position and point O is the target center, r is the radius of the circle as the figures below.

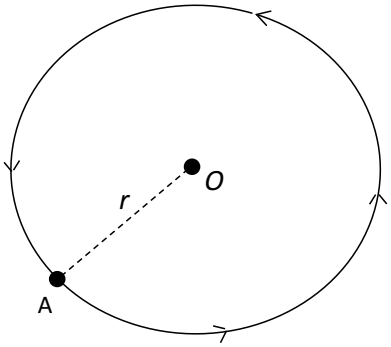


Center O in QI and QIV

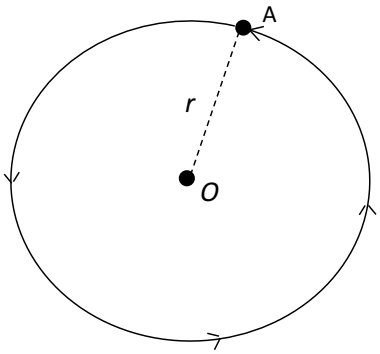


Center O in QII and QIII

The counterclockwise circle drawing:



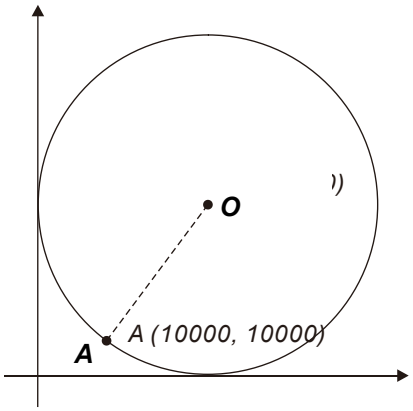
Center O in QI and QIV



Center O in QII and QIII

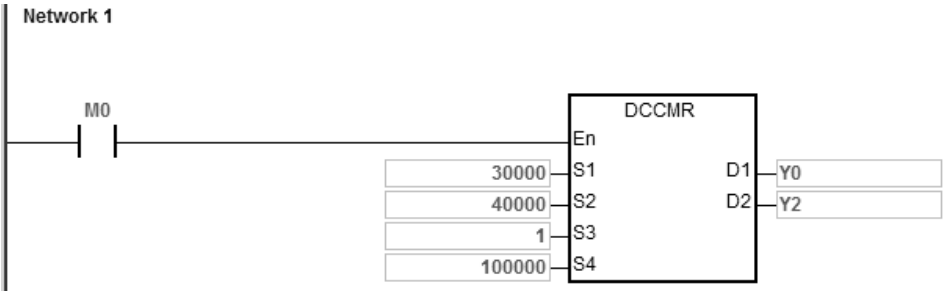
Example

1. Taking (40000, 50000) as the center O and (10000, 10000) as point A, draw a circle as shown below.



2. Steps:

- The calculated relative coordinates of point O: (30000, 40000)
- The coordinates of point A is the present position; SR460=10000 and SR480=10000
- **S_3**=1 and the clockwise circle drawing is performed with a 5° as the basic motion angle.
- When the PLC runs the program and M0 is set to ON, the relative-position circle drawing starts.



3. Operation:

- When the PLC runs the program and M0 is ON, the PLC starts to draw the circle in the clockwise direction at 100 kHz.
- When the circle drawing completes, SM461 and SM481 are ON.

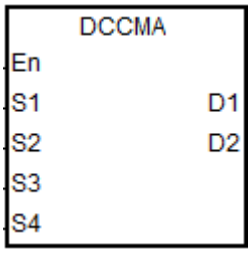
API	Instruction			Operand	Description
2717	D	CCMA		**S_1, S_2, S_3, S_4, D_1, D_2**	The absolute-position circle drawing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1							●	●	●		○		○	○		
S_2							●	●	●		○		○	○		
S_3							●	●	●		○		○	○		
S_4							●	●	●		○		○	○		
D_1		○														
D_2		○														

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : X axis coordinate of the center (absolute positioning)

S_2 : Y axis coordinate of the center (absolute positioning)

S_3 : Function selection

S_4 : Target frequency

D_1 : Pulse output device for the X axis

D_2 : Pulse output device for the Y axis

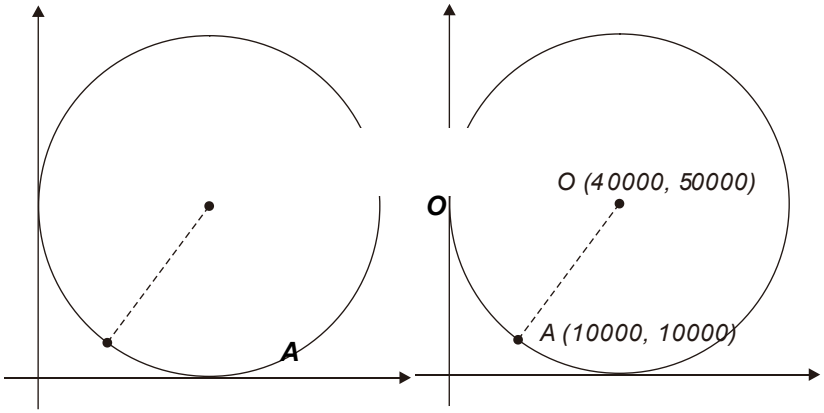
Explanation

1. **S_1** and **S_2** are respectively the X and Y axis coordinates of the center of a circle (absolute positioning). For more information, refer to the DCCMR instruction (API 2716).

2. Refer to Example 2 from the DPPMR instruction (API 2710) for programming in ST language.

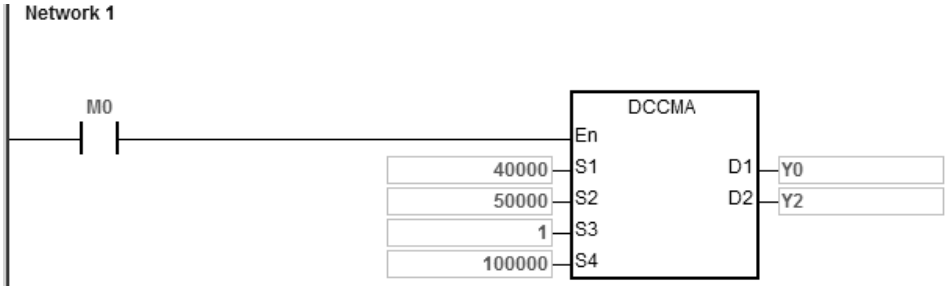
Example

1. Taking (40000, 50000) as the center O and (10000, 10000) as point A, draw a circle as shown below.



3. Steps:

- Set the absolute X coordinate and Y coordinate (40000, 50000) in the instruction
- Point A is the present position, SR460=10000 and SR480=10000
- **S_3** is 1 and the clockwise circle is drawn with 5° as the basic motion angle.
- When the PLC runs the program and M0 is ON, the absolute-position circle drawing starts.



4. Operation:

- When the PLC runs the program and M0 is ON, the PLC starts to draw the circle in the clockwise direction at 100 kHz.
- When the circle drawing completes, SM461 and SM481 are ON.

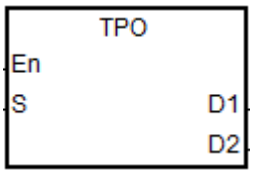
API	Instruction	Operand	Description
2718	TPO	**S**，**D_1**，**D_2**	Position planning table controls the output

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S								●				○	○	○		
D_1								●								
D_2		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●			●	●							
D_1		●			●	●							
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	-

Symbol



S ： The first output number in the position planning table

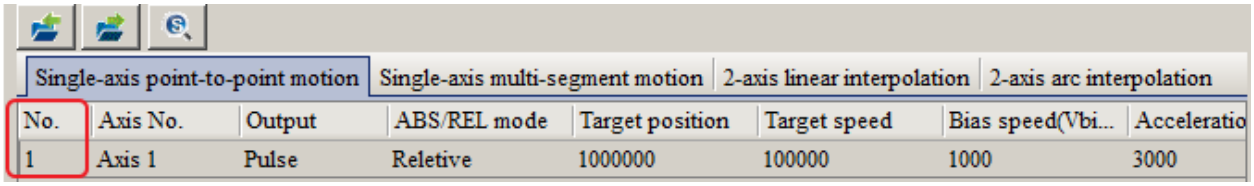
D_1 ： The output number that is being output

D_2 ： Switch flag at the end of the consecutive-number output

6

Explanation:

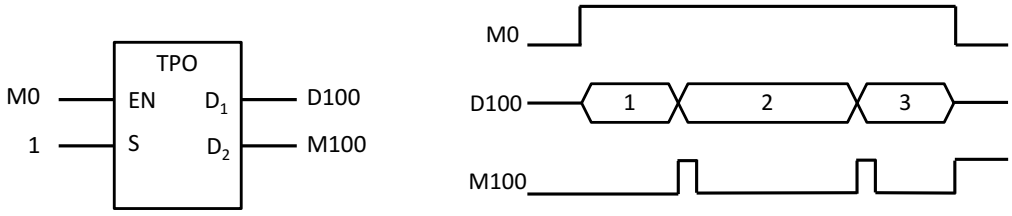
1. This instruction uses the position planning table to control the output. **S** is the number listed in the position planning table. If the number does not exist in the table, the instruction is not executed, SM0 is ON and the error code is SR0=16#2027.



2. **S** is the first output number when PLC runs. Refer to the value in **D_1** if there are consecutive-number outputs during the output.

3. When one single number or the first one of consecutive numbers is output, the switch flag **D_2** is set to OFF. When the consecutive-number output reaches the output number to be switched to, **D_2** is ON for a scan cycle and then is ON again until the last number output is completed.

The example and timing diagram for the flag state switching are shown below.

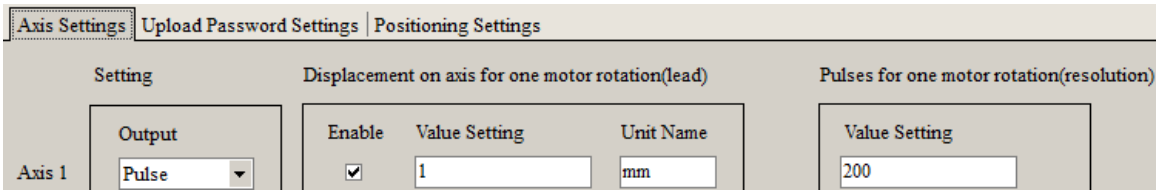


4. The switch flag **D_2** is affected by the PLC scan time. If the time for switching to the next output number is shorter than the scan time, the flag for switching to the output number may not reset to OFF.

5. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Example 1:

1. Enable the mechanical unit conversion function

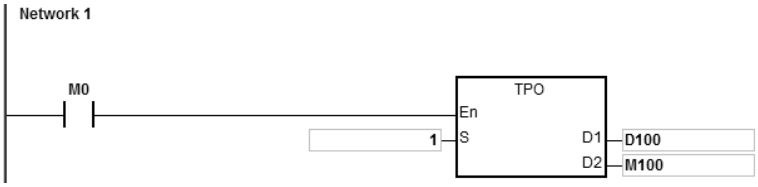


2. Set the position planning table and download the parameters

   The target position is 2000 (Unit:mm). After converting in the position planing table, 400,000 output pulses are downloaded and saved in the PLC.

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	Bias speed(Vbi...	Acceleration ti...
1	Axis 1	Relative	2000	100000	200	200

3. When M0 is ON, Y0 outputs 400,000 pulses at 100kHz. After the output completes, M100 is ON.



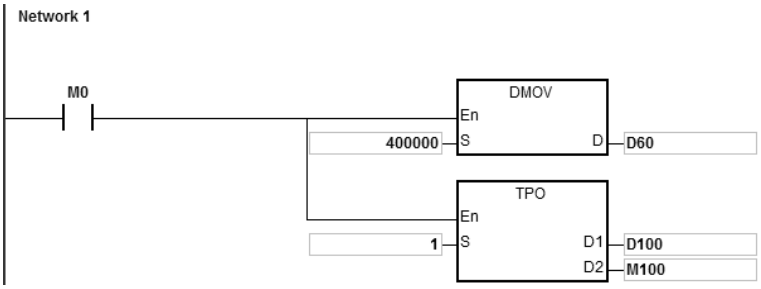
Example 2:

Use global symbol or D device in Target Position

1. Use D device in Target Position of the position planning table and download the parameters.
   The unit for D60 in PLC is pulse by default. If you need to use D devices in Target position and in the mechanical unit, you need to convert the mechanical unit to pulse.

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	Bias speed(Vbi...	Acceleration ti...
1	Axis 1	Relative(Symbol/Address)	D60	100000	200	200

2. When M0 is ON, and D60 is 400,000. Y0 outputs 400,000 pulses at 100kHz. After the output completes, M100 is ON.



Note: the unit is pulse, when you use global symbols, D devices, or instructions including DTPWS, DTPWL, and DTPWC to input values in in Target Postion.

3. If global symbols are used in Target Position

   A. You need to assign an address for the global symbol.

Global Symbols

	Global Symbols				
	Class	Identifiers	Address	Type...	Initial Value (Activ...
▶	VAR	Target_Axis1	D60	DWORD	N/A

   B. Use global symbols in Target Position of the planning table.

Single-axis point-to-point motion | Single-axis multi-segment motion | 2-axis linear interpolation | 2-axis arc interpolation

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	Bias speed(Vbi...	Acceleration ti...
1	Axis 1	Relative(Symbol/Address)	Target_axis1	100000	200	200

Example 3:

Use abort symbol and completion flag

1. Set up the positioning planning table as shown below.

Single-axis point-to-point motion | Single-axis multi-segment motion | 2-axis linear interpolation | 2-axis arc interpolation

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	ACC/DEC time(ms)	Abort signal(I/M)	Completion action	Completion flag(I/M)	No. of next segment
3	Axis 1	Relative	10000	5000	300	M100	Go to next segment		4
4	Axis 1	Relative	15000	8000	100		Completion	M777	---

2. Abort symbol: When No. 3 stops the outputting abruptly, M100 is ON and the system starts to output No. 4 immediately.

   The abort symbol is applicable for the following conditionss: external interrupts, high-speed comparision interrupts and M devices (M0~8191).

3. Completion flag: When No. 4 completes the outputting, M777 is ON. When the execution of this positioning planning table is complete, the system calls for an interrupt or starts a M device.

   You can use the interupt numbers I510~519 and M device (M0~M8191) in the positioning planning table.

Note: If you use interrupts instead of M devices, PLC scan cycle is not affected.

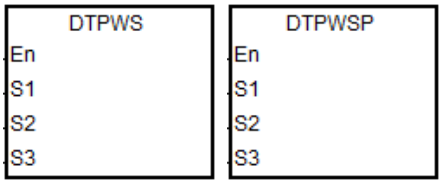
API	Instruction			Operand	Description
2719	D	TPWS	P	S_1, S_2, S_3	Setting single-axis output parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



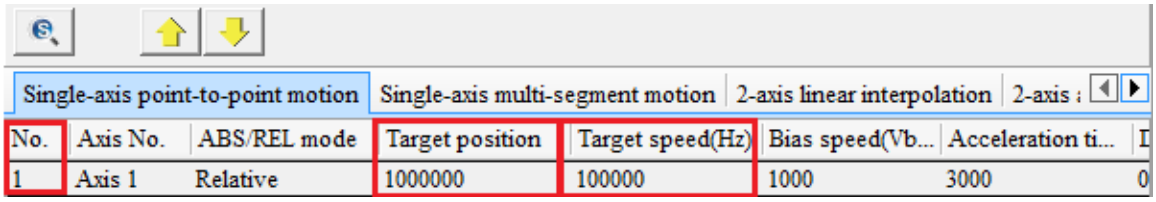
S_1 : A number listed in the position planning table

S_2 : Target position

S_3 : Target speed

Explanation

1. This instruction sets the single-axis output parameters in the position planning table. S_1 is the number listed in the position planning table. If the number does not exist in the table, or the number is not under Single-axis point-to-point motion, or under Single-axis multi-segment motion, the instruction is not executed, SM0 is ON and the error code is SR0=16#2027.



2. S_2 is the target position (must be a 32-bit integer). If you use the mechanical unit conversion when editing the position planning table in ISPSoft, use the conversion instruction for modification first.

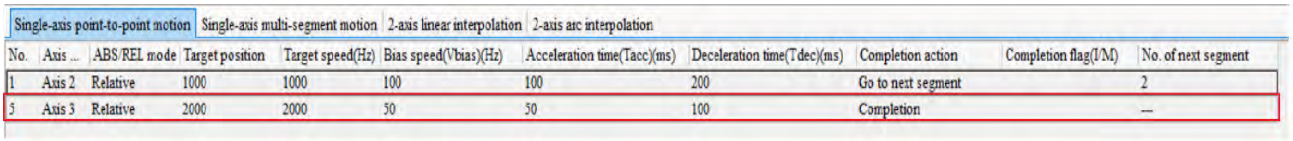
3. S_3 is the target speed. The range is between 1–200,000Hz.

4. When the instruction is executed to modify the parameters for the single axis which is outputting pulses, the modified parameters are kept in the table and are not effective until the next output starts.

5. The parameters modified by the instruction can only be modified while the PLC is running. The last written parameter is not saved when the power turns OFF. The table that you edit in ISPSoft and download to the PLC is processed as the default position planning table when the power is ON.

6. This instruction does NOT modify the acceleration or deceleration time. If you need to modify the time for acceleration or deceleration, you can set SM585 flag to ON, and the system refers to the acceleration and deceleration time set in SR to change the acceleration time in single-axis point-to-point motion and single-axis multi-segment motion or the deceleration time in single-axis point-to-point motion. Re-execute this instruction to actualize this modification.

Example 1: Change the parameters of Axis 3 of Number 5 during execution in Single-axis point-to-point motion.



Single-axis point-to-point motion | Single-axis multi-segment motion | 2-axis linear interpolation | 2-axis arc interpolation

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	Bias speed(Vbias)(Hz)	Acceleration time(Tacc)(ms)	Deceleration time(Tdec)(ms)	Completion action	Completion flag(I/M)	No. of next segment
1	Axis 2	Relative	1000	1000	100	100	200	Go to next segment		2
5	Axis 3	Relative	2000	2000	50	50	100	Completion		---

1) Use DMOV instruction to define new target position and target speed.

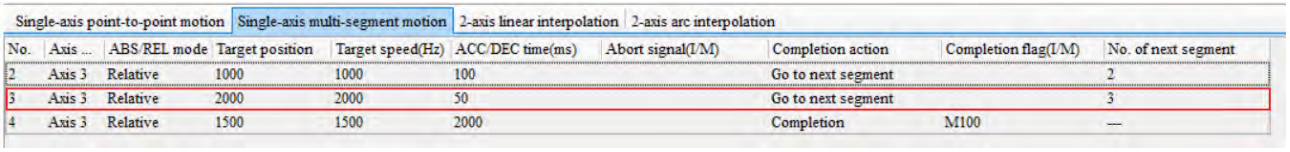
2) Use MOV instruction to define the start/end frequency in SR503, acceleration time in SR504 and deceleration time in SR505 for Axis 3. After that the relative parameters in single-axis point-to-point motion are updated.

3) Set the flag SM585 to ON.

4) Re-execute DTPWS instruciton to actualize the modification.

5) After that, SM585 switches to OFF automatically.

Example 2: Change the parameters of Axis 3 of Number 2 during execution in Single-axis multi-segment motion.



Single-axis point-to-point motion | Single-axis multi-segment motion | 2-axis linear interpolation | 2-axis arc interpolation

No.	Axis ...	ABS/REL mode	Target position	Target speed(Hz)	ACC/DEC time(ms)	Abort signal(I/M)	Completion action	Completion flag(I/M)	No. of next segment
2	Axis 3	Relative	1000	1000	100		Go to next segment		2
3	Axis 3	Relative	2000	2000	50		Go to next segment		3
4	Axis 3	Relative	1500	1500	2000		Completion	M100	---

1) Use DMOV instruction to define new target position and target speed.

2) Use MOV instruction to define the acceleration time in SR504. After that the relative parameters in single-axis multi-segment motion are updated.

3) Set the flag SM585 to ON.

4) Re-execute DTPWS instruciton to actualize the modification.

5) After that, SM585 switches to OFF automatically.

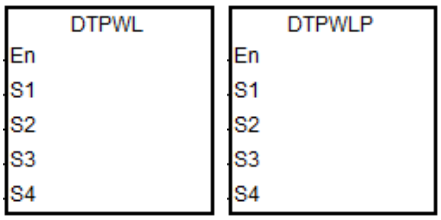
API	Instruction			Operand	Description
2720	D	TPWL	P	S_1, S_2, S_3, S_4	Setting linear interpolation parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : A number listed in the position planning table

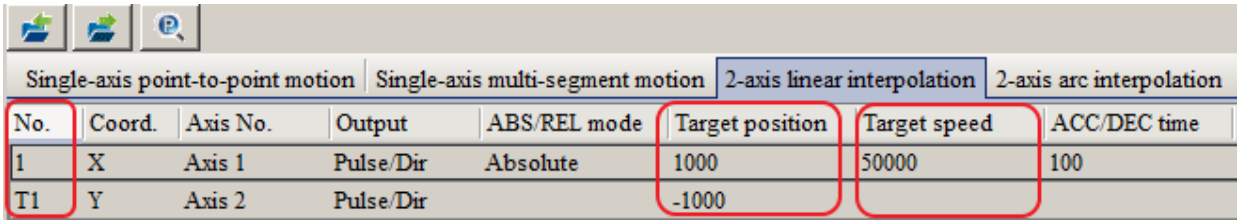
S_2 : Target position of the X axis

S_3 : Target position of the Y axis

S_4 : Target speed

Explanation

1. This instruction sets the 2-axis linear interpolation parameters in the position planning table. S_1 is the number listed in the position planning table. If the number does not exist in the table or the output of the number does not belong to the 2-axis linear interpolation, the instruction is not executed, SM0 is ON and the error code is SR0=16#2027.



2. S_2 and S_3 are respectively the target positions of the X and Y axes, which can only be 32-bit integers. If you use the mechanical unit conversion when editing the position planning table in ISPSoft, use the conversion instruction for modification first.

3. **S_4** is the target speed. The range is between 1–200,000Hz.

4. When the instruction executes the linear interpolation, the target frequency **S_4** automatically corresponds to the output of the axis which is farthest from its target position. If X axis and Y axis cannot simultaneous reach the target positions, the PLC automatically decelerates the frequency to make the two axes reach the target positions simultaneously.

5. When the instruction is executed to modify parameters for the two axes (either of which is outputting), the modified parameters of the two axes are kept in the table and are not effective until the next 2-axis output starts.

6. The parameters modified by the instruction can be modified only while the PLC is running. The last written parameter is not saved when the power turns OFF. The table that you edit in ISPSoft and download to the PLC is processed as the default position planning table when the power is ON.

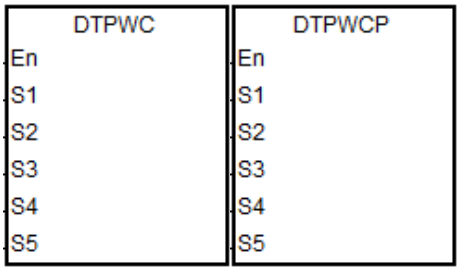
API	Instruction			Operand	Description
2721	D	TPWC	P	S_1, S_2, S_3, S_4, S_5	Setting arc interpolation parameters in the position planning table

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●	●				○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4			●				●						
S_5			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
ES3	-	ES3

Symbol



S_1 : A number listed in the position planning table

S_2 : X axis target position

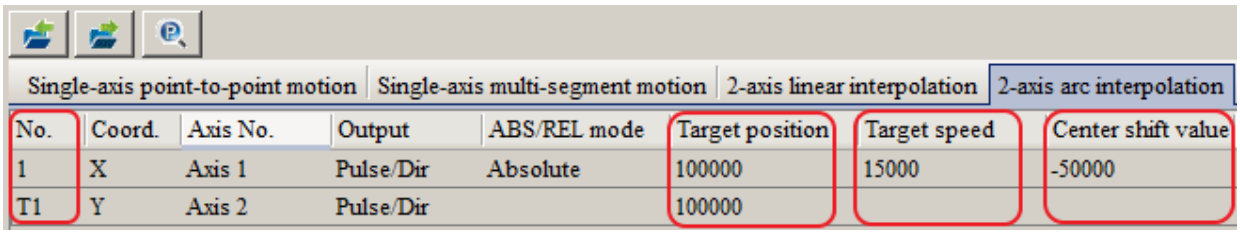
S_3 : Y axis target position

S_4 : The shift of the center

S_5 : Target speed

Explanation

1. This instruction sets the 2-axis arc interpolation parameters in the position planning table. S_1 is the number listed in the position planning table. If the number does not exist in the table, or the output of the number does not belong to the 2-axis arc interpolation, the instruction is not executed, SM0 is ON and the error code is SR0=16#2027.



2. **S_2** and **S_3** are respectively the target positon on the X and Y axes. **S_4** is the distance to shift the center. The three parameters can only be 32-bit integers. If you use the mechanical unit conversion when editing the position planning table in ISPSoft, use the conversion instruction for modification first.

3. **S_5** is the target speed. The range is between 1–200,000 Hz. It is the reference speed for the actual output. The PLC automatically decelerates the speed if the 2-axis synchronized output cannot be achieved.

4. When the instruction is executed to modify parameters for the two axes (either of which is outputting), the modified parameters of the two axes are kept in the table and are not effective until the next 2-axis output starts.

5. The parameters modified by the instruction can be modified only while the PLC is running. The last written parameter is not saved when the power turns OFF. The table that you edit in ISPSoft and downloaded to the PLC is processed as the default position planning table when the power is ON.

This instruction does not support changing the clockwise or counterclockwise motion direction. For direction change, add or modify the output parameters for arc interpolation in the position planning table in ISPSoft.

API	Instruction			Operand	Description
2723	D	PPGB		**S_1, S_2, S_3, S_4, D_1, D_2**	Point-to-point go back and forth

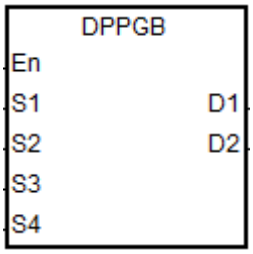
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●								
S_2								●								
S_3								●								
S_4								●								
D_1		○														
D_2			●													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						
S_4									●				
D_1	●												
D_2	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	-	ES3

6

Symbol



S_1 : Relative target position A

S_2 : Relative target position B

S_3 : Target speed

S_4 : Target speed ratio adjusted value (floating point value)

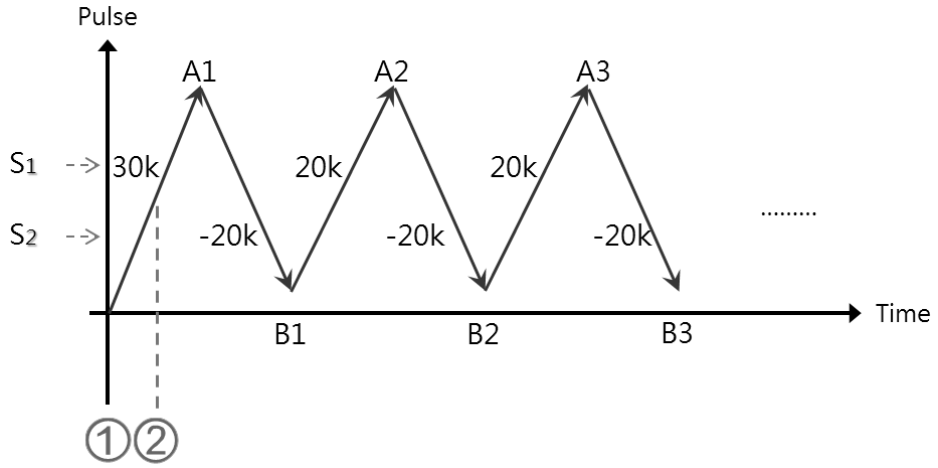
D_1 : Pulse output device

D_2 : Target speed can change indicator

Explanation

1. This high speed output instruction specifies a movement going back and forth between two target positions that are converted from the two relative target positions specified by **S_1** and **S_2**. This instruction is applicable for machines such as warping machines in the textile industry, and winding & binding machines in the cable industry.

2. When the instruction starts the output, the relative positions **S_1** and **S_2** must be specified first so that the instruction can make the calculation for switching to the next output. After the instruction is executed, you can modify the target positions to execute, but you cannot change the outputting target positions.



① DPPGB instruction is executed. When **S₁** is 30,000 and **S₂** is -30,000, the motion goes toward the target position A1=30,000.

② If the pulse position>15,000, the relative positions are modified into **S₁**:20,000 and **S₂**:-20,000, and the target position is momentarily A1=30,000. The target positions obtained through calculation are B1=10,000(A1+ **S₂**) and A2=30,000(B1+ **S₁**), since the outputting target positions cannot be modified.

3. **S₃** is the target speed (must be a 32-bit integer), and **S₄** is the target speed ratio adjusted value (must be a floating point number). The actual speed is the result of multiplying the values in **S₃** and **S₄** and then rounding down the calculation result to a 32-bit integer. The acceptable input speed range is 1–200 kHz. When values are out of range, the PLC automatically processes the value as the minimum or the maximum of the operation range. The ratio formula for the actual target speed is **S₃** x **S₄**. For instance, if target speed is 1 kHz, the adjusted floating point is 1.2345 and the actual value is 1234 Hz.

4. When the instruction is outputting, the target speed and the adjusted ratio can be modified, and the result is updated to the actual output speed once the instruction is scanned. But it is suggested that you do not change the target speed too much in case the calculated deceleration is affected.

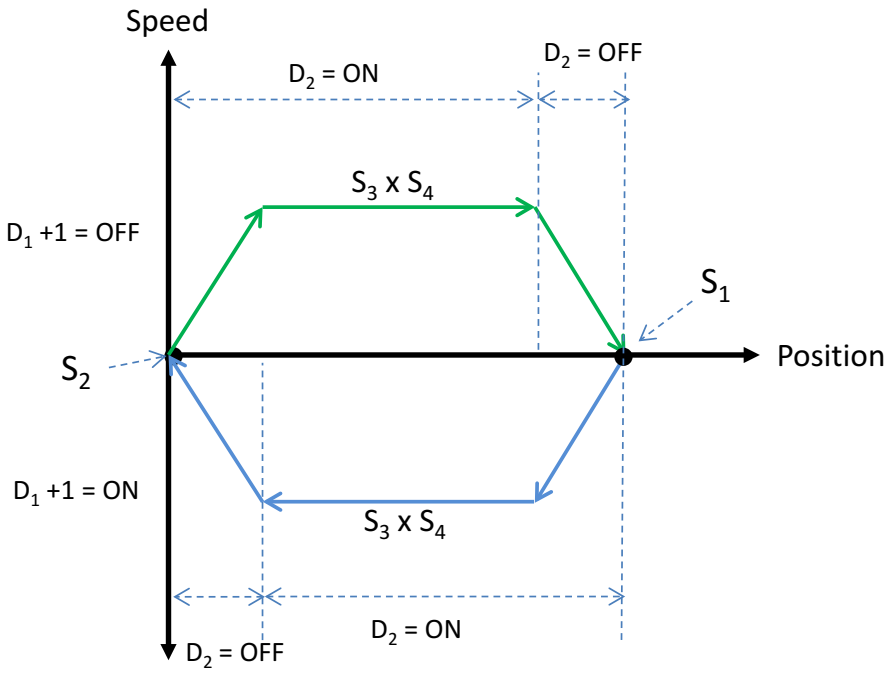
5. The output points for **D₁** can only be Y0, Y2, Y4, and Y6, and it occupies 2 consecutive output points. The output sets and the output special register modes are listed in the following table.

Output axis number	1	2	3	4
D₁+0 output point	Y0	Y2	Y4	Y6
D₁+1 output point	Y1	Y3	Y5	Y7
Output mode	SR462	SR482	SR502	SR522

6. **D_2** is the flag indicating the target speed and the adjusted ratio that you can change. When this flag is ON, the executing target speed can be modified. When the flag switches from ON to OFF, it is now decelerating and the current target speed will be processes as the target speed for the next output.

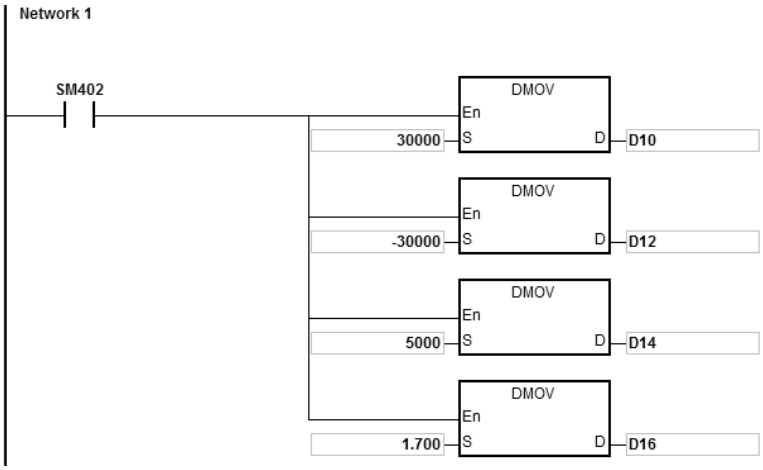
7. There is no limit on the number of times you can execute this instruction, but during execution, the designated high speed axis cannot be occupied by any other instruction, and the other instruction is not executed.

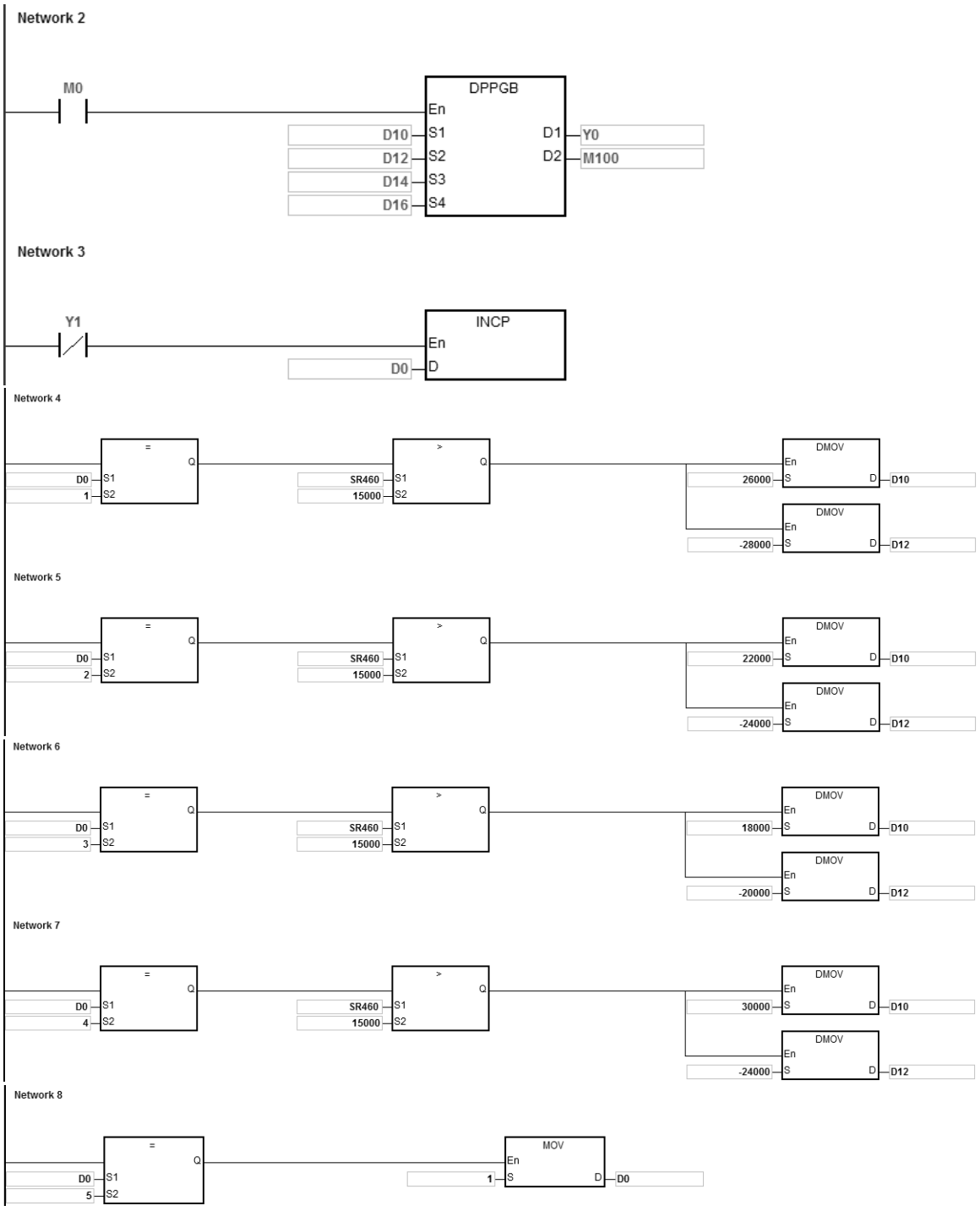
The following graphic shows the output timing diagram.



Example

When M0 is ON, the DPPGB instruction starts to move back and forth between the target positions that are calculated from the two relative target positions specified by **S_1** and **S_2**, at the actual target speed of 8500 Hz (5000*1.7). Y0 is the output point. Y1=OFF means the positive direction.

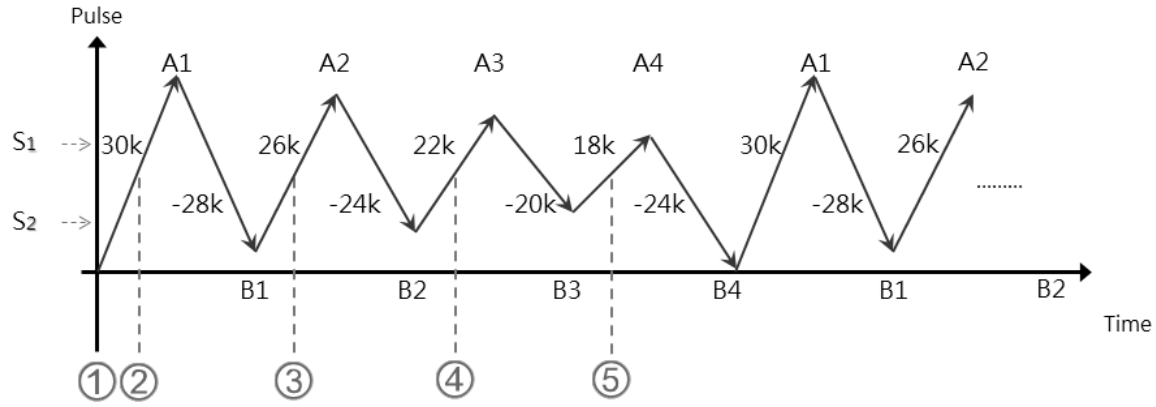



Network 2
M0
DPPGB
En
D10 S1
D12 S2
D14 S3
D16 S4
D1 Y0
D2 M100
Network 3
Y1
INCP
En
D0 D
Network 4
=
Q
D0 S1
1 S2
>
Q
SR460 S1
15000 S2
DMOV
En
26000 S
D D10
DMOV
En
-28000 S
D D12
Network 5
=
Q
D0 S1
2 S2
>
Q
SR460 S1
15000 S2
DMOV
En
22000 S
D D10
DMOV
En
-24000 S
D D12
Network 6
=
Q
D0 S1
3 S2
>
Q
SR460 S1
15000 S2
DMOV
En
18000 S
D D10
DMOV
En
-20000 S
D D12
Network 7
=
Q
D0 S1
4 S2
>
Q
SR460 S1
15000 S2
DMOV
En
30000 S
D D10
DMOV
En
-24000 S
D D12
Network 8
=
Q
D0 S1
5 S2
MOV
En
1 S
D D0

Explanation

1. After the instruction starts the output, modify the relative target positions A for **S_1** and B for **S_2**.

   a) The Y0 output graph:



   b) Explanation:

   ① When **S_1**:30,000, **S_2**:-30,000 and M0 enables the DPPGB instruction, the motion goes toward the position A1=30,000.

   ② When D0=1 and the position: number of pulses in SR460 >15,000, the relative positions are changed to **S_1**:26,000 and **S_2**:-28,000.

   At the moment, the target position is A1=30,000 since the outputting target position cannot be modified. The calculated target positions are B1=2,000(A1-28,000), A2=28,000(B1+26,000).

   ③ When D0=2 and the position: number of pulses in SR460 >15,000, the relative positions are changed to **S_1**:22,000 and **S_2**:-24,000.

   At this time, the target position is A2=28,000 since the outputting target position cannot be modified.

   The calculated target positions are B2=4,000(A2-24,000), A3=26,000(B2+22,000).

   ④ When D0=3 and the position: number of pulses in SR460 >15,000, the relative positions are changed to **S_1**:18,000 and **S_2**:-20,000.

   The calculated target positions are B3=6,000(A3-20,000)，A4=24,000(B3+18,000).

   ⑤ When D0=4 and the position: number of pulses in SR460 >15,000, the relative positions are changed to **S_1**:30,000 and **S_2**:-24,000.

   The calculated target positions are B4=0(A4-24,000) and A1=30,000(B4+30,000).

Step	S_1: Relative position	S_2: Relative position	Target position B	Target position A
①	30,000	-30,000	-	A1=30,000
②	26,000	-28,000	B1=2,000	A2=28,000
③	22,000	-24,000	B2=4,000	A3=26,000
④	18,000	-20,000	B3=6,000	A4=24,000
⑤	30,000	-24,000	B4=0	A1=30,000

API	Instruction			Operand	Description
2724	D	ZRN2		**TFreq, JFreq, Mode, DOG, NL, Pulse, Dir**	Zero return 2 (directional output can be defined)

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
TFreq							●	●	●		○		○	○		
JFreq							●	●	●		○		○	○		
Mode							●	●	●		○		○	○		
DOG	○		○													
NL	○		○													
Pulse		○														
Dir		○	○													

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
TFreq			●				●						
JFreq			●				●						
Mode			●				●						
DOG	●												
NL	●												
Pulse	●												
Dir	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol

TFreq : Target frequency for zero return

JFreq : JOG frequency for DOG

Mode : Zero return mode

DOG : Input device for DOG

NL : Input device for Negative Limit point

Pulse : Pulse output device

Dir : Output direction device

Explanation

1. This instruction causes the machine to return to the zero point. The range of the target frequency (**TFreq**) for zero return is between 1 Hz–200 kHz. The JOG frequency (**JFreq**) should be less than the target frequency (**TFreq**). The JOG frequency (**JFreq**) is the start frequency. If the target frequency (**TFreq**) is less than (**JFreq**), the target frequency (**TFreq**) is automatically revised processed as equal to (**JFreq**).

2. Do not change the operands **DOG, NL, Pulse, or Dir** during instruction execution. The input point for **DOG and NL** is suggested to use the 16 high-speed input points X0–X7 and X10–X17. They will not be affected by PLC instruction scan time. If you use X20 successive input points or M devices, they will be affected by the PLC instruction scan time.

3. See the below descriptions for DOG, NL and Dir.

DOG point	You can use X or M device but do NOT choose the same input point for different axis. When the DOG point uses X0–X7 and X10–X17 for inputs, go to HWCONFIG to set up the filtering time for the inputs if the switch starts to rattle.
NL point	You can use X or M device but do NOT choose the same input point for different axis.
Dir contact	You can use Y or M device but do NOT choose the same input point for different axis.

4. See the below SR and SM table for pulse outputting.

Pulse output points	**Y0**	**Y2**	**Y4**	**Y6**
Setting up the time for directional outputting goes first before pulse outputting (unit: 1 ms)	SR640	SR642	SR644	SR646
Busy flag	SM460	SM480	SM500	SM520
Completion flag	SM461	SM481	SM501	SM521
Present output position	SR460 SR461	SR480 SR481	SR500 SR501	SR520 SR521

Pulse output points	**Y1**	**Y3**	**Y5**	**Y7**
Setting up the time for directional outputting goes first before pulse outputting (unit: 1 ms)	SR641	SR643	SR645	SR647
Busy flag	SM472	SM492	SM512	SM532
Completion flag	SM473	SM493	SM513	SM533
Present output position	SR474 SR475	SR494 SR495	SR514 SR515	SR534 SR535

5. Select the zero return mode. The function code is set by the two high and low16-bit parameters. See the following table for details.

Select the zero return mode				
High 16-bit	**Low 16-bit**			
b31~b16	**b15~b6**	**b5**	**b4**	**b3~b0**
Number of pulses for motion	reserved	Setting NL signal 0: contact A 1: contact B	Setting DOG signal 0: contact A 1: contact B	Mode settig 0~2; 8~10

6. The function code is set by the two high and low 16-bit parameters. See the following table for details.

Functions	Code		Description
	High 16-bit	**Low 16-bit**	
Leaves the zero point in the negative direction and then stops (Mode 0)	0	0	When the instruction is executed, the search for the zero point is in the negative direction with the target frequency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency is decreased to the JOG speed and the motion in the negative direction continues, and does not stop until the zero point signal changes from ON to OFF.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	1	When the instruction is executed, the search for the zero point is in the negative direction with the target frequency. When the zero point is ON (the zero point signal changes from OFF to ON), the frequency decreases to 0 immediately, and then the motion is in the positive direction at the JOG speed, and does not stop until the zero point signal changes from ON to OFF.
Mode 0 Moves again after returning to the zero point	Number of pulses for motion	2	Returning to the zero point is the same as that for the low 16-bit code. After the zero point is ON, the motion continues according to the number of specified pulses. When the high 16-bit code is a positive number, the search is in the positive direction. A negative value means that the search is in the negative direction.
Leaves the zero point in the positive direction and then stops (Mode 1)	0	8+0=8 8+1=9 (bit3=ON)	Refer to mode 1. The operation for zero point return is the same as that for code 1 (mode 1).
Mode 1 outputs the number of pulses after returning to the zero point	Number of pulses for motion	8+2=10 (bit3=ON)	The operation for zero point return is the same as that for low 16-bit code 1. After returning to the zero point, the motion continues in accordance with the number of specified pulses. When the value of the high 16-bit code is a positive number, the motion is in the positive direction. A negative number indicates

Functions	Code		Description
	High 16-bit	Low 16-bit	
			that the motion is in the negative direction.
DOG point is B point		+16 (bit4=ON)	When in the low 16-bit code, bit 4 is ON, it means the zero point is ON as the DOG point changes from ON to OFF and the zero point is left as the DOG point changes from OFF to ON.
NL point is B point		+32 (bit5=ON)	When in the low 16-bit code, bit 5 is ON, it means the zero point is ON as the NL point changes from ON to OFF and the zero point is left as the DOG point changes from OFF to ON.

7. The execution sequence is based on the value of the low 16-bit code in the table above, and described below.

   →The direction of DOG signal is determined by the value of bit 5.

   → The NL negative limit and DOG signal are determined by the value of bit 5 and bit 4.

   → Mode 0 or mode 1 for the zero point return, selected according to the value of bit 3.

   → The operation of the zero point return is performed according to the values of bit 1 and bit 0.

8. The Completion flag is set to ON after the instruction finishes performing the specified function. For example, for function code 2, the PLC sets the Completion flag to ON only when the number of specified pulses completes outputting.

9. After the DZRN2 instruction is executed, an interrupt service program is not executed till the DZRN2 instruction is disabled, if the specified input point for the zero point is the same as that for the external input interrupt in the program.

10. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

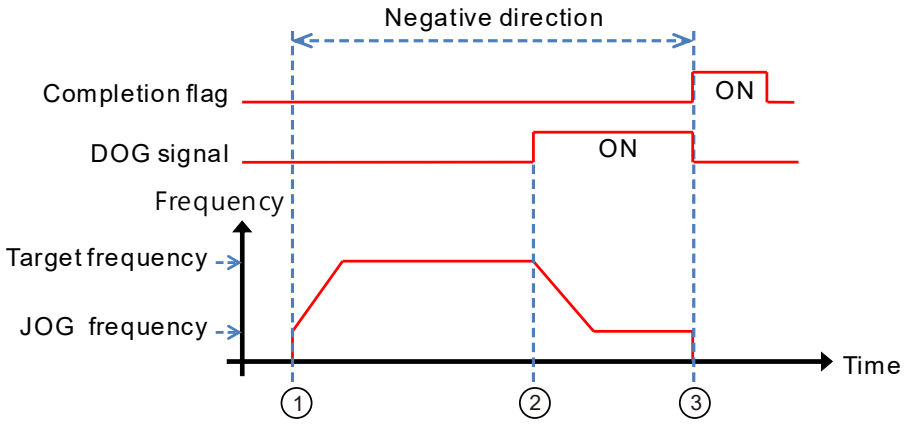
The steps for performing the functions are as below

Function code 0:

① The DZRN2 function is executed and the search for the zero point is in the negative direction with the target frequency (**TFreq**).

② After the DOG signal is received, the output frequency decreases to the JOG frequency (**JFreq**). The output continues in the negative direction and does not stop until the zero point signal changes from ON to OFF.

③ The output stops when the signal changes from ON to OFF and the axis moves away from the DOG signal.

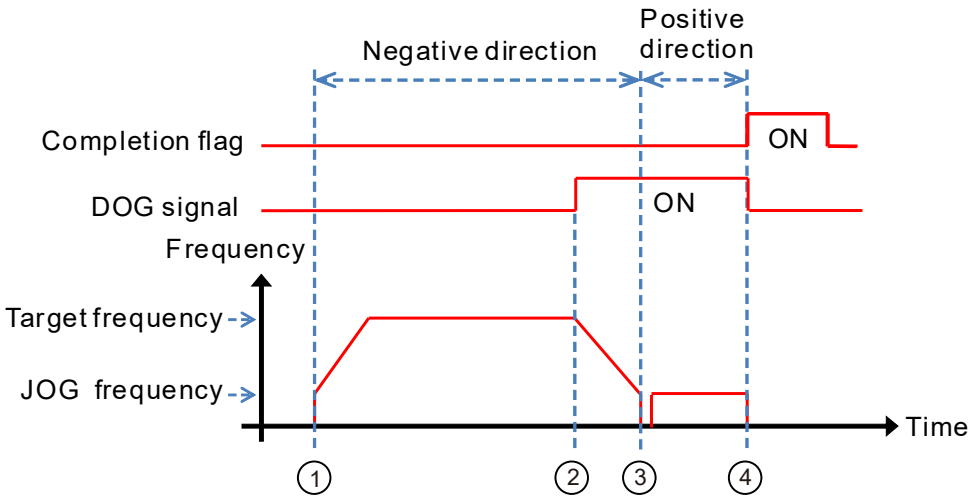


Function code 1:

① The DZRN2 function is executed and the search for the zero point is in the negative direction with the target frequency (**TFreq**).

② After the DOG signal is received, the output is in the positive direction with the JOG frequency (**JFreq**) after the output frequency decreases, and the motion direction reverses. The output does not stop until the zero point signal changes from ON to OFF.

③ The axis moves away from the DOG signal and PLC stops when the signal changes from ON to OFF.



Function code 2:

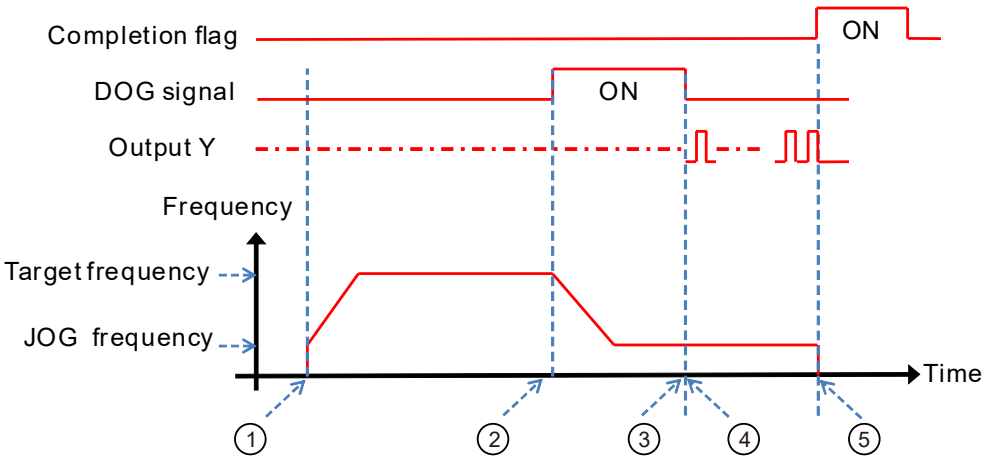
① The DZRN2 function is executed and the search for the zero point is in the negative direction with the target frequency (**TFreq**).

② After the DOG signal is received, the output decreases the frequency to the JOG frequency (**JFreq**) and continues in the negative direction.

③ When the DOG signal is left and the signal changes from ON to OFF, the specified number of pulses are output.

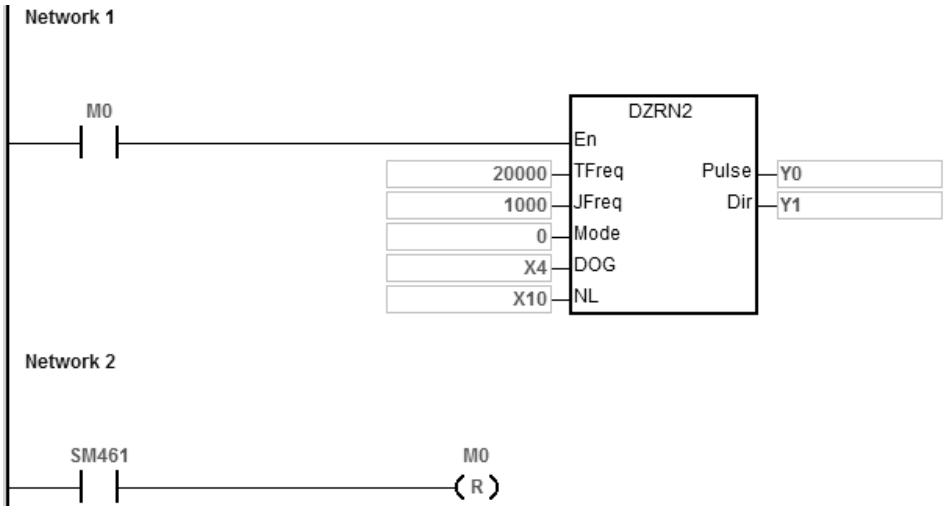
④ The first pulse output starts.

⑤ When the 100th pulse output completes, the PLC stops and the Completion flag is ON.



Example

When M0 is ON, outputting the pulse from Y0 with a frequency of 20 kHz to search for the zero point in the negative direction. When the DOG signal is received and X4 is ON, it keeps moving in the negative directon with the JOG frequency of 1 kHz. The output stops immediately after X4 changes from ON to OFF.



Explanation

Mode is set to 0.

High 16-bit [0000] is to disable the function to move a number of pulses..

Low 16-bit [0000] is Mode 0; when the DOG signal is received, the axis moves in the negative direction; after the axis moves away from the DOG signal, it stops immediately.

6.27 Delta CANopen Communication Instructions

6.27.1 List of CANopen Communication Instructions

The following table lists the High-speed Output instructions covered in this section.

API	Instruction code		Pulse instruction	Function	Delta specific	
	16-bit	32-bit			Servo	Inverter
2800	INITC	–	–	Initializing the servo for CANopen communication	V	V
2801	ASDON	–	–	Servo-ON and Servo-OFF	V	V
2802	CASD	–	–	Setting the acceleration time and deceleration time for a servo	V	V
2803	–	DDRVIC	–	Servo relative position control	V	–
2804	–	DDRVAC	–	Servo absolute position control	V	–
2805	–	DPLSVC	–	Servo speed control	V	V
2806	ZRNC	–	–	Servo homing	V	–
2807	COPRW	–	–	Reading and writing CANopen communication data	V	V
2808	COPWL	DCOPWL	–	Writing multiple CANopen parameter values	V	V
2809	RSTD	–	–	Sending Reset or NMT command	V	V
2810	ZRNM	–	–	Setting the homing mode for Delta servo	V	–
2811	EMER	–	–	Reading Emergency message	V	V
2812	–	DCSFOC	–	Controlling the tracking function of a servo via communication	V	–

6.27.2 Explanation of CANopen Communication Instructions

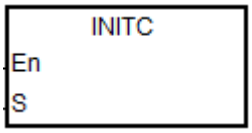
API	Instruction	Operand	Description
2800	INITC	**S**	Initializing the servo for CANopen communication

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●											

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



S : Number of station to be initialized

Explanation

1. Before executing the instruction, be sure to set up CANopen Communication settings, including work mode: Delta Special Driver or Delta Special Driver & CANopen DS 301 and bit rate.
   - Work Mode: Go to HWCONFIG -> Settings -> Options -> Built-in CAN Communication -> CAN port work mode : Delta Special Driver / Delta Special Driver & CANopen DS 301.
   - Bit rate: Go to HWCONFIG -> Settings -> Options -> Built-in CAN Communication -> CAN bit rate -> 10K~1000K.
2. The servo range of **S** is 1–8. When the input value is greater than 8, PLC automatically processes 8 as value of **S** for the initialization. The station address must start at 1 and the following addresses cannot be skipped or reserved.
3. **S** ranging from 21 to 28 is added for inverters. For the input value that is not in the range of 21 to 28, this instruction will not be executed. Same rule applies to the servo; the station address must start at 21 and the following address should be in consecutive order.
4. When setting the number of station to be initialized to 5, this instruction initializes station address from 1 to 5. And when setting the number of station to be initialized to 23, this instruction initializes station address from 21 to 23. When there are 5 servo systems and 3 inverters that need to be initialized, you need to execute this instruction twice. One is INITC 5 for 5 servo systems and the other is INITC 23 for 3 inverters.

5. SM1681 is set to OFF when you execute this instruction on servos (**S** is 1–8). SM1681 is set to ON when the initializations are complete. SM1683 is set to OFF when you execute this instruction on inverters (**S** is 21–28). SM1683 is set to ON when the initializations are complete.

6. SM1682 is set to ON when an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code. (The flag and code here are especially for Delta CANOpen Communication Instructions.)

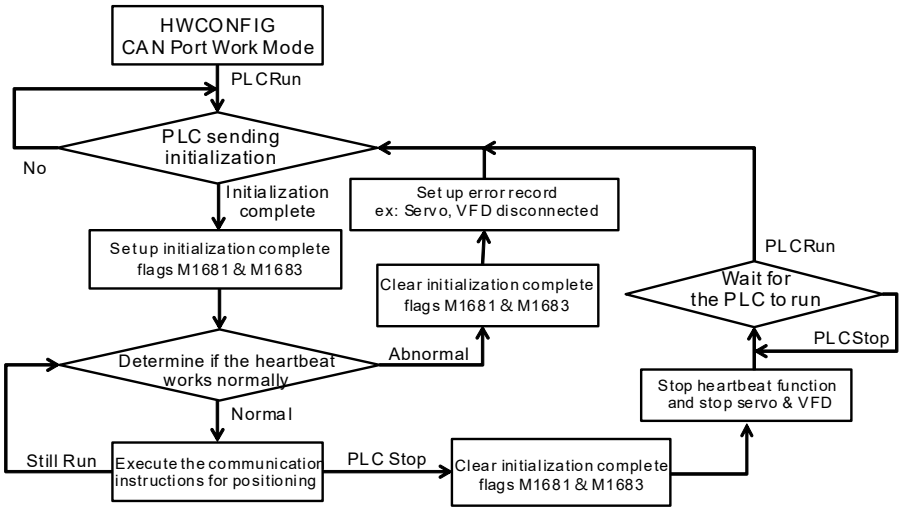
7. If a heartbeat error (SR0 = 0x1900~0x191C; last 2 codes are the slave ID) occurs after the initialization is complete and the SM1684 is OFF (default, indicating when one goes down, all the servos are OFF), the initialization complete flags SM1681 & SM1683 will be cleared to OFF and related actions on other slaves will also be paused. After all the troubles are cleared, you need to initialize every slave to restart the operation. PLC will turn the error LED off after PLC confirms the initialization is complete. This error will be recorded in the error log.

8. If the axes are working independently and the communication is working properly, you can set the SM1684 to ON (indicating when one goes down, only the defective servo is OFF) to notify PLC to record the specific error on the error log and other slaves can keep working.

9. You need to set the communication station and speed of the servos manually and then turn the servos off and then on again to activate the new parameters.

10. This instruction should not be used in the Structured Texts programming language, or interrupt programs or FB programs that only be called once.

Initialization and operation process chart



Example of Communication with Delta servo ASD-A2M

1. Before executing the instruction, be sure to set up CANopen Communication settings, including work mode: Delta Special Driver or Delta Special Driver & CANopen DS 301 and bit rate.

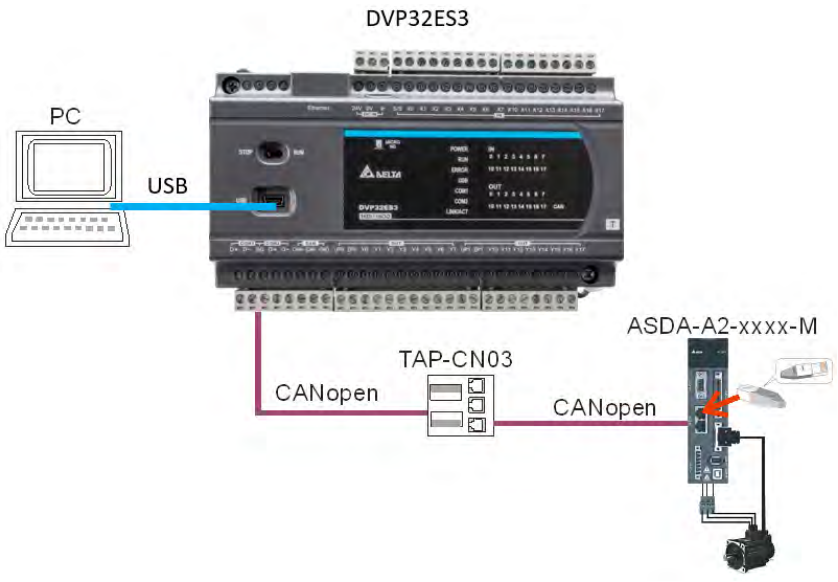
   - Work Mode: Go to HWCONFIG -> Settings -> Options -> Built-in CAN Communication -> CAN port work mode : Delta Special Driver / Delta Special Driver & CANopen DS 301.

Parameter name	Value	Unit	Default	Minimum	Max
CAN port work mode	Delta Special Driver		Delta Special	-	-
CAN port node ID	Delta Special Driver			1	127
CANopen communication time out	CANopen DS301			0	3000
CAN bit rate	Delta Special Driver & CANopen DS301 mode	bps		-	-
Communication data sampling position	Auto		Auto	-	-
CANopen DS301 mode data exchange	Start after power-on		Start after pow	-	-
CAN Hardware error counter	Enable		Enable	-	-

   - Bit rate: Go to HWCONFIG -> Settings -> Options -> Built-in CAN Communication -> CAN bit rate -> 10K~1000K.

Parameter name	Value	Unit	Default	Minimum	Maxi
CAN port work mode	Delta Special Driver		Delta Special	-	-
CAN port node ID	1		1	1	127
CANopen communication time out	100	ms	100	0	3000
CAN bit rate	125k	bps	125k	-	-
Communication data sampling position	10k / 20k / 50k / 125k / 250k / 500k / 1000k		Auto	-	-
CANopen DS301 mode data exchange			Start after pow	-	-
CAN Hardware error counter			Enable	-	-

1. Connect the ES3 Series PLC to an ASDA series with a CANopen communication cable as shown in the figure below.



2. Follow the steps below for the basic settings on the panel of the ASD-A2.

   a. Set the servo parameter P2-08 to 10 to restore the factory settings.

   b. Power the servo off and back on again.

   c. Set P1-01 to 0001 (PR mode).

   d. Set P3-01 to 0400 and the baud rate of the servo for CANopen communication to 1.0 Mbps. The baud rate must be the same as that of the PLC.

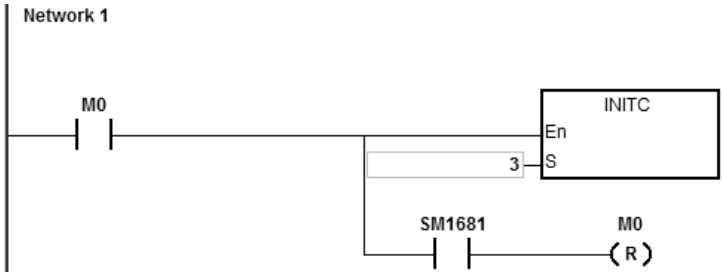
   e. Set a station address for every servo, based on the number of servos. Set P3-00 of each servo to 1, 2, and 3 in order. You can set a maximum of eight servos.

   f. Power the servo off and back on again.

   g. Begin operation after the basic setting is complete.

3. Download the sample program and set M0 to ON. The instruction initializes the servos at station addresses 1–3. When SM1681 is ON, the initialization is complete. When the servo enters CANopen mode successfully, Co-Ld information is displayed.

Ladder diagram:



Structured Texts programming language:

```
0001 IF M0 = TRUE AND SM1681 = FALSE THEN
0002     INITC(3);
0003 END_IF;
0004 IF M0 = TRUE AND SM1681 = TRUE THEN
0005     M0 := FALSE;
0006 END IF;
```

4. The following list shows the settings to initialize a servo drive in the ES3 Series.

   A. Set P2-30 (auxiliary function) to 5 to indicate that the servo does not need to store the settings in EEPROM permanently. This can prolong the servo life span.

   B. Reset P6-02 (PATH#1) to 0 and P6-06 (PATH#3) to 0. This indicates that PATH#1 & #3 in PR mode are both cleared.

   C. Set P3-06 (SDI source) to 16#0100. This indicates that DI1–DI8 are controlled by the hardware, EDI9 is

controlled by the software, and EDI10–EDI14 are controlled by the hardware.

D. Reset P4-07 (SDI status controlled manually) to 0.

E. Set P2-36 (EDI9) to 16#0101. This indicates that the function of EDI9 is set to Servo ON.

F. Set P0-17 (CM1A) to 1. This indicates that the mapping parameter is the pulse command output register CMD_O.

G. Set P0-18 (CM2A) to 64. This indicates that the mapping parameter is the pulse command register CMD_E.

H. Set P5-20–P5-35 (acceleration time) to 1. This indicates that the acceleration time is 1 ms.

I. Set P5-60–P5-75 (target speed) to 1. This indicates that the target speed is 0.1 rpm.

J. Set PDO1 to correspond to P5-07 (PR command), P0-01 (Fault code), P0-46 (state of DO point) and P4-07 (state of DI point)

K. Set PDO2 to correspond to P0-09 (CM1 state: CMD_O) and P0-10 (CM2 state: CMD_E).

L. Set PDO3 to correspond to P0-11 (CM3 state: state of current torque)

M. Set time for the heartbeat and the PLC scan overtime to 200ms (default). When a communication overtime occurs, PLC will send error message (error LED flashing slowly; refer to error code for more details) and the servo turns off automatically.

N. If SM1685 is ON, PDO3 reads self-defined parameters from servo drive P0-12 and stores the data in SR791-SR806. You need to set up the P0-20 (CM4A) before executing. 2 SRs are reserved for every servo drive axis and you can define their data types.

5. Do not use the COPRW instruction (API 2807) to modify the servo parameters of the eight items A, B, F, G, J, K and N above after initialization.

6. When you use an absolute-type servo, use the COPRW communication instruction to write 16#0100 to P3-12, which writes the relevant absolute-type servo parameters to EEPROM at the moment the servo powers off.

7. Set the relevant DI signal configuration parameters manually or with the COPRW instruction to modify the hardware DI signal setting of ASD-A2. Use COPRW to modify the configuration after execution of the INITC instruction is complete and before the servo is enabled.

8. When the initialization is complete, the servo is in the PR mode. Do not make any communication control on servo P5-18.

9. For more details on the servo parameters, refer to the Delta Servo Operation manual.

Example of Communication with Delta Inverter

1. Manually set the inverter parameter P09-36 (ranging from 21 to 28), P09-37 (communication baud rate default is 1MHz) and P09-40 to standard CANopen protocol (default).

2. Manually set the source of frequency command P00-20 to 6 and source of the operation command P00-21 to 3, and have them controlled by CANopen communication.

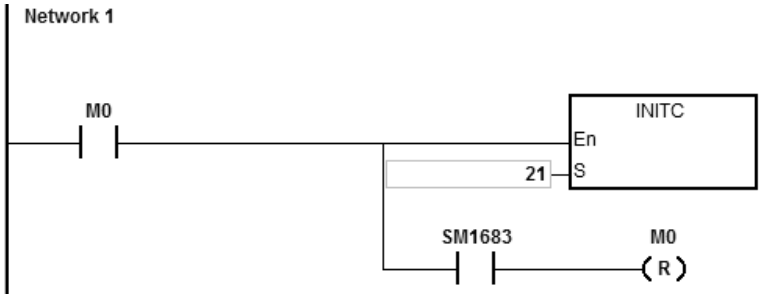
3. The following list shows the settings to initialize an inverter in the ES3 Series.

   A. Set PDO mapping area: the operation frequencies, current state, and states of torque and DI point correspond to PLC SRs.

   B. Set time for the heartbeat and the PLC scan overtime to 200ms (default). When a communication overtime occurs, PLC will send error message (error LED flashing slowly; refer to error code for more details) and the servo turns off automatically.

4. For more details on the servo parameters, refer to the Delta Inverter Operation manual.

Example



1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the inverter at station address 21, until SM1683 is ON. When SM1683 is ON, it indicates the initialization is complete.

Additional Remarks:

When you use Delta Special Driver and execute INITC instruction, the baud rate and time to synchronize is shown below.

If you use Delta Special Driver & CANopen DS301 modes, you can use CANopen Builder to modify the time to synchronize, but be sure to modify the time less than it is shown below.

Baud Rate (bps)	1M	500K	250K	125K
Time to Synchronize (ms)	10	15	30	60

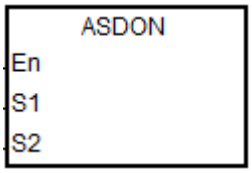
API	Instruction	Operand	Description
2801	ASDON	S_1, S_2	Servo-ON and Servo-OFF

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2													○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



S_1 : Station address of the servo

S_2 : Servo ON and Servo OFF

Explanation

1. This instruction changes the state of the servo at the address in S_1. The INITC instruction must be complete before this instruction is executed.
2. The range of S_1 is 1–8 (for servo) and 21-28 (for inverter). There will be no execution when the input value is out of the range.
3. The range of S_1 is 1–8 (for servo) and S_2 is a non-zero value, the servo is enabled (Servo-ON). If S_2 is 0, the servo is disabled (Servo-OFF).
4. The range of S_1 is 21–28 (for inverter). When S_2 is 1, it indicates it is in inverter speed mode. When S_2 is 2, it indicates it is in inverter torque mode. When S_2 is 0, it indicates it stops operating. Since control modes (speed mode and torque mode) for inverters can be switched directly, the operation can keep going. Before switching control modes, make sure all the inverter parameters are set.

5. Table of the last 2 points:

Drive type	Range of S_1	Descriptions of S_2
Servo	1-8	1 = SERVO-ON 0 = SERVO-OFF
Inverter	21-28	0 = stop operating 1 = Speed mode 2 = Torque mode

6. Every servo (1-8) has a flag (SM1651–M1658) for displaying its state. You can read the actual servo state from the flag. When the flag is ON, the servo is Servo-ON. When the flag is OFF, the servo is Servo-OFF.

7. Every inverter (21-28) has a flag (SM1621–M1628) for displaying its state. You can read the actual inverter state from the flag. When the flag is ON, the inverter is ON. When the flag is OFF, the inverter is OFF.

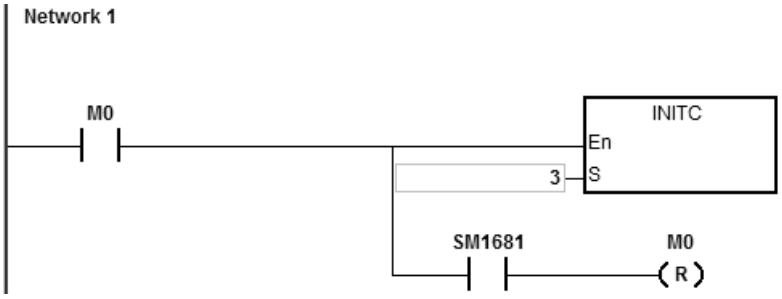
8. SM1682 is ON if an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code.

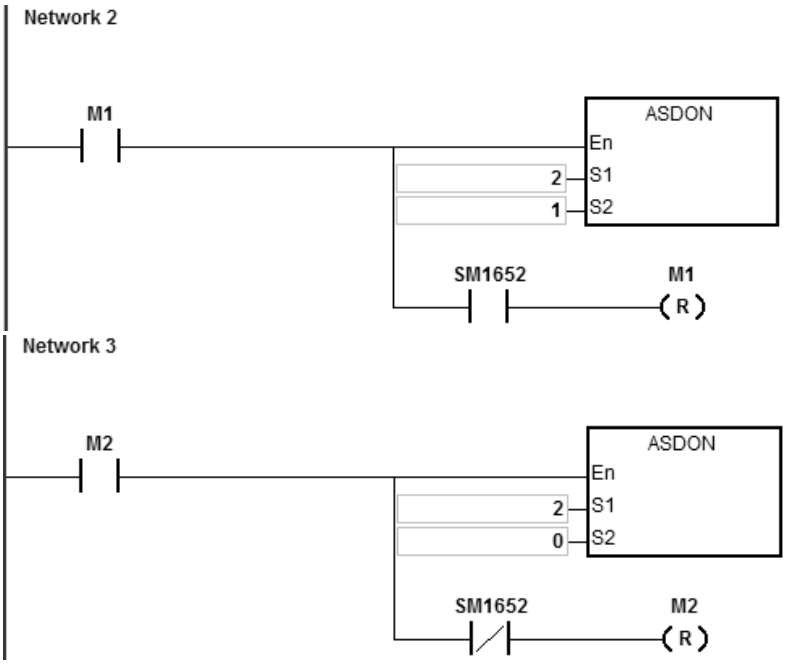
Example 1 (Ladder diagram):

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the servos at station addresses 1–3, until SM1681 is ON.

2. When M1 changes from OFF to ON, the ASDON instruction starts to enable the servo at station address 2. When SM1652 is ON, it indicates Servo-ON.

3. When M2 changes from OFF to ON, the ASDON instruction starts to disable the servo at station address 2. When SM1652 is OFF, it indicates Servo-OFF.





Example 2 (Structured Texts programming language):

```
0001 IF M1 THEN
0002    ASDON(2,1);
0003 END_IF;
0004 IF M1 AND SM1652 THEN
0005    SM1582 := TRUE;
0006    M1 := FALSE;
0007 END_IF;
0008
0009 IF M2 THEN
0010    ASDON(2,0);
0011 END_IF;
0012 IF M2 AND (not SM1652) THEN
0013    SM1582 := TRUE;
0014    M2 := FALSE;
0015 END_IF;
```

1. When M1 changes from OFF to ON, this instruction starts to enable the servo at station address 2. When SM1652 is ON, it indicates Servo-ON.

2. Make sure M1 and SM1652 are both ON and then set the auto return communication control right flag SM1582 to ON for station address 2. This will free the communication control right and the station address 2 can receive and execute other communication instructions.

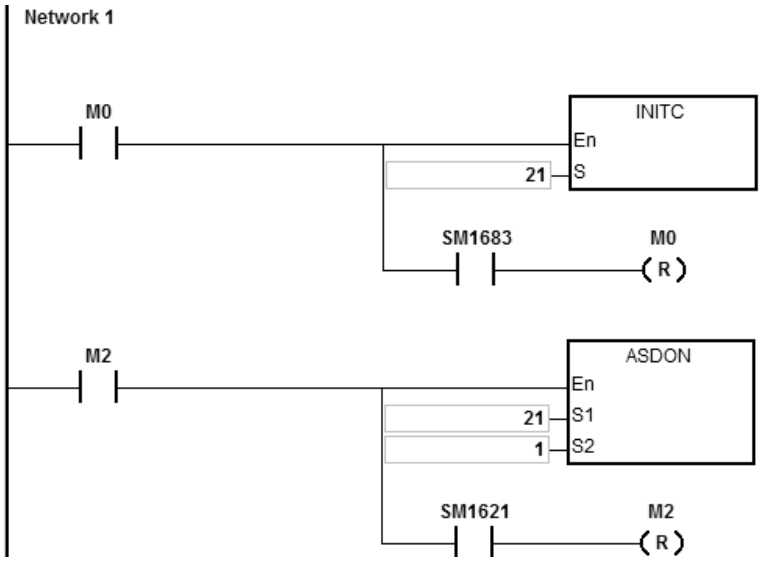
3. When M2 changes from OFF to ON, this instruction starts to disable the servo at station address 2. When SM1652 is OFF, it indicates Servo-OFF.

4. Make sure M2 is ON and SM1652 is OFF and then set the auto return communication control right flag SM1582 to ON for station address 2. This will free the communication control right and the station address 2 can receive and execute other communication instructions.

Example 3 (communication with Delta inverters):

1. When M0 changes from OFF to ON, this instruction starts to initialize the station address 21. When SM1683 is ON, it indicates the initialization is complete.

2. When M2 changes from OFF to ON, this instruction starts to set the control mode at the station address 21. When SM1621 is ON, it indicates the inverter is in operation.



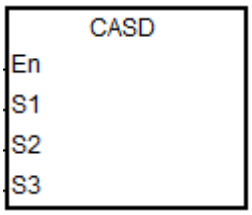
API	Instruction	Operand	Description
2802	CASD	S_1 , S_2 , S_3	Setting the acceleration time and deceleration time for a servo

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1	●	●			●	●		●	●		○	○	○	○		
S_2	●	●			●	●		●	●		○	○	○	○		
S_3	●	●			●	●		●	●		○	○	○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



S_1 : Station address of the servo

S_2 : Acceleration time (ms)

S_3 : Deceleration time (ms)

Explanation

1. This instruction sets the servo acceleration and deceleration time for the servo at the address in S_1. The INITC instruction must be complete before this instruction is executed.

2. The range of S_1 is 1–8 (for servo) and 21-28 (for inverter). There will be no execution when the input value is out of the range.

3. Table to describe S_2 and S_3

Drive type	Range of S_1	Range of S_2 and S_3	Descriptions of S_2 and S_3
Servo	1-8	1-32767 (ms)	S_2: Acceleration time is the period of time during which the servo spins up from 0 to 3000.0 rpm. S_3: Deceleration time is the period of time during which the servo spins down from 3000.0 rpm to 0.
Inverter	21-28	100-32700 (ms) The input value	S_2: Acceleration time is the period of time during which the

		should be multiplied by 100.	inverter spins up from 0.0Hz to the highest operation frequency. **S_3**: Deceleration time is the period of time during which the servo spins down from the highest operation frequency to 0.

4. When you execute this instruction and the range of **S_1** is 1–8 (for servo), SM1681 is set to OFF (parameters are being editing). When the servo responds to the received command, SM1681 is ON (parameters are set successfully). When you execute this instruction and the range of **S_1** is 21–28 (for inverter), SM1683 is set to OFF (parameters are being editing). When the inverter responds to the received command, SM1683 is ON (parameters are set successfully).

5. SM1682 is set to ON when an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code.

Example 1 (communication with Delta ASD-A2M)

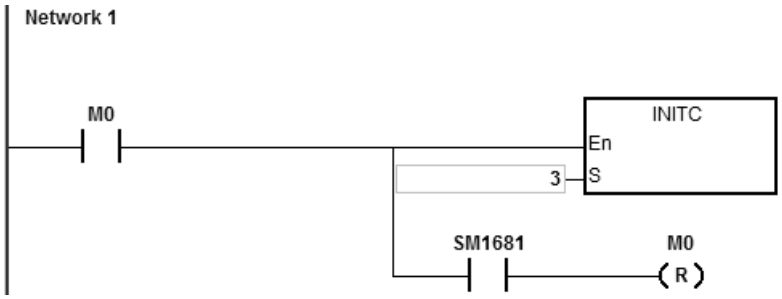
1. When M0 changes from OFF to ON, the INITC instruction initializes the servos at station addresses 1–3, until SM1681 is ON.

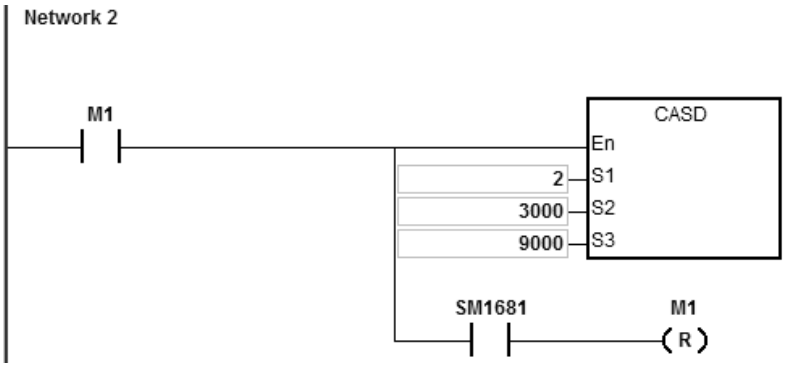
2. When M1 changes from OFF to ON and the target speed of the servo at station address 2 is 3000 rpm, the CASD instruction sets the acceleration time of servo 2 to 3000 ms and the deceleration time to 9000 ms.

3. If the target speed of servo 2 is 1000 rpm, the acceleration time and deceleration time are shown below.

   Acceleration time: [3000 ms / 3000 rpm] × 1000 rpm = 1000 ms

   Deceleration time: [9000 ms / 3000 rpm] × 1000 rpm = 3000 ms

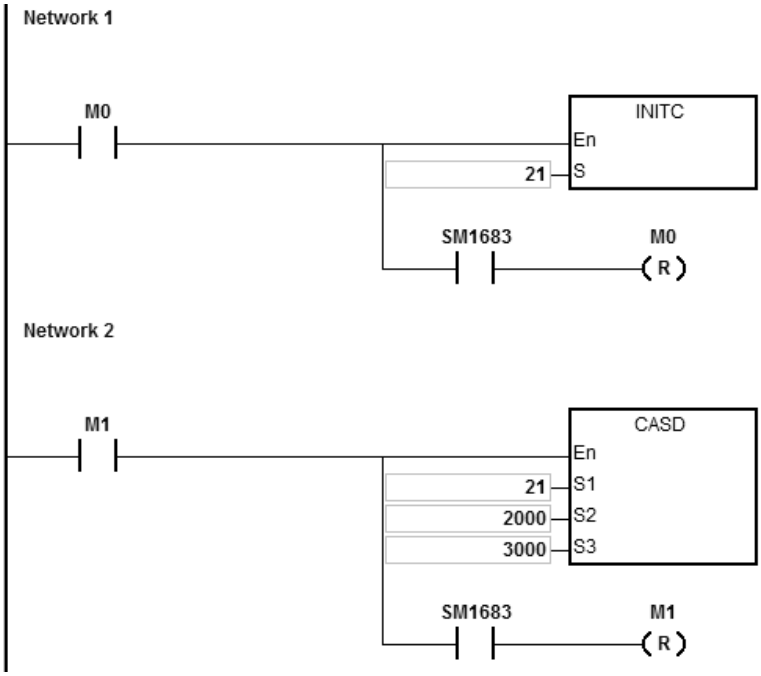




Example 2 (communication with Delta inverters)

1. When M0 changes from OFF to ON, this instruction starts to initialize the station address 21. When SM1683 is ON, it indicates the initialization is complete.

2. When M1 changes from OFF to ON, this instruction starts to set the acceleration time to 2000 ms and deceleration time to 3000 ms at the station address 21.



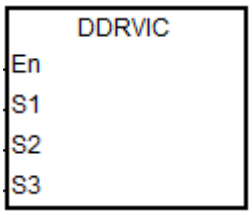
API	Instruction			Operand	Description
2803	D	DRVIC		**S_1, S_2, S_3**	Servo relative position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Station address of servo

S_2 : Relative target position

S_3 : Target speed

Explanation

1. This instruction executes the servo relative position control for the servo at the address specified in **S_1**. The INITC and ASDON (servo ON) instructions must be complete before this instruction is executed.
2. The range of **S_1** is 1–8. There will be no execution when the input value is out of the range.
3. The range of **S_2** is -2147483648 to +2147483647. The +/- sign indicates the forward / reverse direction. The target position is a relative position.
4. The unit of the value of **S_3** is 0.1 rpm. The range is 1–60000, which indicates 0.1–6000.0 rpm.
5. When the target position is reached, the corresponding completion flags of axes SM1631–SM1638 are ON. After the axis starts to output, PLC clears the corresponding completion flag.
6. Refer to the following table for the corresponding SM and SR of the axes.
7. SM1682 is ON when an error occurs during communication. In addition, SR658 retains the number of the axis in

which the error occurs and SR659 retains the error code.

8. This instruction uses SDO (Service Data Object) of CANopen protocol to communicate. Since only one SDO communication can be sent for a time, do NOT use this instruction for outputting positioning in various axis synchronized motion.

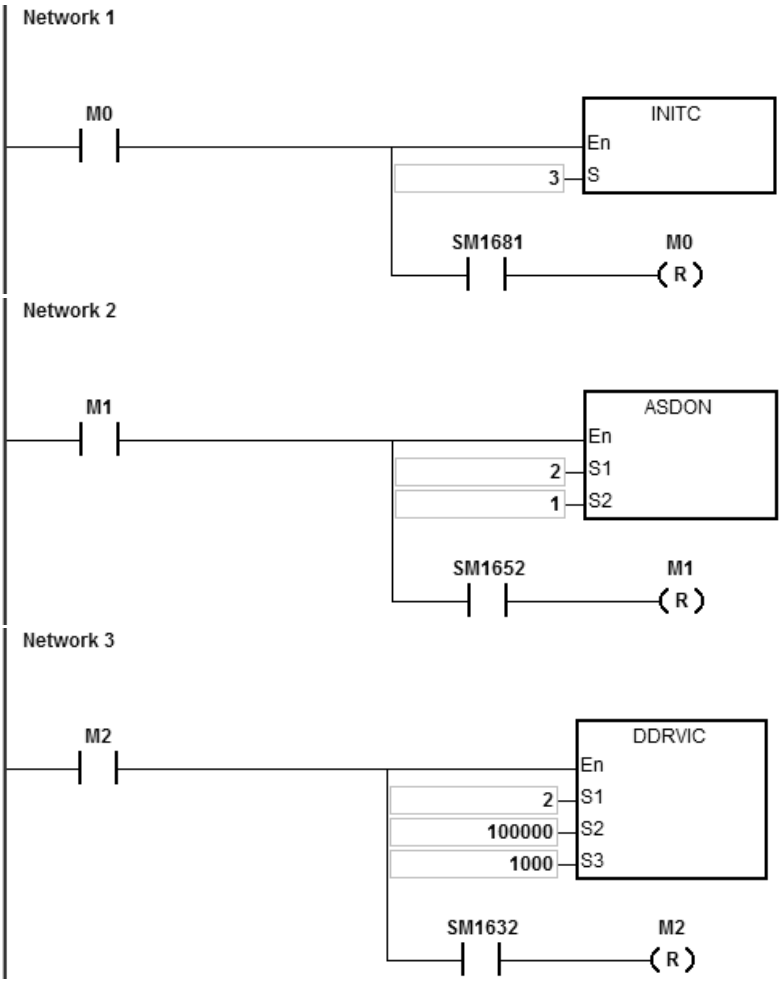
Example 1 (Ladder diagram):

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the servos at station addresses 1–3, until SM1681 is ON.

2. When M1 changes from OFF to ON, the ASDON instruction starts to enable the servo at station address 2. When SM1652 is ON, it indicates Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves to the relative position 100000 PUU at 100.0 rpm.

   The finish flag SM1632 is ON when the target position is reached.



Example 2 (Structured Texts programming language):

```
0001 IF M2 THEN
0002    DDRVIC(2,100000,1000);
0003 END_IF;
0004 IF M2 AND SM1632 THEN
0005    SM1582 := TRUE;
0006    M2 := FALSE;
0007 END_IF;
```

1. When M2 changes from OFF to ON, servo 2 moves to the relative position 100000 PUU at 100.0 rpm.

   The finish flag SM1632 is ON when the target position is reached.

2. Make sure M2 and SM1632 are both ON and then set the auto return communication control right flag SM1582 to ON for station address 2. This will free the communication control right and the station address 2 can receive and execute other communication instructions.

Explanation of special flags (SM) and registers (SR) for Delta special CANopen communication instructions

The following table shows special flags (SM) and registers (SR) related to Delta special CANopen communication.

Flag	R/W	ID. 1	ID. 2	ID. 3	ID. 4	ID. 5	ID. 6	ID. 7	ID. 8
Enable specific function	R/W	The flag in HWCONFIG is set to ON when the PLC changes from Stop to Run.							
Initialization and communication complete (INITC and CASD)	R	SM1681							
CANopen communication error	R	SM1682							
Positioning complete	R/W#1	SM1631	SM1632	SM1633	SM1634	SM1635	SM1636	SM1637	SM1638
Stop	R/W	SM1641	SM1642	SM1643	SM1644	SM1645	SM1646	SM1647	SM1648
Servo-ON	R	SM1651	SM1652	SM1653	SM1654	SM1655	SM1656	SM1657	SM1658
Go-back/go-forth enabled Only DDRVAC is supported.	R/W	SM1661	SM1662	SM1663	SM1664	SM1665	SM1666	SM1667	SM1668
Go-back/go-forth direction indicator Only DDRVAC is supported.	R	SM1671	SM1672	SM1673	SM1674	SM1675	SM1676	SM1677	SM1678
Auto return communication control right	R/W	SM1581	SM1582	SM1583	SM1584	SM1585	SM1586	SM1587	SM1588
Heartbeat error code	R	SM1691	SM1692	SM1693	SM1694	SM1695	SM1696	SM1697	SM1698
Heartbeat error handling	R/W	SM1684 = OFF (default; when one goes down, all the servos are OFF.) SM1684 = ON (when one goes down, only the defective servo is OFF.)							
Number of the axis with a communication error	R	SR658							
Communication error code	R	SR659							

Note1: After the axis starts to output, PLC clears the corresponding completion flag.

The following table shows how Delta servo parameters of axes correspond to special flags and registers in the CANopen communication.

Servo Parameter Name (Number)	R/W	ID. 1	ID. 2	ID. 3	ID. 4	ID. 5	ID. 6	ID. 7	ID. 8
PR command (P5-07)	R	SR661	SR662	SR663	SR664	SR665	SR666	SR667	SR668
Alarm code (P0-01) (hexadecimal)	R	SR671	SR672	SR673	SR674	SR675	SR676	SR677	SR678
DO state (P0-46)	R	SR681	SR682	SR683	SR684	SR685	SR686	SR687	SR688
Servo command positon CMD_O (P0-09)	R	SR691 SR692	SR693 SR694	SR695 SR696	SR697 SR698	SR699 SR700	SR701 SR702	SR703 SR704	SR705 SR706
Servo target position CMD_E (P0-10)	R	SR711 SR712	SR713 SR714	SR715 SR716	SR717 SR718	SR719 SR720	SR721 SR722	SR723 SR724	SR725 SR726
DI state (P4-07)	R	SR731	SR732	SR733	SR734	SR735	SR736	SR737	SR738
Current torque (unit: 0.1%)	R	SR741	SR742	SR743	SR744	SR745	SR746	SR747	SR748
Self-defined (P0-12) * Work with SM1685 Enable/Disable	R	SR791 SR792	SR793 SR794	SR795 SR796	SR797 SR798	SR799 SR800	SR801 SR802	SR803 SR804	SR805 SR806
Positioning completion range setting	R/W	SR811	SR812	SR813	SR814	SR815	SR816	SR817	SR818

*For example, if a servo real position is requested, set the sero values in P0-20, enable SM1685 and then the servo real position can be read in SR791-SR806. For more details, refer to Delta High Resolution AC Servo Drive for Network Communication Applications ASDA-A2 Series User Manual.

The following table shows how Delta inverter parameters of axes correspond to special flags and registers in the CANopen communication.

Description		ID. 21	ID. 22	ID. 23	ID. 24	ID. 25	ID. 26	ID. 27	ID. 28
Initialization complete flag	R	SM1683							
Inverter starting up flag	R	SM1621	SM1622	SM1623	SM1624	SM1625	SM1626	SM1627	SM1628
Inverter Heartbeat error flag	R	SM1611	SM1612	SM1613	SM1614	SM1615	SM1616	SM1617	SM1618
Auto return communication control right	R/W	SM1601	SM1602	SM1603	SM1604	SM1605	SM1606	SM1607	SM1608
Inverter status (Index 6041H-00H)	R	SR751	SR752	SR753	SR754	SR755	SR756	SR757	SR758
Inverter current velocity (Index 6043H-00H)	R	SR761	SR762	SR763	SR764	SR765	SR766	SR767	SR768
Inverter current torque (unit: 0.1%) (Index 2021H-0CH)	R	SR771	SR772	SR773	SR774	SR775	SR776	SR777	SR778
Inverter digital input status (Index 2022H-11H)	R	SR781	SR782	SR783	SR784	SR785	SR786	SR787	SR788

The following table shows the Delta special drier CANopen error codes. (error code in SR659)

Error Code	Cause
0x0002	The slave does not respond to the SDO message.
0x0003	An error occurs in the message received by the slave. This error often occurs when the settings of the COPRW instruction are invalid causing the slave not to receive the complete message.
0x0004	The slave PDO message is not received.
0x0005	An error occurs while using the instruction operand.
0x0006	One of the stations is being used when the INITC instruction is executed.
0x0007	Slave station address exceeds acceptable range of the instruction INITC
0x0008	A slave cannot be reset; check if the communication cable is connected and if the communication of the slave is working properly.

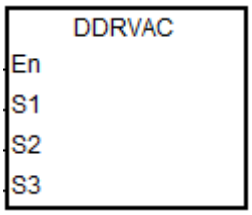
API	Instruction			Operand	Description
2804	D	DRVAC		**S_1, S_2, S_3**	Servo absolute position control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						
S_3			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Station address of servo

S_2 : Absolute target position

S_3 : Target speed

Explanation

1. This instruction executes the servo absolute position control for the servo at the address specified in **S_1**. Before you execute this instruction, the INITC and ASDON (Servo-ON) instructions must be complete.
2. The range of **S_1** is 1–8. There will be no execution when the input value is out of the range.
3. The range of **S_2** is -2147483648 to +2147483647. The +/ - sign indicates the forward / reverse direction. The target position is an absolute position.
4. Refer to the DRVIC instruction (API 2803) for additional information and examples.

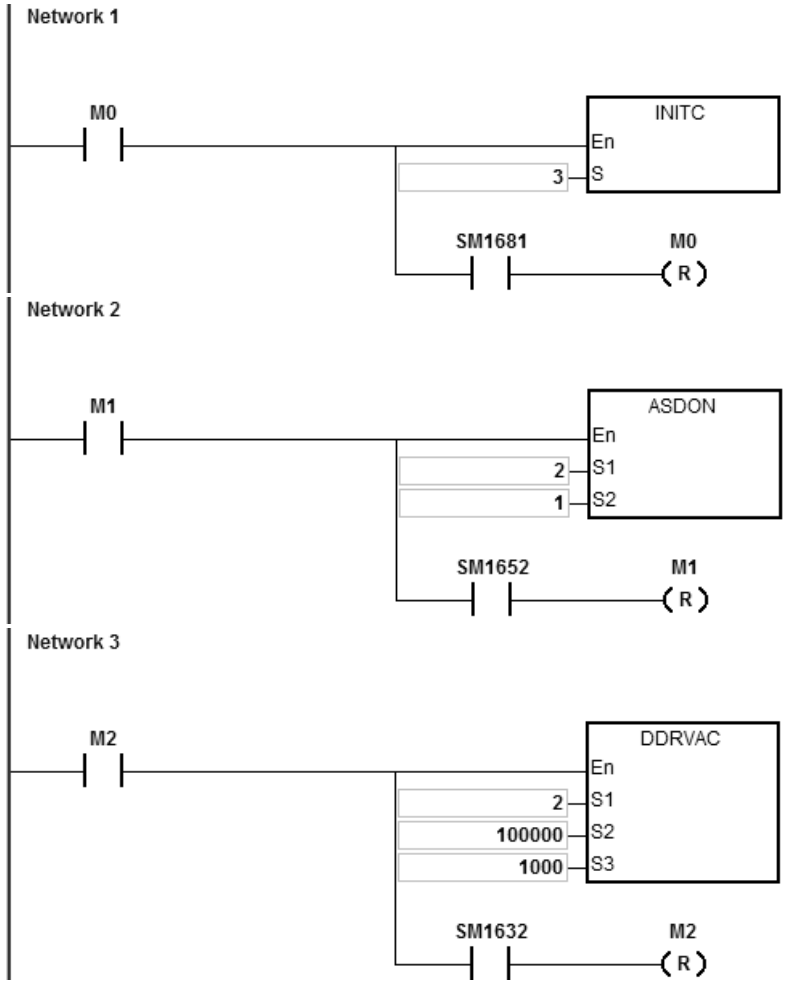
Example 1

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the servos at station addresses 1–3, until SM1681 is ON.

2. When M1 changes from OFF to ON, the ASDON instruction starts to enable the servo at station address 2. When SM1652 is ON, it indicates Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves from the current position to the absolute position 100,000 PUU at 100.0 rpm.

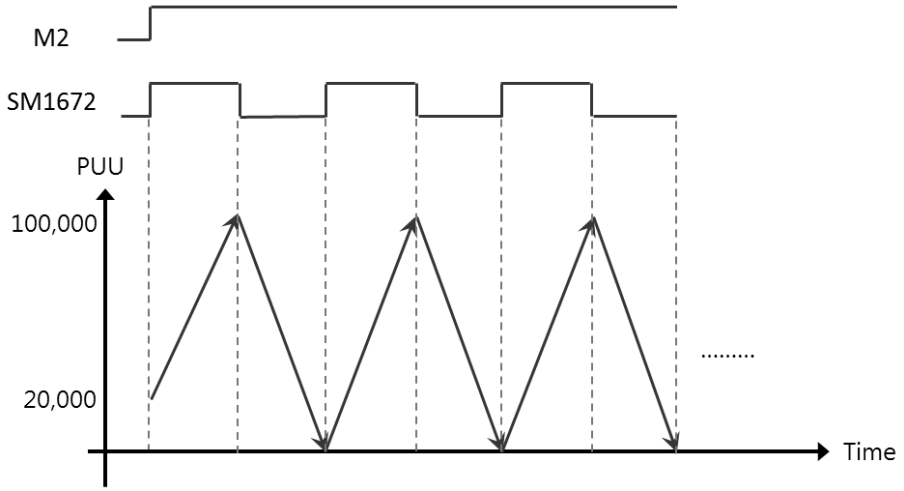
   The finish flag SM1632 is ON when the target position is reached.



Example 2

1. Add one line to the program in Example 1. When the PLC runs and SM1662 is ON, the function is enabled for servo 2 to go back and forth.

2. As the figure shows below, the servo moves from its current position (20,000) to the absolute target position (100,000) after M2 is ON. After that, it goes back and forth between the absolute position 100,000 and 0.
   The direction indication flag SM1672 is ON when the servo goes toward the target position for the first time after Servo-ON. After that, the flag repeats the state, changing from ON to OFF.
3. You can modify the target position at any time in the motion, but the new target position is only valid for the next back and forth cycle.



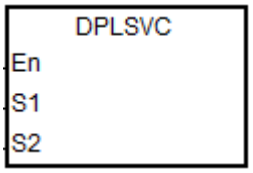
API	Instruction			Operand	Description
2805	D	PLSVC		**S_1, S_2**	Servo speed control

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1			●				●						
S_2			●				●						

Pulse instruction	16-bit instruction	32-bit instruction
—	—	ES3

Symbol



S_1 : Station address of a servo

S_2 Target speed

Explanation

1. This instruction executes the servo speed control for the servo at the address specified in **S_1**. The INITC and ASDON (Servo ON) instructions must be complete before this instruction is executed.

2. The range of **S_1** is 1–8 (for servo) and 21-28 (for inverter). There will be no execution when the input value is out of the range.

3. Table to describe S_2

Servo type	Range of S_1	Range of S_2	Descriptions of S_2
Servo	1-8	-60000 - 60000 (0.1rpm)	The +/- sign indicates the forward / reverse direction. For example, when inputting 6005 in S_2, the value is 600.5rpm.
Inverter	21-28	-32768 - 32767 (1rpm)	The +/- sign indicates the forward / reverse direction. For example, when inputting 605 in S_2, the value is 605rpm.

4. For corresponding SM and SR of the axes, refer to the DRVIC instruction (API 2803).

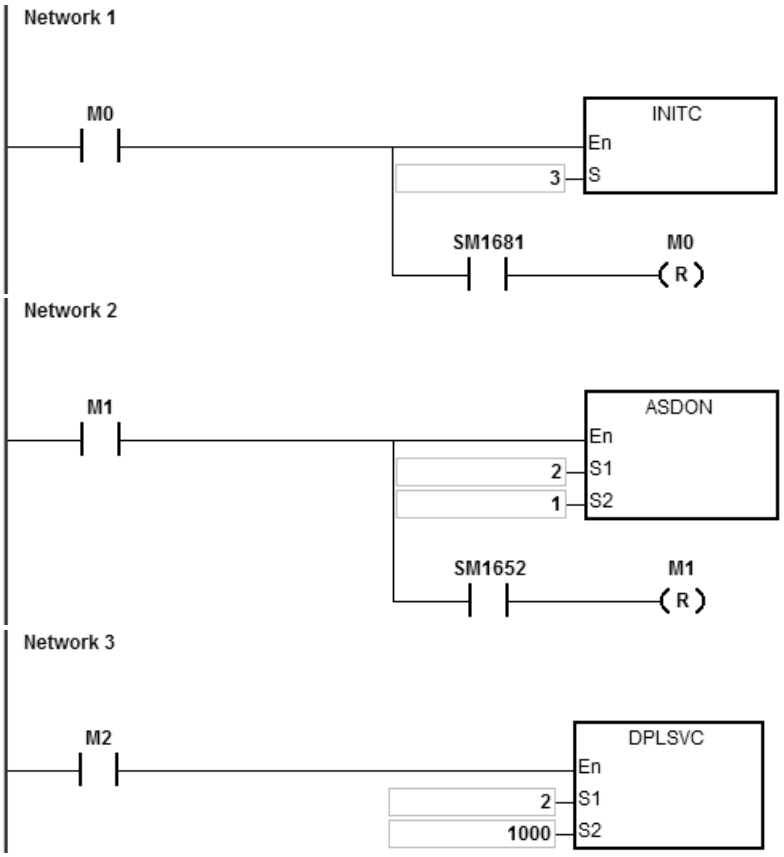
5. SM1682 is set to ON when an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code.

Example 1 (communication with Delta servo systems)

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the servos at station addresses 1–3, until SM1681 is ON.

2. When M1 changes from OFF to ON, the ASDON instruction starts to enable the servo at station address 2. When SM1652 is ON, it indicates Servo-ON.

3. When M2 changes from OFF to ON, servo 2 moves at 100.0 rpm until M2 is OFF.



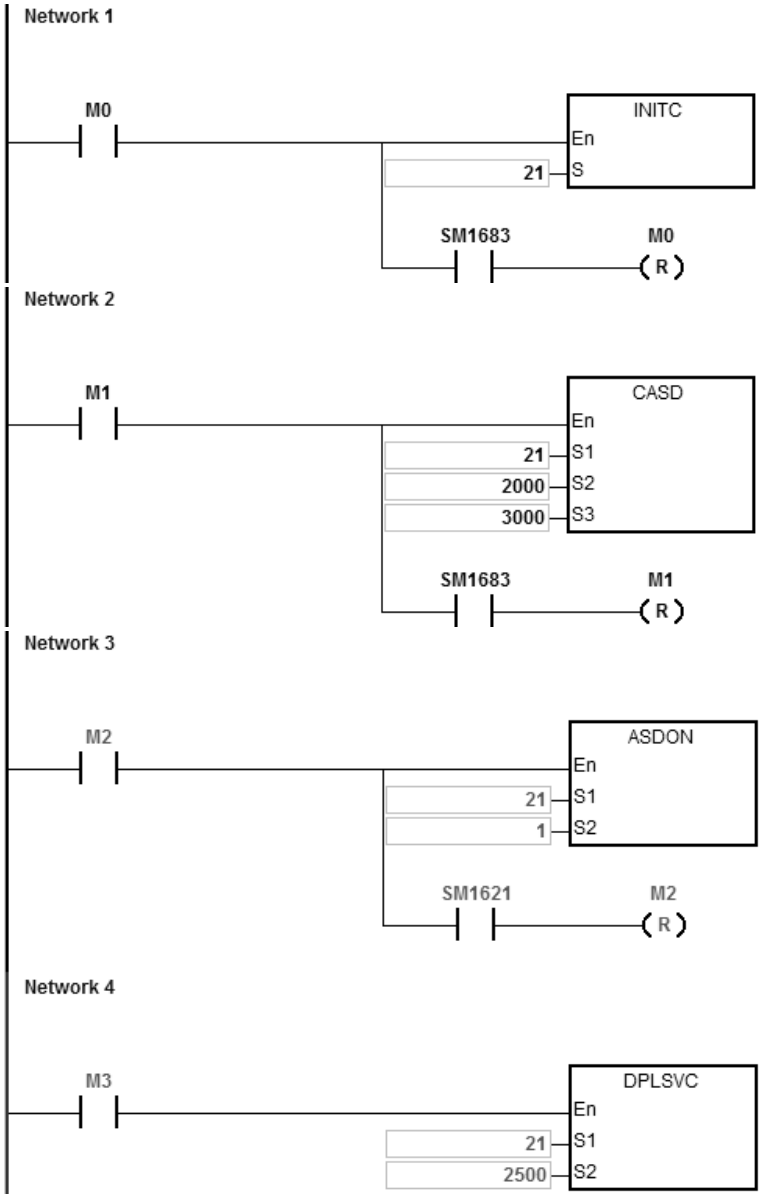
Example 2 (communication with Delta inverters)

1. When M0 changes from OFF to ON, this instruction starts to initialize the station address 21. When SM1683 is ON, it indicates the initialization is complete.

2. When M1 changes from OFF to ON, this instruction starts to set the acceleration time and deceleration time at the station address 21.

3. When M2 changes from OFF to ON, this instruction starts to set the control mode at the station address 21.

4. When M3 changes from OFF to ON, the inverter moves at 2500rpm until M3 goes OFF.



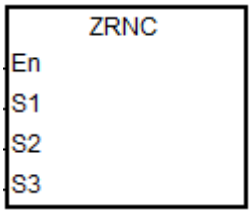
API	Instruction	Operand	Description
2806	ZRNC	**S_1, S_2, S_3**	Servo homing

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							

Pulse instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



S_1 : Station address of servo

S_2 : The 1st – segment speed

S_3 : The 2nd – segment speed

6

Explanation

1. This instruction executes the servo homing control for the servo at the address specified in **S_1**. The INITC and ASDON (Servo-ON) instructions must be complete before this instruction is executed. It is recommended to use the instruction ZRNM (API 2810) to have every servo axis go back to the zero point.
2. The range of **S_1** is 1–8 (for servo. There will be no execution when the input value is out of the range.
3. The range of **S_2** is 1–20000. Unit: 0.1 rpm
4. The range of **S_3** is 1–5000. Unit: 0.1 rpm
5. When the servo returns to the home position, the corresponding finish flags of the axes are ON. Refer to the DRVIC instruction (API 2803) for explanations of special flags (SM) and registers (SR).
6. SM1682 is set to ON when an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code.

Example

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize servo 1, until SM1681 is ON.

2. When M1 changes from OFF to ON, the acceleration time (10000) and deceleration time (20000) of servo 1 are set by using the CASD instruction (API 2802). When SM1651 is ON, it indicates Servo-ON.

3. When M2 changes from OFF to ON,

   2-byte data is written to P5-04 in servo 1. When D100 = 5 and the data is written, M100 is ON.

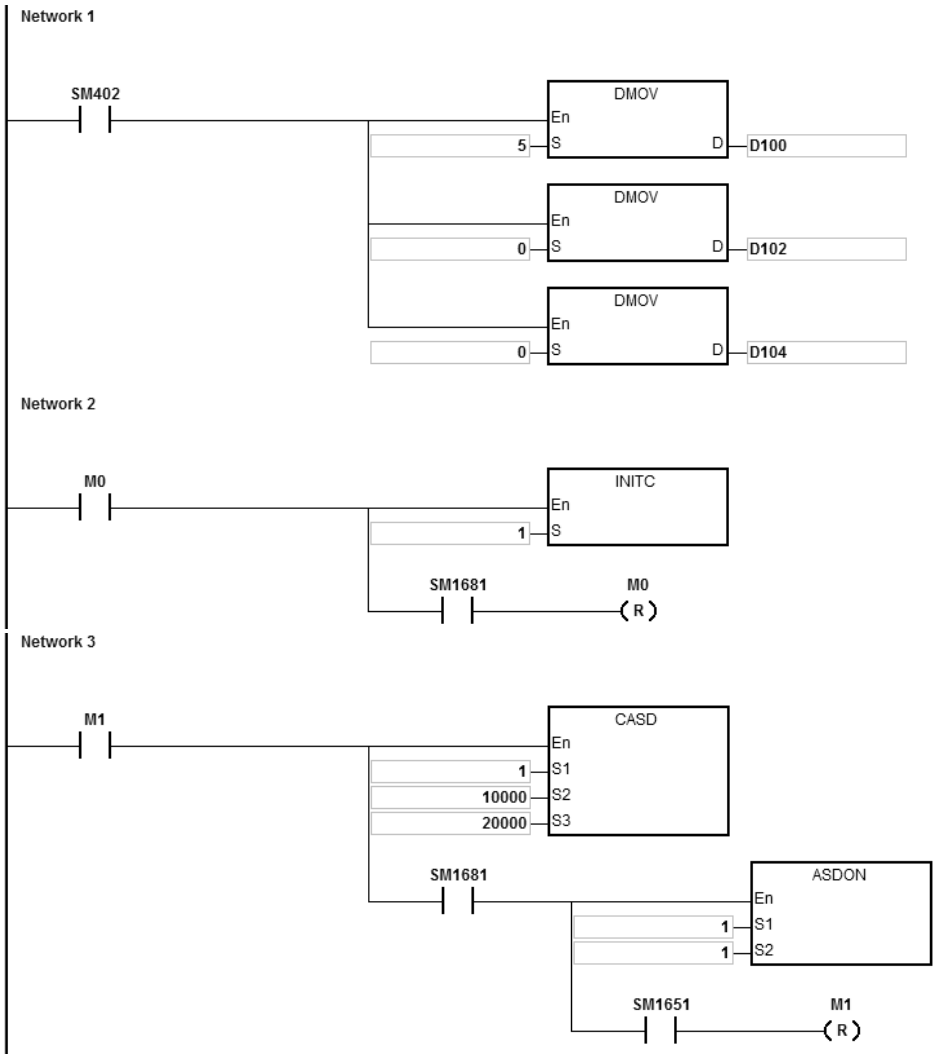
   4-byte data is written to P6-00 in servo 1. When D102 = 0 and the data is written, M101 is ON.

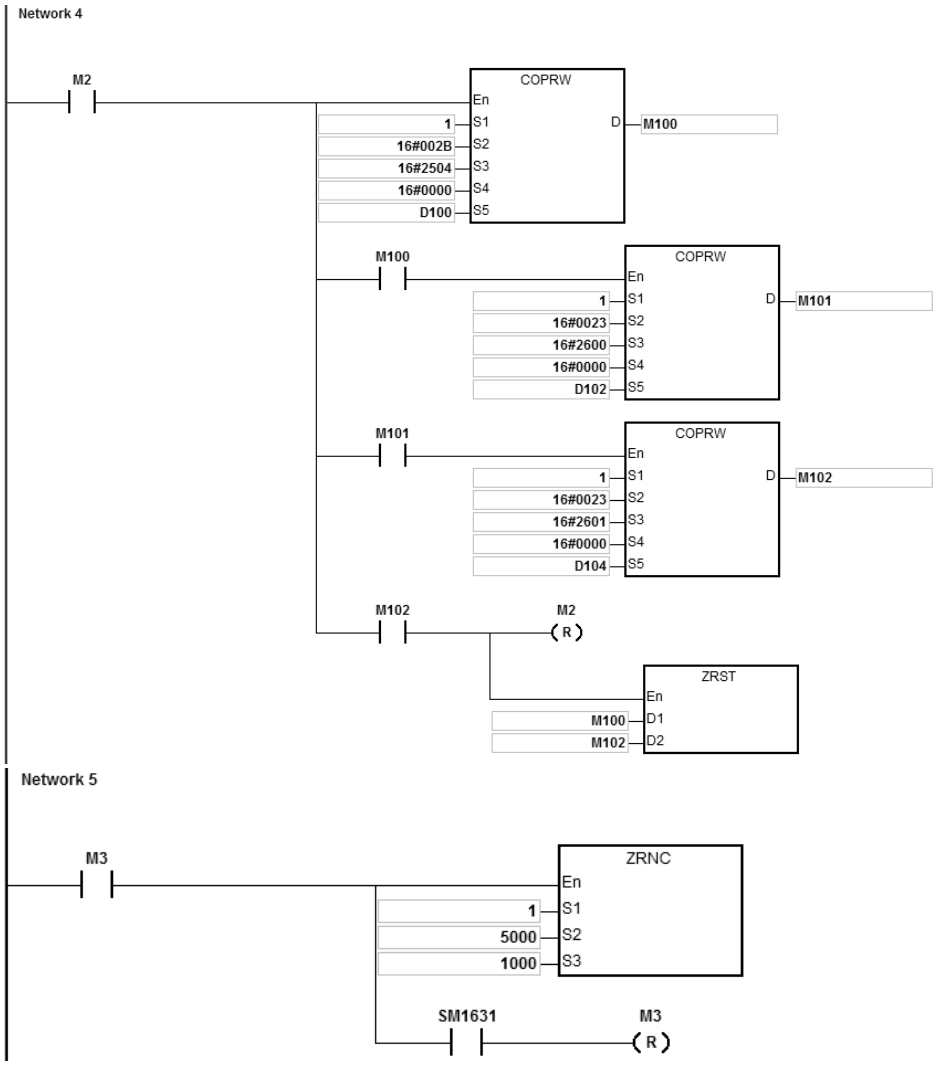
   4-byte data is written to P6-01 in servo 1. When D104 = 0 and the data is written, M102 is ON.

   P5-04 sets the homing mode.

   P6-00 and P6-01 are the homing definitions.

4. When M3 changes from OFF to ON, the homing function is enabled for servo 1.




Network 4
M2
COPRW
En
1
S1
D
M100
16#002B
S2
16#2504
S3
16#0000
S4
D100
S5
M100
COPRW
En
1
S1
D
M101
16#0023
S2
16#2600
S3
16#0000
S4
D102
S5
M101
COPRW
En
1
S1
D
M102
16#0023
S2
16#2601
S3
16#0000
S4
D104
S5
M102
M2
(R)
ZRST
En
M100
D1
M102
D2
Network 5
M3
ZRNC
En
1
S1
5000
S2
1000
S3
SM1631
M3
(R)

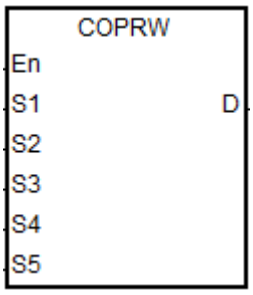
API	Instruction	Operand	Description
2807	COPRW	**S_1, S_2, S_3, S_4, S_5, D**	Reading and writing CANopen communication data

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●				○	○		
S_3								●	●				○	○		
S_4								●	●				○	○		
S_5								●								
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●			●	●							
S_3		●			●	●							
S_4		●			●	●							
S_5		●			●	●							
D	●												

Pulse instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



S_1 : Station address of servo or slave

S_2 : Request code

S_3 : Index

S_4 : Sub-index

S_5 : Read/write device

D : Communication completion flag

Explanation

1. This instruction reads and writes CANopen communication data to the servo at the address specified in **S_1**. The range of **S_1** is 1–127. If the value is out of range (<1 or >127), the minimum or maximum value is automatically processed by the instruction as the value of **S_1**.

2. **S_2** can only specify four types of request codes, as shown in the following table. If you set the request code wrong, it does NOT execute; SM1682 is ON and SR659 is 16#0005.

S_2 (only available for values of lower 8-bit)	Description
16#0023	Writing the 4-byte data; Expedited SDO
16#002B	Writing the 2-byte data; Expedited SDO
16#002F	Writing the 1-byte data; Expedited SDO
16#0040	Reading the data. The data length is contained in the SDO response message; Expedited SDO
16#0021	Data and its length are specified by S_5; Normal SDO
16#0041	Reading the data and store the read data and its length in S_5; Normal SDO

3. For **S_3** and **S_4**, refer to the object dictionary in the Delta servo / inverter operation manual or the object dictionary from a third-party device that complies with CANopen DS301.

4. You should execute the COPRW instruction only after the INITC instruction is complete in case the parameters are overwritten by the INITC instruction. Use the COPRW instruction for CANopen SDO communication. If the station address of a slave exceeds the range 1–8 or 21-28, you can use the COPRW instruction to set, read, and write slave-related parameters.

5. SM1682 is set to ON when an error occurs during communication. In addition, SR658 retains the number of the axis in which the error occurs and SR659 retains the error code. Refer to DRVIC instruction for details.

   Note: When you use the COPRW instruction, you must edit the process for dealing with communication errors in order to avoid invalid communication occurring as a result of unexpected communication errors.

6. The diagram below shows the timing of the COPRW instruction.

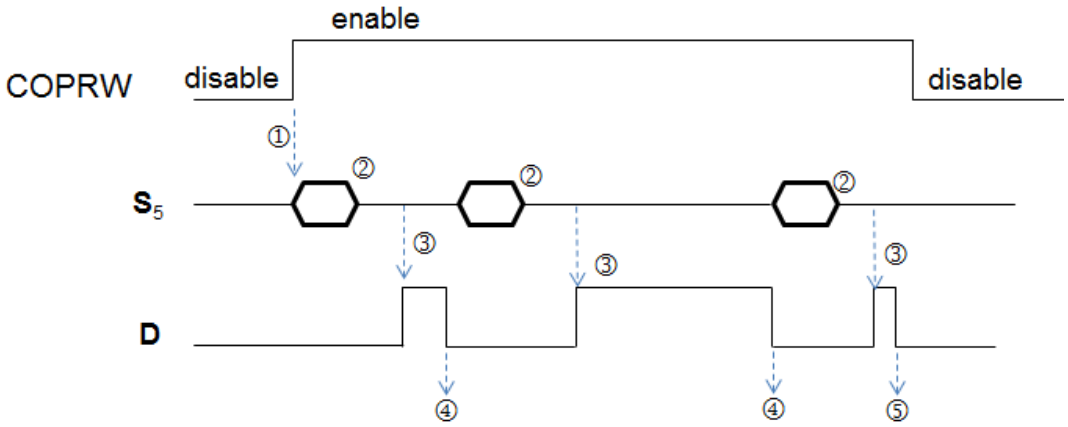
   ① When you enable the COPRW instruction for the first time, the instruction sends the command code immediately if no other CANopen communication is using it.

   ② The instruction sends the command code.

   ③ The code has been sent and the finish flag is set to ON.

   ④ You modify the next data to be sent out. The next command code is sent out immediately after the finish flag is set to OFF.

   ⑤The code has been sent and the COPRW instruction is disabled.



7. The COPRW instruction supports all CAN port work modes.

Parameter name	Value	Unit	Default	Minimum	Maxi
CAN port work mode	Delta Special Driver		Delta Special	-	-
CAN port node ID	Delta Special Driver			1	127
CANopen communication time out	CANopen DS301			0	3000
	Delta Special Driver & CANopen DS301 mode				

8. Most of the parameters in Delta ASDA-A2 are displayed in the decimal format. You can convert the parameters into index addresses, see the example below. 0 is a fixed value for the sub index address.

   Example: The index address of PX-YY=0x2000 +（X << 8）+ YY

   P2-10 = 0x2000 +（0x0002 << 8）+ 0x000A = 0x220A

   P5-04 = 0x2000 +（0x0005 << 8）+ 0x0004 = 0x2504

   P1-44 = 0x2000 +（0x0001 << 8）+ 0x002C = 0x212C

9. Most of the parameters in Delta inverter are also displayed in the decimal format. Use the following formula to convert the parameters.

   Example: The index address of PXX-YY=0x2000 + XX (hexadecimal);
   The sub index address is YY+1 (hexadecimal)

   The index address of P10-15 = 0x2000 + 0x000A = 0x200A
   The sub index address is 0x0F+1= 0x10

10. Since this instruction uses SDO (Service Data Objects) and the communication mode is one transmission (one sending/one receiving) at a time, it is not suggested to use 2 COPRW instructions at a time in one scan cycle. And it is not suggested to use this instruction in a function block. Because when this instruction is written in a function block, there will be 2 POU in the PLC program and when the function block is called, 2 instructions will be enabled at the same time.

11. When the request code is 16#0021, **S_5** is the data length (“n” byte) to write into the slave. **S_5**+1~ **S_5**+((n+1) / 2) is the data to write into the slave. For example, if **S_5** is specified as D100 and the value in **S_5** is 16. This instruction sends data in D101 ~ D108 to the slave in the following order, lower 8-bit in D101, upper 8-bit in D101, lower 8-bit in D102, upper 8-bit in D102 and so forth.

12. When the request code is 16#0041, when received responses from the slave, this instruction stores the data length (“n” byte) in **S_5** and data in **S_5**+1~ **S_5**+((n+1) / 2). For example, if received 15 bytes of data length, D100 is 15 and store the data Byte[0], Byte[1].. Byte[14] in the following order, lower 8-bit in D101, upper 8-bit in D101, lower 8-bit in D102, upper 8-bit in D102 and so forth.

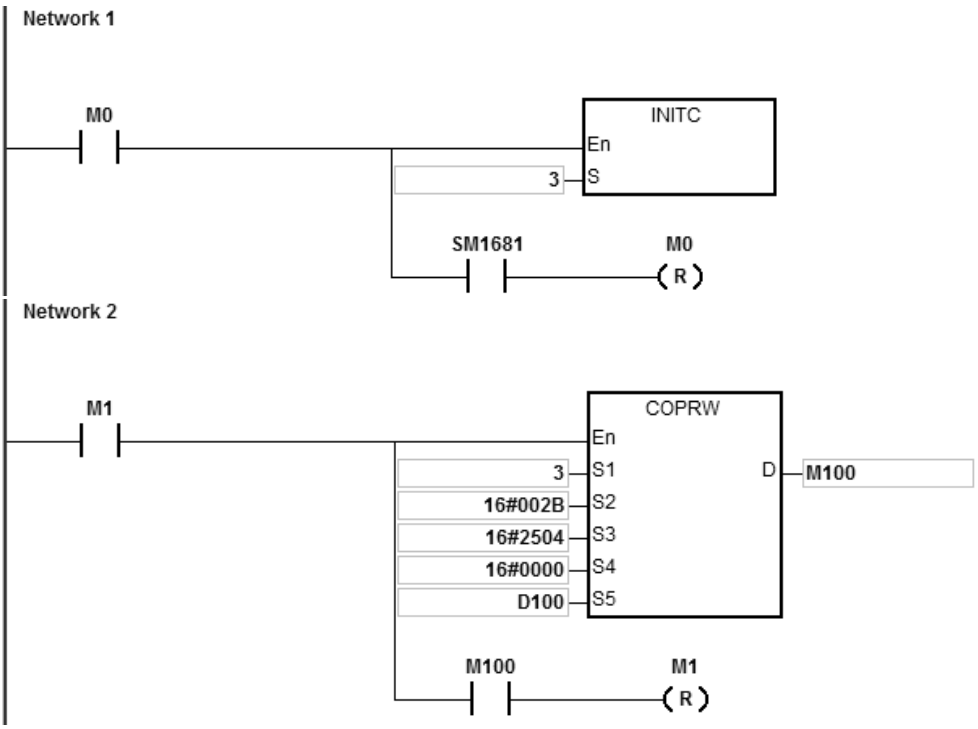
13. The maximum data length is 255 bytes. To prevent too many Normal SDO transmissions, if the data length exceeds 255 bytes, PLC only sends or receives 255 bytes.

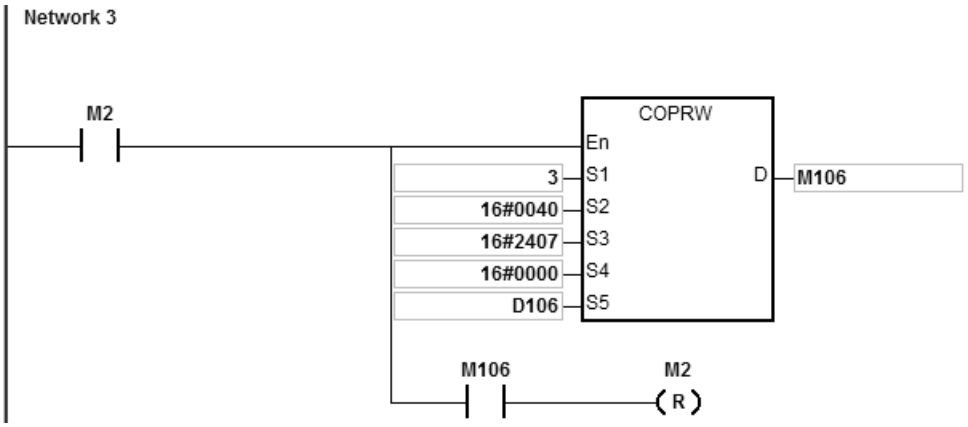
Example (using Expedited SDO)

1. When M0 changes from OFF to ON, the INITC instruction starts to initialize the servos at station addresses 1–3, until SM1681 is ON.

2. When M1 changes from OFF to ON, the PLC writes the 2-byte data (**S_2**=16#002B) in D100 to P5-04 (**S_3**=16#2504) of servo 3, using the COPRW instruction. When the writing is complete, M100 is ON.

3. When M2 changes from OFF to ON, the PLC reads the value of P4-07 (**S_3**=16#2407) of servo 3 and stores the value(**S_2**=16#0040) in D106. When the reading is complete, M106 is ON.




Network 3
M2
COPRW
En
3
S1
D
M106
16#0040
S2
16#2407
S3
16#0000
S4
D106
S5
M106
M2
R

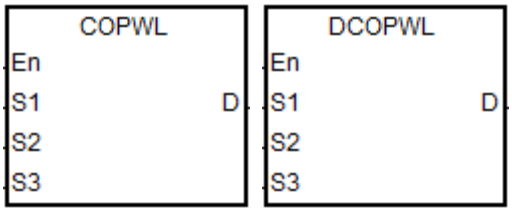
API	Instruction			Operand	Description
2808	D	COPWL		**S_1, S_2, S_3, D**	Writing multiple CANopen parameter values

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
S_1								●					○	○		
S_2								●	●							
S_3								●					○	○		
D		●	●	●				●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1		●			●	●							
S_2		●	●		●	●	●						
S_3		●			●	●							
D	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	ES3

Symbol



S_1 : Station address(Mac ID)

S_2 : Starting source device where written data are stored

S_3 : Number of messages to consecutively write data

D : Communication completion flag

Explanation

1. **S_1** sets the station address within the range of 1~127. If the setting value exceeds the range (< 1 or >127), the instruction will automatically send data at the minimum or maximum value respectively.

2. **S_2** is the starting source device where written data are stored and **S_3** is the number of messages to consecutively write data. E.g., **S_2** specifies D10 as the starting device and the number of messages to consecutively write data is 3. Here is the detailed explanation in the following table.

Instruction name	Message No.	Index address	Subindex address	Written source data
COPWL (Writes 16-bit values)	1	D10	D11	D12
	2	D13	D14	D15
	3	D16	D17	D18
DCOPWL (Writes 32-bit values)	1	D10	D11	D12, D13
	2	D14	D15	D16, D17
	3	D18	D19	D20, D21

The value of **S_3** is in the range of 1~100.

3. For the index addresses and subindex addresses of Delta servo and AC motor drive, refer to the explanation of the COPRW instruction. In principle, the parameter values of Delta servo and AC motor drive are both16-bit or 32-bit values including floating point numbers. If you need write an 8-bit value, use the COPRW instruction.

4. **D** is the communication completion flag. **D** will turn on after the sending of multiple communication messages is complete.

   See the detailed sending process and sequence diagram below.

   ① The COPWL instruction is enabled and starts to send data.

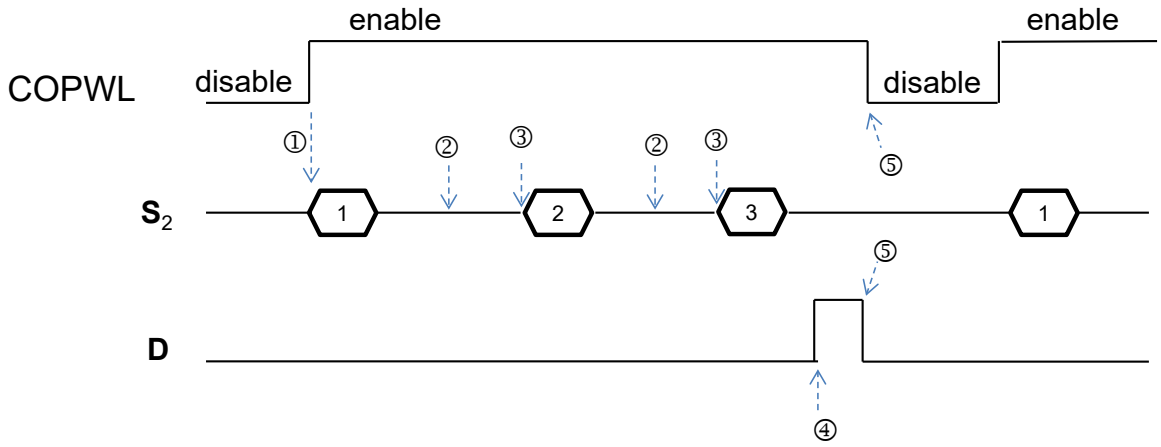
   ② After the COPWL instruction sends one piece of message, the next PLC instruction continues to execute.

   ③ As the COPWL instruction is scanned once again and the prior message has been received by the slave, the COPWL instruction sends the next message.

   ④ When the last written-data sending is done, the instruction will set the completion flag to ON.

   ⑤ When the completion flag turns on, the COPWL instruction need be disabled by manual so that the subsequent COPWL or COPRW instruction can continue to work.

   Note: When you disable the instruction, the completion flag will be automatically cleared accordingly.



   Note: The sequence diagram above shows the sending of 3 pieces of written data.

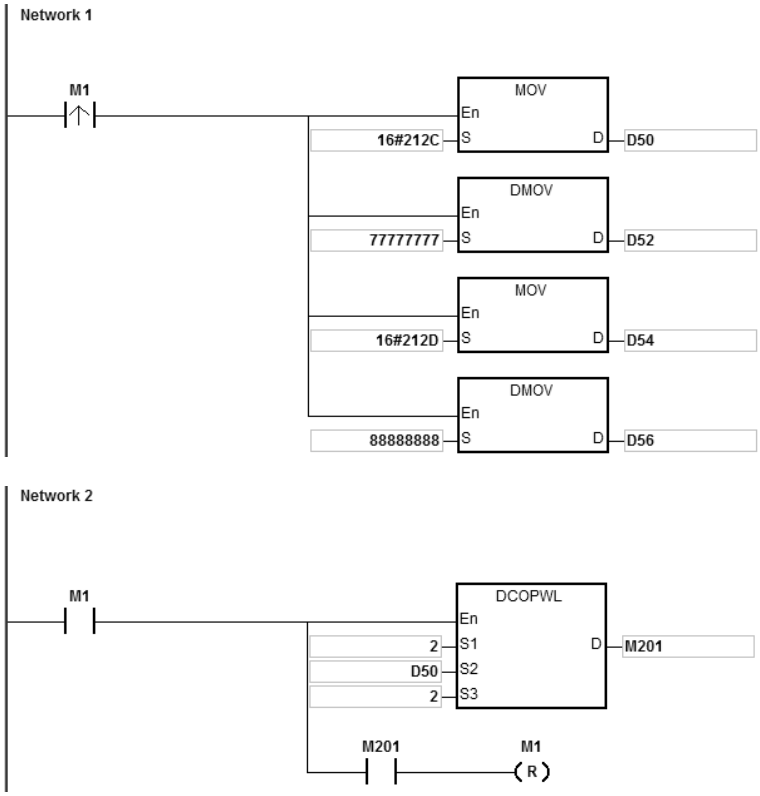
5. After the instruction is enabled, wait until the writing is complete and then disable the instruction. If there is a communication error in the execution, shoot the trouble and then re-enable the instruction to write all data.

Example

1. When M1 = OFF → ON, data are written in D device.

Instruction name	Data No.	Index address	Subindex address	Written data source
DCOPWL (Writes 32-bit values)	1	D50 = 16#212C (E-gear ratio numerator)	D51 = 0	D52, D53 = 77777777
	2	D54 = 16#212D (E-gear ratio denominator)	D55 = 0	D56, D57 = 88888888

2. When M1 = OFF → ON, the instruction writes a 32-bit value for P1-44 of the servo whose station address is 2 and the written value 77777777 is stored in D52. The instruction writes a 32-bit value for P1-45 and the written value 88888888 is stored in D56. As the writing is complete, M201 turns on.



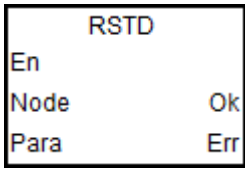
API	Instruction	Operand	Description
2809	RSTD	**Node，Para，Ok，Err**	Sending Reset or NMT command

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Node								●					○	○		
Para								●					○	○		
Ok		●	●	●												
Err		●	●	●												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Node		●											
Para		●											
Ok	●												
Err	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



Node : Station address which is reset

Para : Setting value of the parameter

Ok : The reset completion flag

Err : The reset error flag

Explanation

1. Before the RSTD instruction is used in Delta special instruction mode, make sure that all Delta drives have been initialized via the INITC instruction and they once worked normally.

2. When used in CANopen DS301 mode, the RSTD instruction works as the NMT communication function and can switch network states via the **Para** parameter.

3. When CAN communication port is specified to work in Delta special driver mode, the value of **Node** can be 0 (for the broadcast function) and 1~8 which are for servo station addresses only and 21~28 for station addresses of AC motor drives only. When the station address exceeds the range, the PLC will not perform the reset action and the **Err** flag turns on. (Refer to the explanation of SR659 for error codes)

4. When CAN communication port is specified to work in CANopen DS301 mode, the value of **Node** is in the range of 1~64 and 0 (for the broadcast function). When the value exceeds the range, the PLC will not perform the NMT communication and the **Err** flag turns on. (Refer to the explanation of SR659 for error codes)

5. The setting value of **Para** is NOT applicable to Delta special driver mode. The settings for **Para** (NMT service code) are listed in the following table. If the setting value is not one of the values in the table, the **Err** flag turns on.

NMT service code	16#01	16#02	16#80	16#81	16#82
Function description	Start the slave	Stop the slave	Enter the pre-operational state	Reset the application layer	Reset the communication

6. The RSTD instruction can implement the command action on only one drive or slave every time. If multiple RSTD instructions are enabled simultaneously, the PLC will automatically take priority to perform the instruction which is enabled earlier.

7. The RSTD instruction is executed to send the command when it is enabled. If the instruction is disabled before the **Ok** flag is on, the PLC will not set the **Ok** flag to ON.

8. Apart from notifying the specified drive to clear the error state, the instruction would also re-check if relevant communication parameter values are correct and re-set correct communication parameter values.

   For example, due to the disconnection of the slave of station address 2, the entire system stops running. After the trouble is solved, the slave of station address 2 can return to the state of being controllable by using the RSTD instruction to reset the slave of station address 2 only. So the time of re-initializing all drives are saved.

9. If the slave responds by sending back any communication command fault to the PLC during the communication, the RSTD instruction will turn the **Err** flag on and stop the upcoming actions. (Refer to explanation of SR659 for error codes.)

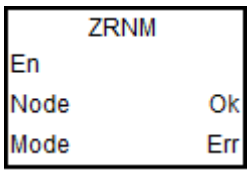
API	Instruction	Operand	Description
2810	ZRNM	**Node，Mode，Ok，Err**	Setting the homing mode for Delta servo

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Node								●					○	○		
Mode								●					○	○		
Ok		●	●	●												
Err		●	●	●												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Node		●											
Mode		●											
Ok	●												
Err	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



Node : Specified node ID

Mode : Homing mode code

Ok : Completion flag

Err : Error flag

Explanation

1. Make sure that all Delta drives have been initialized before executing the reset instruction.

2. The value of **Node** is in the range of 1~8 (exclusive to servo node IDs). If the setting value exceeds the range, the PLC will not perform the action of the homing mode and set the **Err** flag to ON. (Refer to explanation of SR659 for error codes.)

3. The ZRNM instruction can set the homing mode of only one drive every time. If multiple instructions are enabled simultaneously, the PLC will take priority to perform the instruction which is enabled earlier.

4. The ZRNM instruction is executed to send the command when it is enabled. If the instruction is disabled before the **Ok** flag is on, the PLC will not set the **Ok** flag to ON.

5. **Mode** sets a homing mode. If the setting value exceeds the range, the PLC will still send the command and the server itself will decide whether to receive the command or not. The setting mode is the homing mode that ASDA servo parameter P5-04 corresponds to.

The setting value of Delta servo homing mode is a hex value. The value is defined as the format of 0xWZYX. See the explanation of respective codes as below.

Homing mode code	Range	Function and code description	Remark
W	0 ~ 1	Select the final position where the servo stops. 0 = The servo leaves the original point, decelerates and stops and then automatically returns to the real original point. 1 = After leaving the original point, decelerating and stopping, the servo will not perform any action any more.	
Z	0 ~ 1	Handling mechanism when the limit is encountered. 0 = Output stops. 1 = Output is conducted in the reverse direction.	
Y	0 ~ 2	Z pulse signal setting (used for X code 0~8) 0 =Look for Z pulse when coming back. Do not look for Z phase when going forward. 1 =Go forward to Z pulse. Do not look for Z pulse when coming back. 2 =Do not look for Z pulse. Z pulse signal handling method (applicable to X code: 9~A) 0 =Look for Z pulse when coming back. 1 =Do not look for Z pulse both when coming back and going forward.	
X	0 ~ A	Homing method: 0~8 0 = Homing in the forward direction; **PL** is the original point 1 = Homing in the reverse direction; **NL** is the original point. 2 = Homing in the forward direction; **ORGP:** OFF > ON, as the original point. 3 = Homing in the reverse direction; **ORGP:** OFF > ON, as the original point. 4 = Homing in the forward direction; look for Z pulse and regard it as the original point. 5 = Homing in the reverse direction; look for Z pulse and regard it as the original point. 6 = Homing in the forward direction; **ORGP:** ON > OFF, as the	

		original point. 7 = Homing in the reverse direction; **ORGP:** ON > OFF, as the original point. 8 = Current position is the original point. Homing method: 9~A 9 = Homing in the forward direction; the collision point is the original point. A = Homing in the reverse direction; the collision point is the original point.	

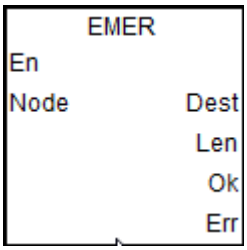
API	Instruction	Operand	Description
2811	EMER	**Node，Dest，Len，Ok，Err**	Reading Emergency message

Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	“$”	F
Node								●					○	○		
Dest								●								
Len								●								
Ok		●	●	●												
Err		●	●	●												

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Node		●											
Dest		●											
Len		●											
Ok	●												
Err	●												

Pulse Instruction	16-bit instruction	32-bit instruction
—	ES3	—

Symbol



Node : Specified node ID

Dest : Target device for storing data

Len : Total number of 4 words of data which have been read

Ok : Completion flag

Err : Error flag

Explanation

1. The EMER instruction is applicable to CANOpen DS301 mode and Delta special mode.
2. After receiving the Emergency message from the slave **Node**, the PLC will automatically store the data in the specified storage device and set the **Ok** flag to ON.
3. It is recommended that the **Node** value should be specified from the slave node IDs which have already existed. If the value is not one existing node ID or the slave has been disconnected, the PLC will not be able to receive any message, set the **Err** flag to ON and show error code of communication timeout. (Refer to explanation of SR659 for error codes.)
4. The way the EMER instruction reads Emergency messages is the same as Emergency communication method in ES3 operation manual. Please just select one communication method from them when reading Emergency

messages.

5. The EMER instruction can read 5 Emergency messages at most. Every time the reading is successful, the **Ok** flag turns on and **Len** displays the total number of messages which are read. You can judge how many consecutive words are occupied by **Dest** based on the length. Every message uses 4 words. The data are stored in the order from lower 8 bits to higher 8 bits. The storage format is shown as below. (E.g. **Dest** is D10, **Len** is 2 which is the number of messages stored in D5.)

D device no.	Value
D5	2

D device no.	Higher 8 bits	Lower 8 bits
D10	The second byte in the first message	The first byte in the first message
D11	The forth byte in the first message	The third byte in the first message
D12	The sixth byte in the first message	The fifth byte in the first message
D13	The eighth byte in the first message	The seventh byte in the first message
D14	The second byte in the second message	The first byte in the second message
D15	The forth byte in the second message	The third byte in the second message
D16	The sixth byte in the second message	The fifth byte in the second message
D17	The eighth byte in the second message	The seventh byte in the second message

API	Instruction			Operand	Description
2812	D	CSFOC		**Xno ~ OutSpd**	Controlling the tracking function of a servo via communication

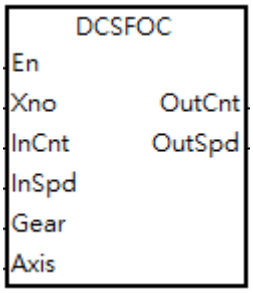
Device	X	Y	M	S	T	C	HC	D	FR	SM	SR	E	K	16#	"$"	F
Xno	○															
InCnt								●								
InSpd								●								
Gear								●								●
Axis								●					●	●		
OutCnt								●								
OutSpd								●								

Data type	BOOL	WORD	DWORD	LWORD	UINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
Xno	●												
InCnt			●				●						
InSpd			●				●						
Gear									●				
Axis		●											
OutCnt			●				●						
OutSpd			●				●						

Pulse Instruction	16-bit instruction	32-bit instruction
—	—	ES3

6

Symbol



Xno : Signal input device

InCnt : Number of input pulses

InSpd : Pulse input frequency

Gear : Proportion of the input frequency and output frequency (floating-point value)

Axis : CANopen communication station address of Delta servo

OutCnt Number of output pulses (unit: per user unit (PUU))

OutSpd Pulse output frequency

Explanation

1. The INITC and ASDON (Servo-ON) instructions must be complete before this instruction is executed.

2. **Xno** can only specify X0, X2, X4, X6, X10 and X12 as the input points, and the operand occupies two consecutive input points. You cannot execute the instruction if the input points are the points specified for **Xno** above. After you select the input points, the high-speed counter is automatically specified. If there is a DCNT instruction (API 1004) or CSFO instruction (API 2708) with the same high-speed counter in the program, the PLC first executes the instruction that starts the counter first. The input points and corresponding high-speed counters are shown in the following table.

Group number	1	2	3	4	5	6
Xno+0 input point (Phase A)	X0	X2	X4	X6	X10	X12
Xno+1 input point (Phase B)	X1	X3	X5	X7	X11	X13
High-speed counter number	HC202	HC206	HC210	HC214	HC218	HC222
Flag for reversing input direction	SM270	SM271	SM272	SM273	SM274	SM275

Note: refer to instruction DCNT for more information on the maximum input bandwidth of X0~X13.

3. If the high-speed counters for the instruction can use only the phase A/B input mode, set the flag for reversing the input direction to ON when MPG is connected but has not rotated yet, and the PLC input point is ON. Set the function to detect the state of PLC run cycle to OFF.

4. **InCnt** is the number of input pulses. Use a 32-bit variable to declare the parameter.

5. **InSpd** is the frequency of input pulses. Use a 32-bit variable to declare the parameter with the unit of 1 Hz.

6. **Gear** is the proportion of the input frequency and output frequency (floating-point value). The formula is that output frequency equals to the result of input frequency multiplying Gear. For instance, if Gear is 2.5 and the input frequency is 50, the output is 125 (50*2.5). Note: the output count and the output frequency are rounded down to the nearest whole digit when outputting.

7. **Axis** is the station address of Delta servo for output, ranging from 1 to 8 and it should be within the station address that is initiated by the INITC instruction.

8. **OutCnt** is the number of output pulses. Use a 32-bit variable to declare the parameter with the unit of PUU.

9. **OutSpd** is the frequency of the output pulses. Use a 32-bit variable to declare the parameter with the unit of 1 Hz.

10. There is no limit to the number of times you can use this instruction. Several instructions can use one group of input points as its source of input count. But you need to clear the value in the input counter before execution of this instruction; especially when you are using one input and several outputs. And this instruction executes according to the axis number to record information about its input and output. The same axis can be used for several times but you must not output the same axis at the same time. If you use the same axis to output for several times, PLC outputs the first starting one and then the second one.

11. The instruction cannot be used in the ST programming language, interrupt tasks or function block which is called only once.

Note

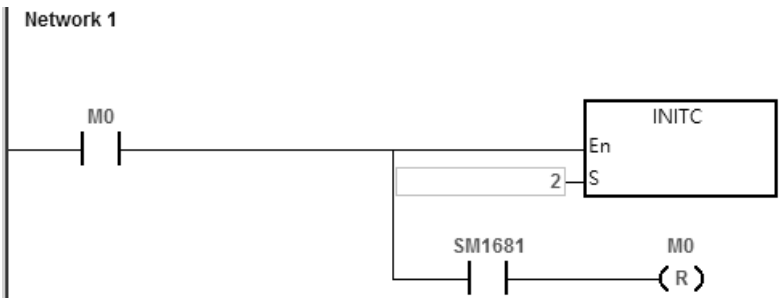
1. The PLC calculates the input pulse frequency based on the input pulse width (ON) in the positive half cycle. If the proportion of the pulse width for ON:pulse width for OFF is not 1:1, the PLC takes the ON width as the standard for conversion by default.

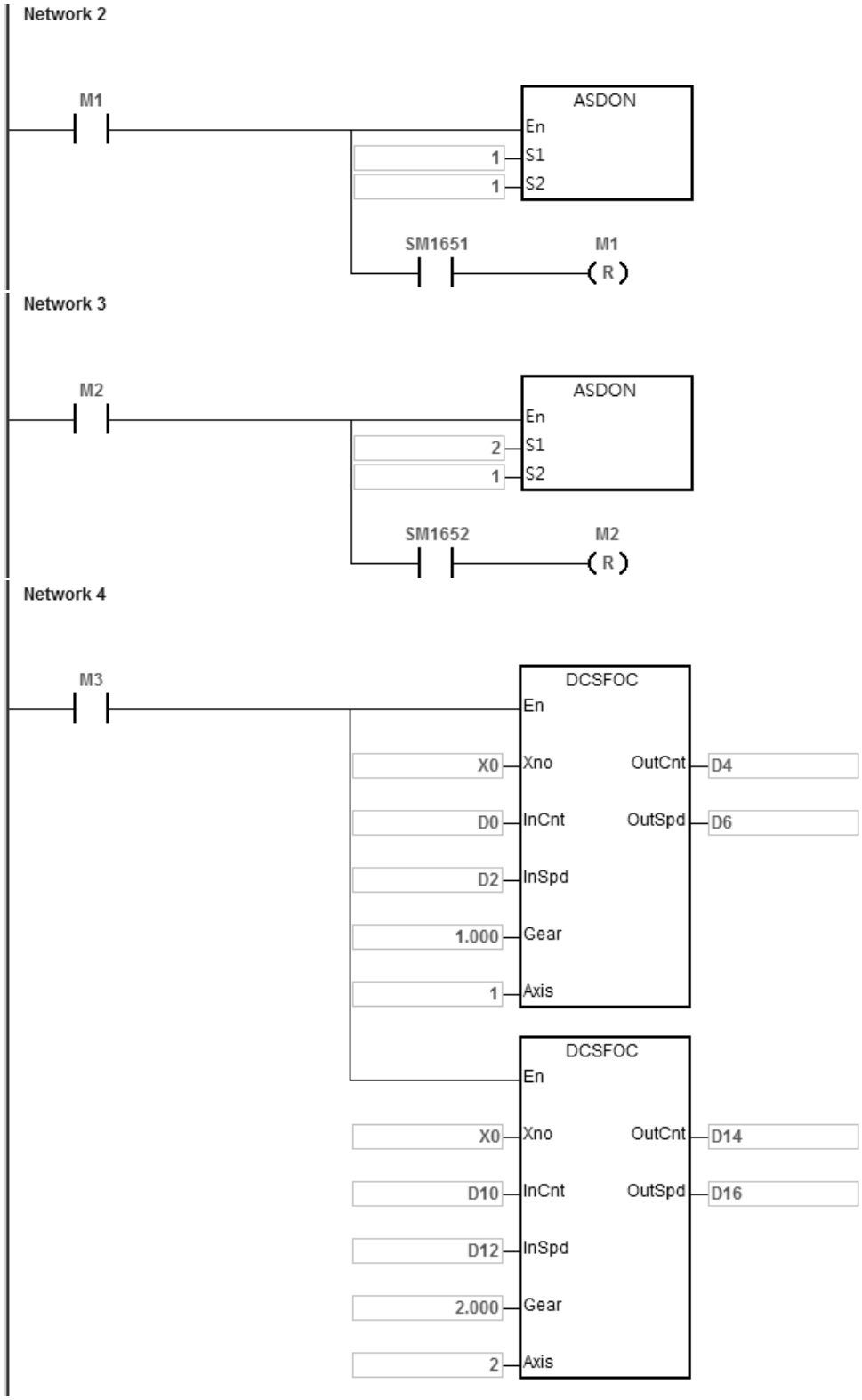
2. The input pulse=ON means the input point LED is on. When using the MPG but it has not rotated yet, you can check if the input point LED is OFF and the flag for reversing the input direction is activated .

3. There is a 20 ms time difference in CANopen communication when executing this instruction. If this worries you, use CSFO instruction instead, controlling the servo by the pulse output.

Example

- When M0 switches from OFF to ON, PLC starts to execute this instruction to initialize servo station address 1 and 2 until SM1681 is ON.
- When M1 switches from OFF to ON, PLC starts to execute this instruction to start servo station address 1 and when the servo is started, SM1651 is ON, meaning Servo-ON.
- When M2 switches from OFF to ON, PLC starts to execute this instruction to start servo station address 1 and when the servo is started, SM1652 is ON, meaning Servo-ON.
- When M3 switches from OFF to ON, input points X0 and X1 connect to the MPG to control 2 servos. The proportions for the 1st servo is 1 and for the 2nd servo is 2.




Network 2
M1
ASDON
En
1
S1
1
S2
SM1651
M1
R
Network 3
M2
ASDON
En
2
S1
1
S2
SM1652
M2
R
Network 4
M3
DCSFOC
En
X0
Xno
OutCnt
D4
D0
InCnt
OutSpd
D6
D2
InSpd
1.000
Gear
1
Axis
DCSFOC
En
X0
Xno
OutCnt
D14
D10
InCnt
OutSpd
D16
D12
InSpd
2.000
Gear
2
Axis

6.27.3 Frequently asked questions in Delta special CANopen communication and Troubleshooting

Question 1: After the PLC starts the INITC instruction, the initialization completion flag SM1681 is not set to ON.

Steps:

1) Make sure that CANopen communication mode is Delta special driver or Delta special driver & CANopen DS301 mode. Then check if the communication rate is the specified baud rate. After above settings are correct, re-download the hardware configuration table and then re-check if the initialization works. If the initialization still does not work, proceed with the next step.
2) Ensure that the servo initialization has been set by manual. Then re-power the servo on after power off. After doing so, continue with step 3.
3) Ensure that the servo node ID (P3-00), baud rate and mode (P3-01) have been set by manual and communication mode (P1-01) has been set to PR mode by manual. Then proceed with the step 4.
4) Ensure that the CAN communication cable connection is proper, the terminal resistor in ES3 Series PLC is switched on and the last servo has been connected with a terminal resistor. Then move forward to the step 5.
5) Observe whether the word: **code** appears temporarily on the digital display of the servo when the PLC enables the initialization instruction. If there is no code appearing temporarily, return to step 1 or replace the servo. If there is the word **code** appearing temporarily, go to step 6.
6) Check if the actual states of DI0~DI7 (P2-10 ~ P2-17) of the servo are consistent with the function settings of servo parameters. E.g. DI0 is the negative limit input (contact A). Check if the actual input state has been OFF (Contact B). If DI function has been enabled, make the function disabled. If the function has not been enabled, go on with step7.
7) The communication mode is switched back to CANopen DS301 mode. Use the CANopen Builder software to search the slave and observe which slave has not been scanned. Remove the one which has not been scanned and then make the initialization again by adjusting the servo node ID and the ID in INITC in the PLC.

7

Chapter 7 Troubleshooting

Table of Contents

7.1 Troubleshooting

7.1.1 Basic troubleshooting steps

This chapter includes the possible errors the can occur during operation, their causes, and corrective actions.

(1) Check the following:

- The PLC should be operated in a safe environment (consider environmental, electronic, and vibration safeties).
- Connect power supply correctly to the PLC.
- Secure the module, terminal, and cable installations.
- All LED indicators show correctly.
- Set all switches correctly.

(2) Check the following operational functions:

- Switch the RUN/STOP state
- Check the settings for the DVP-ES3 Series to RUN/STOP
- Check and eliminate errors from external devices
- Use the System Log function in ISPSoft to check system operation and logs

(3) Identify possible causes:

- DVP-ES3 Series or external device
- CPU or extension modules
- Parameters or program settings

7.1.2 Clear the Error States

Use the following methods to clear the error states. If the error source is not corrected, the system continues to show errors.

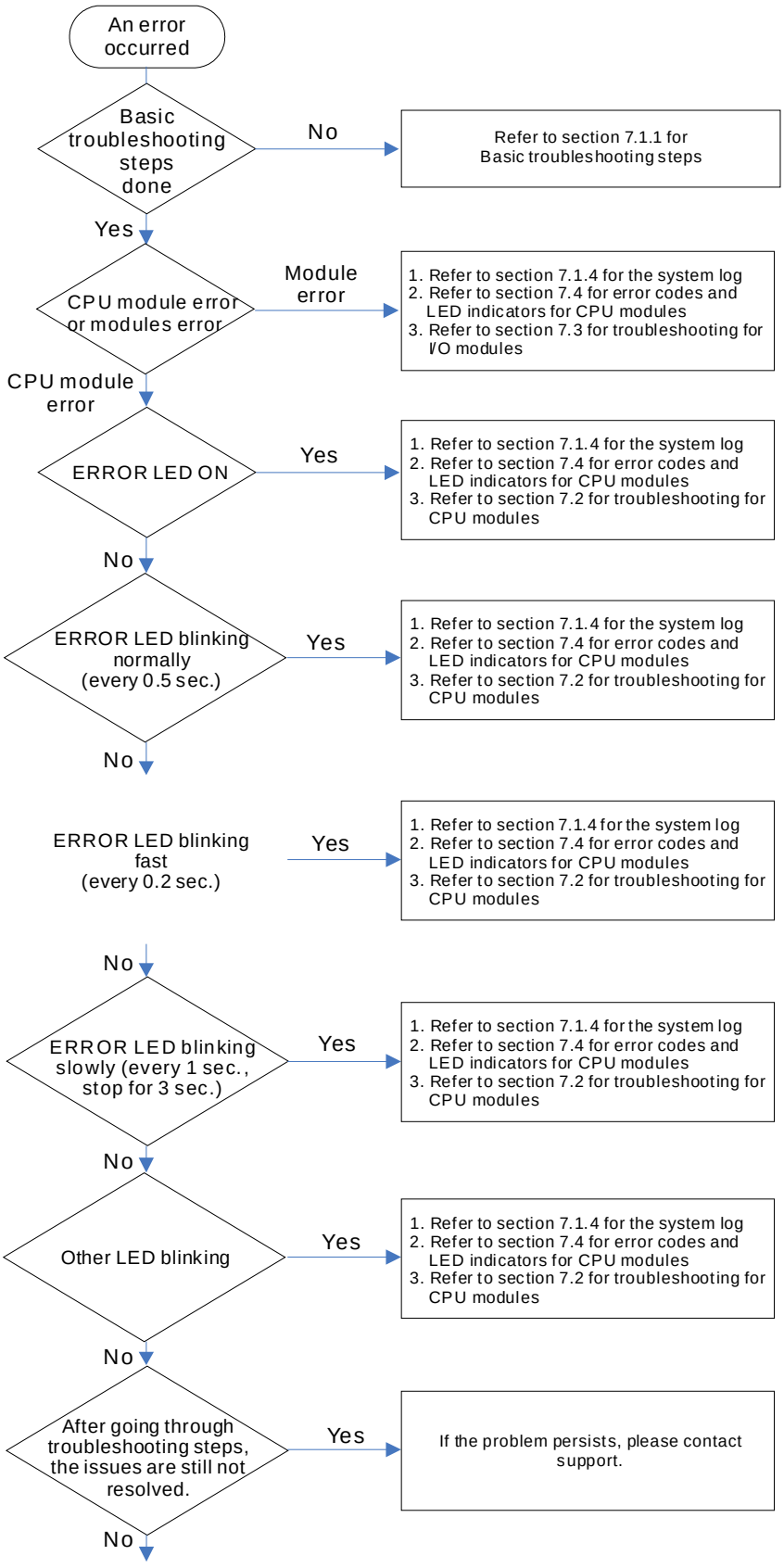
(1) Switch the CPU model state to STOP and then to RUN.

(2) Turn off the CPU and turn it on again.

(3) Use ISPSoft to clear the error logs.

(4) Reset the CPU to the default settings and download the program again.

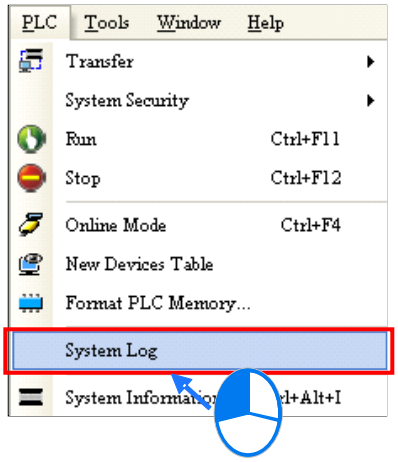
7.1.3 Troubleshooting SOP



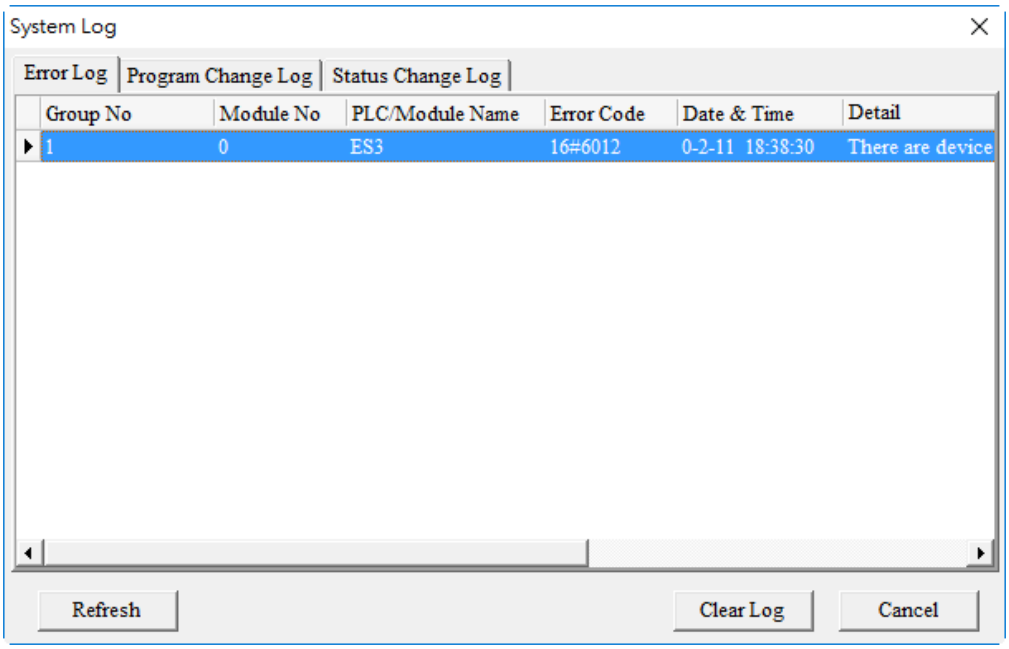
7.1.4 System Log

If ISPSoft is connected to a DVP-ES3 Series, you can view actions and errors in the DVP-ES3 Series by clicking **System Log** on the **PLC** menu. The CPU can store up to 20 error log sets. After the 20 sets are stored, the 1st log is replaced with the 21st if there are new logs coming in, and the old logs are replaced with the new ones sequentially. When the memory card is installed in the CPU module, 20 sets of the old logs are backed up in the memory card and up to 1000 logs can be recorded. If the stored logs exceed 1000, the oldest 20 logs are replaced with the newest 20 logs in the memory card.

(1) On the **PLC** menu, click **System Log**.



(2) The **System Log** window appears. Click **Clear Log** to clear the error log in the window and the error log in the CPU module, and reset the CPU module.



- Group No.: The number 1 indicates that the error occurred in the CPU module or the right-side module 1.
- Module No.: The number 0 indicates that the error occurred in the CPU module or the remote module. T
- PLC/Module name: Model names of the CPU, remote, and extension modules.
- Error Code: Error codes in the error log.
- Date & Time: The date and time the error occurred. The most recently occurring error is listed on the top.
- The last column shows the descriptions for the error.

7.2 Troubleshooting for CPU Modules

Check the LED indicators and the error codes from the CPU module and refer to the following table for troubleshooting. V in the Log column indicates the error is recorded in the log. X in the Log column indicates the error is not recorded in the log. H in the Log column indicates whether or not you can set recording the error in the log in HWCONFIG.

7.2.1 ERROR LED Indicators Are ON

Error Code (16#)	Description	Solution	Flag	Log
000A	Scan timeout	1. Check the setting of the watchdog timer in HWCONFIG. 2. Check whether the program causes a long scan time	SM8	V

7.2.2 ERROR LED Indicators Blinking Every 0.5 Seconds

Error Code (16#)	Description	Solution	Flag	Log
000C	The program in the PLC is damaged.	Download the program again.	SM9	V
0010	CPU memory is denied.	Contact the factory.	SM9	V
002E	CPU external memory access is denied.	Contact the factory.	SM9	V
002F	PLC programs are not consistent with the system logs.	Download the program again.	SM34	V
0102	The interrupt number exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0202	The MC instruction exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0302	The MCR instruction exceeds the range.	Check the program, compile the program again, and download the program again.	SM5	X
0D03	The operands used in DHSCS are not used properly.	Check the program, compile the program again, and download the program again.	SM5	X
0E05	The operands HCXXX used in DCNT are not used properly.	Check the program, compile the program again, and download the program again.	SM5	X
200A	Invalid instruction	Check the program, compile the program again, and download the program again.	SM5	V
6010	The number of MODBUS TCP connections exceeds the range.	Check the number of superior devices (maximum is 32).	SM 1092	V
6011	The number of EtherNet/IP connections exceeds the range.	Check the number of connections (maximum is 16).	SM 1093	V
C000 - CFFF	The program syntax is incorrect.	Save the PLC program and hand the file to the company or the technicians.		

7.2.3 ERROR LED Indicators Blinking Rapidly Every 0.2 Seconds

This happens when the power supply 24 VDC of the CPU module is disconnected, or the power supply is not sufficient, not stable or abnormal.

Error Code (16#)	Description	Solution	Flag	Log
002A	The external voltage is abnormal.	Check whether the external 24 V power supply to the module is normal.	SM7	V

7.2.4 ERROR LED Indicators Slow Blinking Every 3 Seconds and Lighting up for 1 Second

Error Code (16#)	Description	Solution	Flag	Log
1900 - 191C	Heartbeat errors occurred in the slave for the Delta ASD-A2 control.	1. Check the CANopen connection cable. 2. Check if the specific slave is working properly. Note: The last 2 digits of the error code represent the ID number of the slave (convert hexadecimal to decimal).	-	V

7.2.5 The LED RUN and ERROR Indicators are Blinking Simultaneously Every 0.5 Seconds

This happens when the firmware of the CPU module is being upgraded. If this happens once the power is supplied to the CPU module, it means errors occurred during the previous firmware upgrade. Users need to upgrade the firmware again or contact your point of purchase.

7.2.6 The RUN and LED Indicators are Blinking One After Another Every 0.5 Seconds.

This happens when the CPU module memory card is backing up, restoring, or saving.

7.2.7 Other Errors (Without LED Indicators)

Error Code (16#)	Description	Solution	Flag	Log
0011	The PLC ID is incorrect.	Check the PLC ID.	SM34	V
0012	The PLC password is incorrect.	Check the PLC password.	SM34	V
002D	The PLC maximum password attempts exceeded.	Reset the CPU module or restore the CPU module to its factory settings.	SM34	V
0050	The memories in the latched special auxiliary relays are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
0051	The latched special data registers are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again.	SM6	V

		2. If the error still occurs, contact the factory.		
0052	The memories in the latched auxiliary relays are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
0054	The latched counters are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, please contact the factory.	SM6	V
0055	The latched 32-bit counters are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
0056	The latched special auxiliary relay is abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
0059	The latched data registers are abnormal.	1. Reset the CPU module or restore the CPU module to its factory settings, and then download the program and the parameters again. 2. If the error still occurs, contact the factory.	SM6	V
005D	The CPU module does not detect a memory card.	Check that the memory card is inserted correctly into the CPU module.	SM453	V
005E	The memory card is initialized incorrectly.	Check whether the memory card is broken.	SM453	V
0063	An error occurs when data is written to the memory card.	Check whether the file path is correct or whether the memory card is malfunctioning.	SM453	V
0064	A file in the memory card cannot be read.	Check whether the file path is correct, or whether the file is damaged.	SM453	V
1950	The initialization of the Delta ASD-A2 control has not yet been completed, the CANopen instructions cannot be executed.	1. Check the CANopen connection cable. 2. Check if the specific slave is working properly. 3. If nothing is wrong, initialize the Delta ASD-A2 again.	-	V
2001	Not in the right mode for the ASDA-A2 while using the CANopen communication instruction.	Check if the CANopen operation mode is correct.	SM0	V
2003	The device used in the program exceeds the device range.	Check the program, compile the program again, and download the program again.	SM0	V
200B	The operand n or the other constant operands K/H exceed the range.	Check the program, compile the program again, and download the program again.	SM0	V
200C	The operands overlap.	Check the program, compile the program again, and download the program again.	SM0	V
200D	The binary to binary-coded decimal conversion is incorrect.	Check the program, compile the program again, and download the program again.	SM0	V
200E	The string does not end with 00.	Check the program, compile the program again, and download the program again.	SM0	V
2012	Incorrect division operation	Check the program, compile the program again, and download the program again.	SM0	V
2013	The value exceeds the range of values that can be represented by the	Check the program, compile the program again, and download the program again.	SM0	V

	floating-point numbers.			
2014	The task designated by the TKON or YKOFF instruction is incorrect or exceeds the range.	Check the program, compile the program again, and download the program again.	SM0	V
2017	The instruction BREAK is written outside of the FOR-NEXT loop.	Check the program, compile the program again, and download the program again.	SM0	V
2027	No such position planning table number or the format is incorrect.	1. Check the program, compile the program again, and download the program again. 2. Check the settings of the position planning table.	SM0	V
2028	High speed output instruction is being executed. Only one instruction can be executed at a time.	Refer to SR28 for the record of the axis number and rearrange the output control procedures.	-	V
6004	The IP address filter is set incorrectly.	Set the Ethernet parameter for the CPU module in HWCONFIG again.	SM1108	X
600D	RJ45 port is not connected.	Check the connection.	SM1100	X
6012	There are devices using the same IP address.	1. Check if there are devices using the same IP address. 2. Check if there is more than 1 DHCP or BOOTP server on the network.	SM1101	V
6100	The email connection is busy.	Retry the email connection later. This error does not cause the PLC to stop running. Solve the problem by means of the related flag in the program.	SM1113	X
6103	The trigger attachment mode in the email is set incorrectly.	Set up the trigger attachment mode in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email > Trigger Setting > Trigger Attachment Mode.	SM1113	X
6104	The attachment in the email does not exist.	Check whether the attachment exists in the memory card.	SM1113	X
6105	The attachment in the email is oversized.	Check the size of the attachment. If the size is over 2 MB, the file cannot be sent as an attachment.	SM1113	X
6106	There is an SMTP server response timeout.	Check for the correct address and set up the SMTP server in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email again.	SM1113	X
6107	There is an SMTP server response timeout.	1. Check whether the status of the SMTP server is normal. 2. Retry sending of the email later. This error does not cause the PLC to stop running. Solve the problem by means of the related flag in the program.	SM1113	X
6108	SMTP verification failed	Check for the correct ID/Password and set up in HWCONFIG > CPU Module > Device Setting > Options > Ethernet Port Advanced > Email again.	SM1113	X
6200	The remote communication IP address set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6201	The local communication port set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module	-	X

		in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.		
6202	The remote communication port set in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6203	The device from which the data is sent in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6206	The device which receives the data in the TCP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6208	The data received through the TCP socket exceeds the device range.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > TCP Socket.	-	X
6209	The remote communication IP address set in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620A	The local communication port set in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620C	The device from which the data is sent in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
620F	The device which receives the data in the UDP socket function is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X

6210	The data received through the UDP socket exceeds the device range.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
6212	There is no response from the remote device after the timeout period.	Make sure that the remote device is connected.	-	X
6213	The data received exceeds the limit.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG CPU Module > Device Setting > Options > Ethernet Port Advanced > UDP Socket.	-	X
6214	The remote device refuses the connection.	Make sure the remote device operates normally.	-	X
6215	The socket is not opened.	Check whether operational sequence in the program is correct.	-	X
6217	The socket is opened.	Check whether operational sequence in the program is correct.	-	X
6218	The data has been sent through the socket.	Check whether operational sequence in the program is correct.	-	X
6219	The data has been received through the socket.	Check whether operational sequence in the program is correct.	-	X
621A	The socket is closed.	Check whether operational sequence in the program is correct.	-	X
7011	The device communication function code in COM1 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7012	The device communication address used in COM1 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7013	The device used in COM1 exceeds the device range.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7014	The device length of the communication data in COM1 exceeds the limit.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7017	The device checksum for the communication serial port of COM1 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7021	The device communication function code in COM2 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7022	The device communication address used in COM2 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7023	The device used in COM2 exceeds the device range.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H

7024	The device length of the communication data in COM2 exceeds the limit.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7027	The device checksum for the communication serial port of COM2 is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7031	The device communication function code in the Ethernet is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7032	The device communication address used in the Ethernet is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7033	The device used in the Ethernet exceeds the device range.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7034	The device length of the communication data in the Ethernet exceeds the limit.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7037	The device checksum for the communication serial port of the Ethernet is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7041	The device communication function code in the USB is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7042	The device communication address used in the USB is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7043	The device used in the USB exceeds the device range.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7044	The device length of the communication data in the USB exceeds the limit.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7047	The device checksum for the communication serial port of the USB is incorrect.	1. Check the communication setting in the master and the slave. 2. Check the communication cable.	-	H
7203	Invalid communication function code	1. Refer to the function codes defined by the communication protocols. 2. Check if the product firmware and the software used are the most updated versions. 3. Make a note of the operation procedures and screenshots of the error windows and hand this note to the company or the technicians from the agents.	-	H
8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	1. Download the program and parameters again. 2. Check the communication cable. 3. Save all the projects and compress the projects into one compressed file and then hand this file to the company or the technicians from the agents.	-	H

8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	1. Download the program and parameters again. 2. Save all the projects and compress the projects into one compressed file and then hand this file to the company or the technicians from the agents.	-	H
8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	1. Download the program and parameters again. 2. Save all the projects and compress the projects into one compressed file and then hand this file to the company or the technicians from the agents.	-	H
8000 - 8FFF	Errors occur between software and PLC.	1. Check if the product firmware and the software used are the most updated versions. 2. Make a note of the operation procedures and screenshots of the error windows and hand this note to the company or the technicians from the agents.		

7.3 Troubleshooting for Analog Modules

Digital I/O, analog I/O, and temperature measurement modules can be installed in a DVP-ES3 Series system.

Error status is shown in error codes in CR#43. 0 indicates normal and 1 indicates error. Refer to the following table for more details.

Error Code	RUN LED	ERROR LED	Description	Solution
bit0	OFF	ON	The external voltage is abnormal.	Check the power supply.
bit1	RUN: Blinking STOP: OFF	Blinking	Hardware failure	Contact the factory.
bit2			Conversion value exceeds the set upper/lower value	Check the upper and lower value
bit3			The signal received by channel 1 exceeds the range of analog inputs (temperature).	Check the signal received by channel 1
bit4			The signal received by channel 2 exceeds the range of analog inputs (temperature).	Check the signal received by channel 2
bit5			The signal received by channel 3 exceeds the range of analog inputs (temperature).	Check the signal received by channel 3
bit6			The signal received by channel 4 exceeds the range of analog inputs (temperature).	Check the signal received by channel 4
bit7			The signal received by channel 5 exceeds the range of analog inputs (temperature).	Check the signal received by channel 5
bit8			The signal received by channel 6 exceeds the range of analog inputs (temperature).	Check the signal received by channel 6
bit9			Mode setting error	Check the mode setting
bit10			Average time setting error	Check the average time setting
bit11			Upper/lower value setting error	Check the upper/lower value setting
bit12			Set value cannot be changed.	Check the value in CR#40 (set value changing prohibited)
bit13			The later module is disconnected.	Check the wiring.

7.4 LED Indicators and Error Codes for CPU Modules

The error codes and LED indicators are presented in the following tables.

- **Descriptions used in the table**

a. Error code: If an error occurs in the system, an error code is generated.

b. Description: The description of the error

c. CPU status: If the error occurs, the CPU stops running, keeps running, or shows the status you defined for the error.

- Stop: The CPU stops running when the error occurs.
- Continue: The CPU keeps running when the error occurs.

d. LED indicator status: If the error occurs, the LED indicator is ON, OFF, or blinks.

- ERROR: System error

- **LED Indicator Description Table**

There are five types of error indicator states for of the CPU module errors, including LED indicator ON, OFF, blinking fast, blinking normally, and blinking slowly. When the LED indicator is ON, blinking fast/normally, clear the problems first in order to run the CPU module. When the LED indicator is blinking slowly, indicating a warning type of error codes, it does not require immediate action. Clear the problems when the module is powered off.

Module Type	LED Indicator	Descriptions
CPU	Error Type	**ON:** A serious error occurs in the module.
		Blinking fast (every 0.2 seconds): unstable power supply or hardware Failure.
		Blinking normally (every 0.5 second): system program errors or system cannot run.
	Warning Type	**Blinking slowly (every one second and off for three seconds):** a warning is triggered, but the system can still run.
		OFF: a warning is triggered, but the system can still run. You can modify the rules of how a warning is triggered, or use the SM/SR to show the warnings.

- **Error Code Description Table**

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
000A	Scan timeout	Stop	V				
000C	The program in the PLC is damaged.	Stop			V		
0010	The access to the memory in the CPU is denied.	Stop			V		
0011	The PLC ID is incorrect.	Continue					V
0012	The PLC password is incorrect.	Continue					V
0026	RTC cannot keep track of the current time	Continue					V
002A	24VDC power supply is not sufficient and then is recovered from low-voltage for less than 10 ms.	Continue		V			

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
002D	The PLC maximum password attempts exceeded.	Continue					V
002E	The access to the external memory of the CPU is denied.	Stop			V		
002F	PLC programs are not consistent with the system logs.	Stop			V		
0050	The memories in the latched special auxiliary relays are abnormal.	Continue					V
0051	The latched special data registers are abnormal.	Continue					V
0052	The memories in the latched auxiliary relays are abnormal.	Continue					V
0054	The latched counters are abnormal.	Continue					V
0055	The latched 32-bit counters are abnormal.	Continue					V
0056	The latched special auxiliary relay is abnormal.	Continue					V
0059	The latched data registers are abnormal.	Continue					V
005D	The CPU module does not detect a memory card.	Continue					V
005E	The memory card is initialized incorrectly.	Continue					V
0063	An error occurs when data is written to the memory card.	Continue					V
0064	A file in the memory card cannot be read.	Continue					V
0102	The interrupt number exceeds the range.	Stop			V		
0202	The MC instruction exceeds the range.	Stop			V		
0302	The MCR instruction exceeds the range.	Stop			V		
0D03	The operands used in DHSCS are not used properly.	Stop			V		
0E05	The operands HCXXX used in DCNT are not used properly.	Stop			V		
1800 - 180F	Errors occurred in the extension modules	Continue				V	
1900 - 191C	Heartbeat errors occurred in the slave of the Delta ASD-A2 control.	Continue				V	
1950	The initialization of the Delta ASD-A2 control has not yet been completed, the CANopen instructions cannot be executed.	Continue					V
2001	Not in the right mode for the ASDA-A2 while using the CANopen communication instruction.	Continue					V
2003	The device used in the program exceeds the device range.	Continue					V
200A	Invalid instruction	Stop			V		
200B	The operand n or the other constant operands K/H exceed the range.	Continue					V
200C	The operands overlap.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
200D	The binary to binary-coded decimal conversion is incorrect.	Continue					V
200E	The string does not end with 00.	Continue					V
2012	Incorrect division operation	Continue					V
2013	The value exceeds the range of values which can be represented by the floating-point numbers.	Continue					V
2014	The task designated by the TKON or YKOFF instruction is incorrect, or exceeds the range.	Continue					V
2017	The instruction BREAK is written outside of the FOR-NEXT loop.	Continue					V
2027	No such position planning table number or the format is incorrect.	Continue					V
2028	The high speed output instruction is being executed. Only one instruction can be executed at a time.	Continue					V
6004	The IP address filter is set incorrectly.	Continue					V
600D	RJ45 port is not connected.	Continue					V
6010	The number of the MODBUS TCP connections exceeds the range.	Continue			V		
6011	The number of the EtherNet/IP connections exceeds the range.	Continue			V		
6012	There are devices using the same IP address.	Continue					V
6100	The email connection is busy.	Continue					V
6103	The trigger attachment mode in the email is set incorrectly.	Continue					V
6104	The attachment in the email does not exist.	Continue					V
6105	The attachment in the email is too big.	Continue					V
6106	There is an SMTP server response timeout.	Continue					V
6107	There is an SMTP server response timeout.	Continue					V
6108	SMTP verification failed	Continue					V
6200	The remote communication IP address set in the TCP socket function is illegal.	Continue					V
6201	The local communication port set in the TCP socket function is illegal.	Continue					V
6202	The remote communication port set in the TCP socket function is illegal.	Continue					V
6203	The device from which the data is sent in the TCP socket function is illegal.	Continue					V
6206	The device that receives the data in the TCP socket function is illegal.	Continue					V
6208	The data that is received through the TCP socket exceeds the device range.	Continue					V
6209	The remote communication IP address set in the UDP socket function is illegal.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
620A	The local communication port set in the UDP socket function is illegal.	Continue					V
620C	The device from which the data is sent in the UDP socket function is illegal.	Continue					V
620F	The device that receives the data in the UDP socket function is illegal.	Continue					V
6210	The data that is received through the UDP socket exceeds the device range.	Continue					V
6212	There is no response from the remote device after the timeout period.	Continue					V
6213	The data received exceeds the limit.	Continue					V
6214	The remote device refuses the connection.	Continue					V
6215	The socket is not opened.	Continue					V
6217	The socket is opened.	Continue					V
6218	The data has been sent through the socket.	Continue					V
6219	The data has been received through the socket.	Continue					V
621A	The socket is closed.	Continue					V
7011	The device communication function code in COM1 is incorrect.	Continue					V
7012	The device communication address used in COM1 is incorrect.	Continue					V
7013	The device used in COM1 exceeds the device range.	Continue					V
7014	The device length of the communication data in COM1 exceeds the limit.	Continue					V
7017	The device checksum for the communication serial port of COM1 is incorrect.	Continue					V
7021	The device communication function code in COM2 is incorrect.	Continue					V
7022	The device communication address used in COM2 is incorrect.	Continue					V
7023	The device used in COM2 exceeds the device range.	Continue					V
7024	The device length of the communication data in COM2 exceeds the limit.	Continue					V
7027	The device checksum for the communication serial port of COM2 is incorrect.	Continue					V
7031	The device communication function code in Ethernet is incorrect.	Continue					V
7032	The device communication address used in Ethernet is incorrect.	Continue					V
7033	The device used in Ethernet exceeds the device range.	Continue					V
7034	The device length of the communication data in Ethernet exceeds the limit.	Continue					V

Error code	Description	CPU status	ERROR LED indicator status				
			ON	Blinking fast	Blinking normally	Blinking slowly	OFF
7037	The device checksum for the communication serial port of Ethernet is incorrect.	Continue					V
7041	The device communication function code in USB is incorrect.	Continue					V
7042	The device communication address used in USB is incorrect.	Continue					V
7043	The device used in USB exceeds the device range.	Continue					V
7044	The device length of the communication data in USB exceeds the limit.	Continue					V
7047	The device checksum for the communication serial port of USB is incorrect.	Continue					V
7203	Invalid communication function code	Continue					V
8105	The contents of the downloaded program are incorrect. The program syntax is incorrect.	Continue					V
8106	The contents of the downloaded program are incorrect. The length of the execution code exceeds the limit.	Continue					V
8107	The contents of the downloaded program are incorrect. The length of the source code exceeds the limit.	Continue					V
8000 - 8FFF	Errors occur between software and PLC.	Continue					V