

Industrial Automation Headquarters

Delta Electronics, Inc.
 Taoyuan Technology Center
 No.18, Xinglong Rd., Taoyuan City,
 Taoyuan County 33068, Taiwan
 TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Jiangsu) Ltd.
 Wujiang Plant 3
 1688 Jiangxing East Road,
 Wujiang Economic Development Zone
 Wujiang City, Jiang Su Province, P.R.C. 215200
 TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
 238 Min-Xia Road, Pudong District,
 Shanghai, P.R.C. 201209
 TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
 Tokyo Office
 2-1-14 Minato-ku Shibadaimon,
 Tokyo 105-0012, Japan
 TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
 1511, Byucksan Digital Valley 6-cha, Gasan-dong,
 Geumcheon-gu, Seoul, Korea, 153-704
 TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd.
 4 Kaki Bukit Ave 1, #05-05, Singapore 417939
 TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
 Plot No 43 Sector 35, HSIIDC
 Gurgaon, PIN 122001, Haryana, India
 TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas

Delta Products Corporation (USA)
 Raleigh Office
 P.O. Box 12173, 5101 Davis Drive,
 Research Triangle Park, NC 27709, U.S.A.
 TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A.
 Sao Paulo Office
 Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
 01332-000-São Paulo-SP-Brazil
 TEL: 55 11 3568-3855 / FAX: 55 11 3568-3865

Europe

Delta Electronics (Netherlands) B.V.
 Eindhoven Office
 De Witbogt 20, 5652 AG Eindhoven, The Netherlands
 TEL : +31 (0)40-8003800 / FAX : +31 (0)40-8003898

AH-0259720-01

2016-07-15

*We reserve the right to change the information in this manual without prior notice.

AH Motion - Standard Instructions Manual



AH Motion - Standard Instructions Manual

www.deltaww.com

AH Motion – Standard Instructions Manual

Contents

Preface

P.1	Introduction	II
P.1.1	Applicable Products	II
P.1.2	Related Manuals	II
P.2	Navigation between Manuals	III

Chapter 1 Introduction to Standard Instructions

1.1	Overview	1-2
1.2	Categories of Standard Instructions	1-2

Chapter 2 Devices, Symbols and Instructions

2.1	Common Devices	2-2
2.1.1	Functions of Common Devices	2-2
2.1.2	Common Device List.....	2-2
2.1.3	Latched Devices.....	2-3
2.1.4	Input Relays (X)	2-5
2.1.5	Output Relays (Y)	2-5
2.1.6	Auxiliary Relays (M)	2-6
2.1.7	Special Auxiliary Relays (SM/AM)	2-6
2.1.8	Data Registers (D)	2-6
2.1.9	Special Data Registers (SR/AR)	2-6
2.1.10	Link Registers (L).....	2-7
2.1.11	Stepping Relays (S)	2-7
2.1.12	Timers (T).....	2-7
2.1.13	Counters (C)	2-7
2.1.14	32-bit Counters (HC/AC)	2-8
2.1.15	Index Registers (E)	2-8
2.1.16	Values and Constants (K, 16#).....	2-8
2.1.17	Floating-point Numbers (F, DF).....	2-9
2.1.18	Strings (“\$”).....	2-9
2.1.19	Pointers (PR).....	2-9
2.1.19.1	Pointer Registers of Timers (T_Pointer) (TR).....	2-9

2.1.19.2. Pointer Registers of 16-bit Counters (C_Pointer) (CR).....	2-10
2.1.19.3. Pointer Registers of 32-bit Counters (HC_Pointer) (HCR)	2-10
2.2 Symbols.....	2-10
2.2.1 Application of Symbols	2-10
2.2.2 Classes.....	2-11
2.2.3 Data Types	2-11
2.2.4 Using Devices, Symbols and Instructions	2-13
2.2.5 Modifying a Symbol with an Index Register	2-14
2.3 Data Type Unit (DUT): ENUM	2-15
2.4 Instructions.....	2-15
2.4.1 Categories of Standard Instructions	2-16
2.4.2 List of Standard Instructions	2-17

Chapter 3 Standard Instructions

3.1 Applying This Chapter	3-3
3.1.1 Items of Standard Instructions	3-3
3.1.2 Restrictions on the Use of Standard Instructions	3-8
3.2 Ladder Instructions	3-10
3.3 Comparison Instructions	3-34
3.4 Arithmetic Instructions	3-70
3.5 Data Conversion Instructions	3-109
3.6 Data Transfer Instructions	3-154
3.7 Jump Instructions	3-180
3.8 Program Execution Instructions	3-189
3.9 I/O Refreshing Instructions	3-197
3.10 Convenience Instructions	3-200
3.11 Logic Instructions	3-237
3.12 Rotation Instructions	3-261
3.13 Basic Instructions	3-272

3.14	Shift Instructions	3-280
3.15	Data Processing Instructions	3-311
3.16	Structure Creation Instructions	3-360
3.17	Module Instructions	3-368
3.18	Floating Point Instructions	3-374
3.19	Real-time Clock Instructions	3-419
3.20	Peripheral Instructions	3-435
3.21	Communication Instructions	3-451
3.22	Other Instructions	3-477
3.23	String Processing Instructions	3-489
3.24	Ethernet Instructions	3-554
3.25	Memory Card Instructions	3-563
3.26	Task Control Instructions	3-576
3.27	SFC Control Instructions	3-579

Appendices

A.1.	Special Auxiliary Relays Table (SM)	A-2
A.1.1.	Refresh Time of Special Auxiliary Relay	A-25
A.2.	Special Data Registers Table (SR)	A-30
A.2.1.	Refresh Time of Special Data Registers	A-45
A.3.	Additional Remarks on SM and SR	A-48
A.4.	Standard Instructions (Sort by Alphabet)	A-60
A.5.	Error Codes and Troubleshooting	A-78
A.5.1.	Error Codes and Indicators.....	A-78
	AHxxEMC-5A.....	A-80
	Analog I/O Modules and Temperature Measurement Modules	A-106
	AH02HC-5A/AH04HC-5A.....	A-108
	AH05PM-5A/AH10PM-5A/AH15PM-5A	A-109

AH20MC-5A	A-110
AH10COPM-5A	A-112
AH10SCM-5A	A-113
A.5.2. Error Codes and Troubleshooting	A-113
AHxxEMC-5A.....	A-113
Analog I/O Modules and Temperature Measurement Modules	A-142
AH02HC-5A/AH04HC-5A.....	A-145
AH05PM-5A/AH10PM-5A/AH15PM-5A	A-148
AH20MC-5A	A-149
AH10SCM-5A	A-151
AH10COPM-5A	A-152
A.5.3. Troubleshooting for Limitation Errors.....	A-153
Troubleshooting for the software limit errors.....	A-153
Troubleshooting for the hardware limit errors.....	A-153
A.6. Table of Data Type Unit(DUT): Enum.....	A-155



Preface

Table of Contents

P.1	Introduction	II
P.1.1	Applicable Products	II
P.1.2	Related Manuals	II
P.2	Navigation between Manuals.....	III

P.1 Introduction

Thank you for purchasing the AH series Motion CPU with our advanced motion control system.

This manual introduces the standard instructions including basic instructions and the applied instructions. Please ensure that you understand the configuration and operations of the AH series motion control system, and use the AH series Motion CPU correctly.

To obtain required information for different system configurations, you can navigate between different manuals of AH Motion series manuals and other related manuals.

P.1.1 Applicable Products

This manual relates to the following products

- AHxxEMC-5A (AH08EMC-5A/AH10EMC-5A/AH20EMC-5A)

P.1.2 Related Manuals

The related manuals of the AH Motion series motion controllers are composed of the following.

1. **AH Motion - Hardware Manual**

It introduces function specifications, electrical specifications, appearances, dimensions, and etc.

2. **ISPSOft User Manual**

It introduces the use of ISPSOft, the programming languages (ladder diagrams, sequential function charts, function block diagrams, and structured texts), the concept of POUs, the concept of tasks, and the operation of motion control programming.

3. **AH Motion - Standard Instructions Manual**

It introduces the elements for standard programming including devices, symbols and standard instructions.

4. **AH Motion - Operation Manual**

It introduces basic knowledge of motion control structure, software/hardware setup, quick start of Software operations, devices to be used, motion control operations and troubleshooting.

5. **AH Motion - Motion Control Instructions Manual**

It introduces the elements for motion control programming including axis parameters, symbols and single axis/multi-axes motion instructions.

6. **AH Motion - Communication Manual**

It introduces the concept of communication protocols (e.g. EtherCAT) and the configuration of AH Motion products based on the protocols.

7. **AH500 Motion Control Module Manual**

It introduces the specifications for the AH500 series motion control modules, the wiring, the instructions, and the functions.

8. **AH500 Module Manual**

It introduces the use of special I/O modules of AH500 series PLCs. For example, network modules, analog I/O modules, temperature measurement modules, and etc.

P.2 Navigation between Manuals

Before using the products, there are three manuals that should be utilized as fundamental information: **AH Motion - Hardware Manual**, **ISPSOft User Manual**, and **AH Motion - Standard Instructions Manual**.

With the fundamental manuals, you can understand the basic information of hardware configuration, operation procedures of the software, and the basic instructions for using the system.

To obtain required information for different system configurations and applications, refer to other manuals as indicated in the table below. Reading all manuals related to your system configuration helps you make the most use of the AH series motion control system.

Related manuals	AH Motion series manuals							
	Fundamental			AH Motion – Operation Manual	AH Motion – Motion Control Instructions Manual	AH Motion – Communication Manual	AH500 Motion Control Module Manual	AH500 Module Manual
	AH Motion – Hardware Manual	ISPSOft User Manual	AH Motion – Standard Instructions Manual					
General operation procedures								
1. Overview of AH Motion series products	V							
2. Setting up hardware configuration for the system	V							
for motion control applications				V				
for communication (e.g. EtherCAT)						V		
for additional motion control modules							V	
for I/O extension using AH500 series modules								V
3. Getting started with the software	V							
for motion control applications				V				
for communication (e.g. EtherCAT)						V		
for additional motion control modules							V	
for I/O extension using AH500 series modules								V
4. Programming	V		V					
for motion control applications				V	V			
for communication (e.g. EtherCAT)						V		
for additional motion control modules							V	
for I/O extension using AH500 series modules								V

Related manuals	AH Motion series manuals							
	Fundamental			AH Motion – Operation Manual	AH Motion – Motion Control Instructions Manual	AH Motion – Communication Manual	AH500 Motion Control Module Manual	AH500 Module Manual
	AH Motion – Hardware Manual	ISPSOFT User Manual	AH Motion – Standard Instructions Manual					
General operation procedures								
5. Testing and troubleshooting								
for motion control applications					V*			
for communication (e.g. EtherCAT)		V		V		V		
for additional motion control modules			V*				V	
for I/O extension using AH500 series modules			V*					V
6. Maintenance and Inspection	V							

***Note:** Information regarding Error codes and Indicators and the associate troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to **AH Motion – Operation Manual**.

Chapter 1 Introduction to Standard Instructions

Table of Contents

1.1 Overview 1-2

1.2 Categories of Standard Instructions 1-2

1.1 Overview

This manual introduces the elements for standard programming including devices, symbols and standard instructions.

Most instructions contain operands which function as input parameters and output results. Every operand has applicable devices, such as X, Y, D, SM, SR, and the device can be assigned to a symbol according to the data type of the symbol.

Before using the instructions, please be sure that you understand the devices, symbols and the function of instructions sufficiently.

You can also refer to the **Appendices** for a quick reference of the special auxiliary relays (SM), special data registers (SR), standard instructions (sorted by alphabet), and error codes.

1.2 Categories of Standard Instructions

Categories	Description
<u>Ladder Instructions</u>	Basic instructions frequently used in ladder diagram programming such as LD, LDP, AND, OR, OUT, and etc
<u>Comparison instructions</u>	Comparisons such as =, <>, >, >=, <, <=, and etc.
<u>Arithmetic instructions</u>	Using binary numbers or binary-coded decimal numbers to add, subtract, multiply, or divide.
<u>Data conversion instructions</u>	Converting the binary-coded decimal number into the binary number, and converting the binary number into the binary-coded decimal number
<u>Data transfer instructions</u>	Transfer the specified data
<u>Jump instructions</u>	The program jumps.
<u>Program execution instructions</u>	Enabling or disabling the interrupt
<u>I/O refreshing instructions</u>	Refreshing the I/O.
<u>Convenience instructions</u>	Instructions which are applied to the counters, the teaching timers, the special timers, and etc.
<u>Logic instructions</u>	Logical operations such as logical addition, logical multiplication, and etc.
<u>Rotation instructions</u>	Rotating/Shifting the specified data
<u>Basic instructions</u>	Timer instructions and counter instructions
<u>Shift instructions</u>	Shifting the specified data
<u>Data processing instructions</u>	16-bit data processing such as decoding and encoding.
<u>Structure creation instructions</u>	Nested loops
<u>Module instructions</u>	Reading the data from the special module and writing the data into the special module
<u>Floating-point number instructions</u>	Floating-point number operations
<u>Real-time clock instructions</u>	Reading/Writing, adding/subtracting and comparing the time
<u>Peripheral instructions</u>	I/O points connected to the peripheral

Categories	Description
<u>Communication instructions</u>	Controlling the peripheral though communication
<u>Other instructions</u>	Instructions which are different from those mentioned above
<u>String processing instructions</u>	Conversion between binary/binary-coded decimal numbers and ASCII codes; conversion between binary numbers and strings; conversion between floating-point numbers and strings; string processing
<u>Ethernet instructions</u>	Controlling the Ethernet data exchange
<u>Memory card instructions</u>	Reading the data from the memory card and writing the data into the memory card
<u>Task control instructions</u>	Controlling the task in the program
<u>SFC control instructions</u>	Controlling the SFC operation

Memo

1

Chapter 2 Devices, Symbols and Instructions

Table of Contents

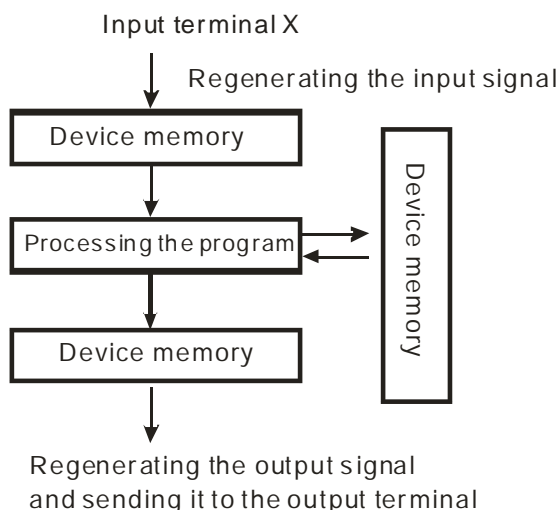
2.1	Common Devices	2-2
2.1.1	Functions of Common Devices	2-2
2.1.2	Common Device List.....	2-2
2.1.3	Latched Devices.....	2-3
2.1.4	Input Relays (X)	2-5
2.1.5	Output Relays (Y)	2-5
2.1.6	Auxiliary Relays (M)	2-6
2.1.7	Special Auxiliary Relays (SM/AM)	2-6
2.1.8	Data Registers (D)	2-6
2.1.9	Special Data Registers (SR/AR)	2-6
2.1.10	Link Registers (L).....	2-7
2.1.11	Stepping Relays (S)	2-7
2.1.12	Timers (T)	2-7
2.1.13	Counters (C)	2-7
2.1.14	32-bit Counters (HC/AC).....	2-8
2.1.15	Index Registers (E)	2-8
2.1.16	Values and Constants (K, 16#).....	2-8
2.1.17	Floating-point Numbers (F, DF)	2-9
2.1.18	Strings (“\$”)	2-9
2.1.19	Pointers (PR).....	2-9
2.1.19.1.	Pointer Registers of Timers (T_Pointer) (TR).....	2-9
2.1.19.2.	Pointer Registers of 16-bit Counters (C_Pointer) (CR).....	2-10
2.1.19.3.	Pointer Registers of 32-bit Counters (HC_Pointer) (HCR).....	2-10
2.2	Symbols.....	2-10
2.2.1	Application of Symbols	2-10
2.2.2	Classes.....	2-11
2.2.3	Data Types	2-11
2.2.4	Using Devices, Symbols and Instructions	2-12
2.2.5	Modifying a Symbol with an Index Register	2-13
2.3	Data Type Unit (DUT): ENUM	2-15
2.4	Instructions.....	2-15
2.4.1	Categories of Standard Instructions	2-16
2.4.2	List of Standard Instructions	2-17

2.1 Common Devices

This section describes the concept of common devices which include input/output/auxiliary relays, timers, counters, and data registers. For detailed descriptions on functions as well as characteristics of devices, you can refer to **AH Motion – Operation Manual**.

2.1.1 Functions of Common Devices

The procedure for processing the 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 of the input signal.
 2. When program is executed, the state in the memory of the input signal does not change even if the input signal changes from ON to OFF or from OFF to ON. Not until the next scan begins will the input signal be refreshed.
- 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, and the results are stored in the device memories.
- 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.

2.1.2 Common Device List

Type	Device name		Number of devices	Range
Bit device	Input relay	X	8192	X0.0~X511.15 (Supporting Force ON/OFF)
	Output relay	Y	8192	Y0.0~Y511.15 (Supporting Force ON/OFF)
	Data register	D	1048576	D0.0~D65535.15
	Link registers	L	1048576	L0.0~ L65535.15
	Auxiliary relay	M	8192	M0~M8191
	Special auxiliary relay	SM/AM/AR	SM: 2048 AM: 16384 (AHxxEMC) AR:1048576 (AHxxEMC)	SM0~SM2047 AM0~AM16383 (AHxxEMC) AR0.15~AR65535.15 (AHxxEMC)
	Stepping relay	S	2048	S0~S2047
	Timer	T	2048	T0~T2047
	Counter	C	2048	C0~C2047
	32-bit counter	HC/AC	HC: 64 AC: 56 (AHxxEMC)	HC0~HC63 AC0~AC55 (AHxxEMC)

Type	Device name		Number of devices	Range
Word device	Input relay	X	512	X0~X511
	Output relay	Y	512	Y0~Y511
	Data register	D	65536	D0~D65535
	Special data register	SR/AR	SR: 2048 AR: 65536 (AHxxEMC)	SR0~SR2047 AR0~AR65535 (AHxxEMC)
	Link registers	L	1048576	L0~ L65535
	Timer	T	2048	T0~T2047
	Counter	C	2048	C0~C2047
	32-bit counter	HC/AC	HC: 64 AC: 56 (AHxxEMC)	HC0~HC63 AC0~AC55 (AHxxEMC)
	Index register	E	32	E0~E31
Constant*	Decimal system	K	16 bits: -32768~32767 32 bits: -2147483648~2147483647	
	Hexadecimal system	16#	16 bits: 16#0~16#FFFF 32 bits: 16#0~16#FFFFFFFF	
	Single-precision floating-point number	F	32 bits: $\pm 1.17549435^{-38} \sim \pm 3.40282347^{+38}$	
	Double-precision floating-point number	DF	64 bits: $\pm 2.2250738585072014^{-308} \sim \pm 1.7976931348623157^{+308}$	
String*	String	"\$"	1~31 characters	
Pointer*	Pointer	PR		

*1: The decimal forms are notated by K in the device table in chapter 3, whereas they are entered directly in ISPSOft. For example, entering 50 in ISPSOft indicates the value K50.

*2: The floating-point numbers are notated by F/DF in the device table in chapter 3, whereas they are represented by decimal points in ISPSOft. For example, entering 500.0 in ISPSOft indicates the value F500.

*3: The strings are notated by \$ in chapter 3, whereas they are represented by adding quotes (" ") to the value in ISPSOft. For example, entering "1234" in ISPSOft indicates the string 1234.

2.1.3 Latched Devices

- Latched areas of each type of device

Device	Function	Device range	Latched area
X	Input relay	X devices (bit): X0.0~X511.15 X devices (word): X0~X511	All devices are non-latched.
Y	Output relay	Y devices (bit):	All devices are non-latched.

Device	Function	Device range	Latched area
		Y0.0~Y511.15 Y devices (word): Y0~Y511	
M*	Auxiliary relay	M0~M8191	Default: M0~M8191
SM/AM/AR	Special auxiliary relay	SM: SM0~SM2047 AM: AM0~AM16383 (AHxxEMC) AR: AR0.15~AR65535.15 (AHxxEMC)	Some devices are latched, and can not be changed. Refer to the table of special auxiliary relays in the appendices for more information.
S	Stepping relay	S0~S2047	All devices are non-latched.
T*	Timer	T0~T2047	Default: T0~T2047.
C*	Counter	C0~C2047	Default: C0~C2047.
HC/AC*	32-bit counter	HC: HC0~HC63 AC: AC0~ AC55 (AHxxEMC)	Default: HC0~HC63. AC devices are non-latched.
D*	Data register	D device (bit): D0.0~D65535.15 D device (word): D0~D65535	Default: D0~D32767. At most 32768 devices can be latched devices.
SR/AR	Special data register	SR: SR0~SR2047 AR: AR0~AR65535 (AHxxEMC)	Some are latched, and can not be changed. Refer to the list of special data registers for more information.
L	Link register	L0~ L65535	All devices are non-latched.
E	Index register	E0~E31	All devices are non-latched.

***Note:** You can define the range of latched areas for T, C, HC and D and set the devices to be non-latched. Note that the range should not exceed the available device range, and only 32768 D devices at most can be non-latched. For example, you can set D50~D32817 or D32768~D65535 to latched area though the default range of latched areas is D0~D32767.

● Behavior of non-latched and latched devices

PLC action		Device type	Non-latched	Latched	Output relay
Power: OFF→ON			Cleared	Retained	Cleared
STOP ↓ RUN	The output relay is cleared.		Retained	Retained	Cleared
	The state of the output relay is retained.		Retained	Retained	Retained
	The state of the output relay returns to that before the PLC's stopping.		Retained	Retained	The state of the output relay returns to that before the PLC's stopping.
	The non-latched devices are cleared.		Cleared	Retained	Cleared
	The state of latched devices is retained.		Retained	Retained	Retained

PLC action \ Device type	Non-latched	Latched	Output relay
RUN→STOP	Retained	Retained	Retained
SM204 is ON. (All non-latched devices are cleared.)	Cleared	Retained	Cleared
SM205 is ON. (All latched devices are cleared.)	Retained	Cleared	Retained
Default value	0	0	0

2.1.4 Input Relays (X)

- The function of the input

The input is connected to the input device (e.g. external devices such as button switches, rotary switches, number switches, and etc.), and the input signal is read into the PLC. Besides, contact A or contact B of the input can be used several times in the program, and the ON/OFF state of the input varies with the ON/OFF state of the input device.

- The input number (the decimal number)

For the PLC, the input numbers start from X0.0. The number of inputs varies with the number of inputs on the digital input/output modules, and 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 on the PLC can reach up to 8192, and the range is between X0.0 and X511.15.

- The input type

The inputs are classified into two types.

1. Regenerated input: Before the program is executed, the data is fed into the PLC according to the states of the inputs which are regenerated. For example, LD X0.0.
2. Direct input: During the execution of the instructions, the data is fed into the PLC according to the states of the inputs. For example, LD DX0.0.

2.1.5 Output Relays (Y)

- The function of the output

The task of the output is sending the ON/OFF signal to drive the load connected to the output. The load can be an external signal lamp, a digital display, or an electromagnetic valve. There are three types of outputs. They are relays, transistors, and TRIACs (AC thyristors). Contact A or contact B of the output can be used several times in the program, but the output should be used only once in the program. Otherwise, according to the program-scanning principle of the PLC, the state of the output depends on the circuit connected to the last output in the program.

- The output number (the decimal number)

For the PLC, the output numbers start from Y0.0. The number of outputs varies with the number of outputs on the digital input/output modules, and 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 can reach up to 8192, and the range is between Y0.0 and Y511.15.

The output which is not practically put to use can be used as a general device.

- The output type

The outputs are classified into two types.

1. Regenerated output: Not until the program executes the instruction END is the information fed out according to the states of the outputs. For example, OUT Y0.0.
2. Direct output: When the instructions are executed, the information is fed out according to the states of the outputs. For example, OUT DY0.0.

2.1.6 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 means of the auxiliary relay, but can not drive the external load by means of the auxiliary relay. The auxiliary relays can be divided into two types according to their attributes.

1. For general use: If an electric power cut occurs when the PLC is running, the auxiliary relay for general use will be reset to OFF. When the power supply is restored, the auxiliary relay for general use is still OFF.
2. For latched use: If an electric power cut occurs when the PLC is running, the state of the auxiliary relay for latched use will be retained. When the power supply is restored, the state remains the same as that before the power electric cut.

2.1.7 Special Auxiliary Relays (SM/AM)

SM: special auxiliary relays

Every special auxiliary relay has its specific function. Please do not use the special auxiliary relays which are not defined. For function descriptions of each special auxiliary relay (SM), refer to **Appendix 1: Special Auxiliary Relays Table**.

AM: special auxiliary relays for motion axis

For detailed explanation of special auxiliary relays for axis (AM), refer to **Chapter 6: Understanding Motion Control Devices** of *AH Motion - Operation Manual*.

2.1.8 Data Registers (D)

The data register stores the 16-bit data. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -32,768 to +32,767. Two 16-bit registers can be combined into a 32-bit register, i.e. (D+1, D) in which the register whose number is smaller represents the low 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -2,147,483,648 to +2,147,483,647. Besides, four 16-bit registers can be combined into a 64-bit register, i.e. (D+3, D+2, D+1, D) in which the register whose number is smaller represents the lower 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -9,223,372,036,854,776 to +9,223,372,036,854,775,807. The data registers also can be used to refresh the values in the control registers in the modules other than digital I/O modules. Please refer to ISPSOFT User Manual for more information regarding refreshing the values in the control registers.

The registers can be classified into two types according to their properties:

1. General-purpose register: If the PLC begins to run, or is disconnected, the value in the register will be cleared to zero. If you want to retain the data when the PLC begins to RUN, you can refer to ISPSOFT User Manual for more information. Please notice that the value will still be cleared to zero if the PLC is disconnected.
2. Latched register: If the PLC is disconnected, the data in the latched register will not be cleared. In other words, the value before the disconnection is still retained. If you want to clear the data in the latched area, you can use RST or ZRST.

2.1.9 Special Data Registers (SR/AR)

SR: special data registers.

Every special data register has its definition and specific function. The system statuses and the error messages are stored in the special data registers. Besides, the special data registers can be used to monitor the system statuses.

For function descriptions of each special data register (SR), refer to **Appendix 2: Special Data Registers Table**

AR: special data registers for motion axis

For detailed explanation of special data registers for axis (AR), refer to **Chapter 6: Understanding Motion Control Devices** of *AH Motion - Operation Manual* for more information

2.1.10 Link Registers (L)

The link register is mainly used in for automatic data exchange function. When the data exchange occurs between the AHxxEMC series PLCs, the link register can be used as the buffer.

The link registers L0~L65535 have 65536 words and can be used as the common auxiliary registers.

2.1.11 Stepping Relays (S)

The function of the stepping relay:

The stepping relay can be easily used in the industrial automation to set the procedure. It is the most basic device in the sequential function chart (SFC). Please refer to ISPSOFT User Manual for more information related to sequential function charts.

There are 2048 stepping relays, i.e. S0~S2047. Every stepping relay is like an output relay in that it has an output coil, contact A, and contact B. It can be used several times in the program, but it can not directly drive the external load. Besides, the stepping relay can be used as a general auxiliary relay when it is not used in the sequential function chart.

2.1.12 Timers (T)

1. 100 millisecond timer: The timer specified by the instruction TMR takes 100 milliseconds as the timing unit.
2. 1 millisecond timer: The timer specified by the instruction TMRH takes 1 millisecond as the timing unit.
3. The timers for the subroutine's exclusive use are T1920~T2047.
4. The accumulative timers are ST0~ST2047. If you want to use the device-monitoring function, you can monitor T0~T2047.
5. If the same timer is used repeatedly in the program, including in different instructions TMR and TMRH, the setting value is the one that the value of the timer matches first.
6. If the same timer is used repeatedly in the program, it is OFF when one of the conditional contacts is OFF.
7. If the same timer is used repeatedly in the program as the timer for the subroutine's exclusive use and the accumulative timer in the program, it is OFF when one of the conditional contacts is OFF.
8. When the timer is switched from ON to OFF and the conditional contact is ON, the timer is reset and counts again.
9. 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 state of the contact is as follows.

Normally open (NO) contact	ON
Normally closed (NC) contact	OFF

Refer to **Chapter 5: Understanding Common Devices** of *AH Motion - Operation Manual* for more information.

2.1.13 Counters (C)

The function of the counter:

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

The 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 there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
3. If you use the instruction MOV or ISPSOft to transmit a value bigger than the setting value to the current value register C0, the contact of the counter C0 will be ON and the current value will become the same as the setting value next time X0.1 is switched from OFF to ON.
4. You can use either the constant or the value in the register as the setting value of the counter.
5. The setting value of the counter can be a positive or a negative. If the counter counts up from 32,767, the next current value becomes -32,768.

Refer to **Chapter 5: Understanding Common Devices** of *AH Motion - Operation Manual* for more information.

2.1.14 32-bit Counters (HC/AC)

HC: The 32-bit general-purpose addition/subtraction counter

AC: The 32-bit counters used specifically for motion axis. The function of **AC** is the same as that of **HC**

1. Setting range: -2,147,483,648~2,147,483,647
2. The switch 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 the addition counter when SM621 is OFF, whereas HC0 is the subtraction counter when SM621 is ON.
3. You can use either the constant or the value in the data registers as the setting value of the counter, and the setting value can be a positive or a negative. If you use the value in the data registers as the setting value of the counter, the setting value occupies two consecutive registers.
4. For the general-purpose counter, the current value of the counter is cleared when there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
5. If the counter counts up from 2,147,483,647, the next current value becomes -2,147,483,648. If the counter counts down from -2,147,483,648, the next current value becomes 2,147,483,647.

Refer to **Chapter 5: Understanding Common Devices** of *AH Motion - Operation Manual* for more information.

2.1.15 Index Registers (E)

The index register is the 16-bit data register. It is like the general register in that the data can be read from it and written into it. However, it is mainly used as the index register. The range of index registers is E0~E13.

Refer to **2.2.5 Modifying a Symbol with an Index Register** for more information.

2.1.16 Values and Constants (K, 16#)

The PLC uses four types of values to execute the operation according to different control purposes. The functions of these values are illustrated as follows:

1. Binary number (BIN)

The PLC adopts the binary system to operate the values.

2. Decimal number (DEC)

The decimal number in the PLC is used as

- the setting value of the timer (T) or the setting value of the counter (C/HC). For example, TMR C0 50 (**constant K**).
- the device number. For example, M10 and T30 (device number)
- the number before or after the decimal point. For example, X0.0, Y0.11, and D10.0 (device number).
- **the constant K**: It is used as the operand in the applied instruction. For example, MOV 123 D0 (**constant K**).

3. Binary-coded decimal (BCD)

A decimal value is represented by a nibble or four bits, and therefore sixteen consecutive bits can represent a four-digit decimal value.

4. Hexadecimal number (HEX)

The hexadecimal number in the PLC is used as

- **the constant 16#**: It is used as the operand in the applied instruction. For example, MOV 16#1A2B D0 (hexadecimal constant).

Refer to **Chapter 5: Understanding Common Devices of AH Motion - Operation Manual** for more information.

2.1.17 Floating-point Numbers (F, DF)

The floating-point numbers are represented by decimal points in ISPSOft. For example, the floating-point number of 500 is 500.0.

Refer to **Chapter 5: Understanding Common Devices of AH Motion - Operation Manual** for more information.

2.1.18 Strings (“\$”)

What strings can process are ASCII codes (*1). A complete string begins with a start character, and ends with an ending character (NULL code). If what you enter is a string, you can enter 31 characters at most, and the ending character 16#00 will be added automatically in ISPSOft.

Refer to **Chapter 5: Understanding Common Devices of AH Motion - Operation Manual** for more information.

2.1.19 Pointers (PR)

- **ISPSOft** supports the function blocks. When the symbol declaration type is VAR_IN_OUT, and the data type is POINTER, the symbol will be assigned as pointer registers (PR). The value in the PR can refer directly to the value of device X, Y, D, or L, and the PR can also refer to the symbols with addresses which are automatically allocated by ISPSOft.
- You can declare 16 pointer registers in every function block: PR0~PR15 or PR0.0~PR15.15.

2.1.19.1. Pointer Registers of Timers (T_Pointer) (TR)

- **ISPSOft** supports the function blocks. If you want to use the timer in the function block, you have to declare a pointer register of the timer (TR) in the function block. The address of the timer is transmitted to TR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is T_POINTER, the symbol will be assigned as TR. The value in the TR can refer directly to the value of device T or the symbols which are assigned as timers by **ISPSOft**.

- You can declare 8 pointer registers of the timers in every function block: TR0~TR7.
- If you want to use an instruction in the function block, and the timer is supported among the operands, you have to use the pointer register of the timer.

2.1.19.2. Pointer Registers of 16-bit Counters (C_Pointer) (CR)

- **ISPSOft** supports the function blocks. If you want to use the 16-bit counter in the function block, you have to declare a pointer register of the 16-bit counter (CR) in the function block. The address of the 16-bit counter is transmitted to the CR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is C_POINTER, the symbol will be assigned as CR. The value in the CR can refer directly to the value of device C or the symbols which are assigned as 16-bit counters by **ISPSOft**.
- You can declare 8 pointer registers of the 16-bit counters in every function block: CR0~CR7.
- If you want to use an instruction in the function block, and the counter is supported among the operands, you have to use the pointer register of the 16-bit counter.

2.1.19.3. Pointer Registers of 32-bit Counters (HC_Pointer) (HCR)

- **ISPSOft** supports the function blocks. If you want to use the 32-bit counter in the function block, you have to declare a pointer register of the 32-bit counter (HCR) in the function block. The address of the 32-bit counter is transmitted to the HCR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is HC_POINTER, the symbol will be assigned as HCR. The value in the HCR can refer directly to the value of device HC or the symbols which are assigned as 32-bit counters by **ISPSOft**.
- You can declare 8 pointer registers of the 32-bit counters in every function block: HCR0~HCR7.
- If you want to use an instruction in the function block, and the 32-bit counter is supported among the operands, you have to use the pointer register of the 32-bit counter.

Refer to **Chapter 5: Understanding Common Devices** of *AH Motion - Operation Manual* for more information.

2.2 Symbols

During the process of developing a traditional program for a PLC, it generally takes much time to manage device addresses. Besides, managing or debugging the program in a big project is a burden on users. As a result, the concept of symbols* in a high-level programming language is introduced by IEC 61131-3. A device in a PLC can be represented by a symbol, and a device can be automatically assigned to a symbol. In this way, the time to assign devices is saved, the program is more readable, and the efficiency of developing a program increases.

***Note:** Variables in **ISPSOft** are called symbols. As a result, variables are the same as symbols in terms of meaning in this manual.

2.2.1 Application of Symbols

A symbol has to be declared before it is used. There are two types of symbols. They are global symbols and local symbols. The global symbols in a project can be used in all the POU*s in the project, and the local symbols in a project can only be used in the POU in which the local symbols are declared. Besides, the identifier of a local symbol in a POU can be the same as the identifier of a local symbol in another POU. However, if this local symbol identifier is the same as a global symbol identifier, the system will automatically regard it as a local symbol.

***Note:** For further explanations regarding a POU, refer for **ISPSOft User Manual**.

2.2.2 Classes

Symbols can be categorized into four classes. The features of the four classes are described below.

- **VAR - General symbol**

The symbols of this class are for general operations only. The significance of a symbol of this class depends on the data type of the symbol.

- **INPUT - Symbol used as an input pin of a function block**

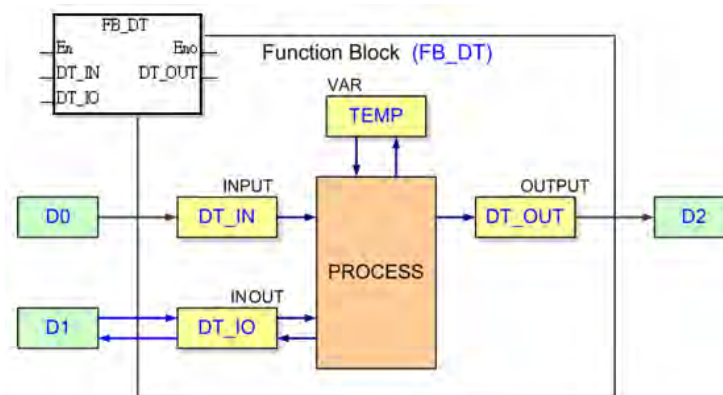
A symbol of this class is used as an input pin of a function block. It can only be declared in the function block. If a function block is called, the symbols of this class can receive the input values sent by the caller. In a ladder diagram, INPUTs are put at the left sides of function blocks with pins receiving the input values sent by the caller.

- **OUTPUT - Symbol used as an output pin of a function block**

A symbol of this class is used as an output pin of a function block. It can only be declared in a function block. After the execution of a function block is complete, the operation results will be sent to the caller through the symbols of this class. In a ladder diagram, OUTPUTs are put at the right sides of function blocks with pins sending the operation results to the caller.

- **INOUT - Symbol used as a feedback pin of a function block**

A symbol of this class is used as a feedback pin of a function block. It can only be declared in the function block. Please refer to the following example. When the function block is called, the caller sends the value in D1 to DT_IO, which is a symbol of the INOUT class. After the operation comes to an end, the final value of DT_IO is sent to D1. In a ladder diagram, the symbols of this class are put at the left sides of function blocks.



2.2.3 Data Types

The data type of a symbol determines the format of the value of the symbol. Suppose there are two symbols VAR_1 and VAR_2. The data type of VAR_1 is BOOL, and the data type of VAR_2 is WORD. If VAR_1 and VAR_2 are used in a program, VAR_1 will represent a contact, and VAR_2 will represent a 16-bit device which can be involved in arithmetic operation or transferring the data.

The data types supported by ISPSOFT are listed below.

Data type	Name	Description	Program	Function block
BOOL	Boolean	A Boolean value represents the state of a contact, could be TRUE or FALSE.	✓	✓
WORD	Word	Bit string of length 16.	✓	✓
DWORD	Double Word	Bit string of length 32.	✓	✓

2

Data type	Name	Description	Program	Function block
LWORD	Long Word	Bit string of length 64.	✓	✓
UINT	Unsigned integer	16-bit data.	✓	✓
UDINT	Unsigned double integer	32-bit data.	✓	✓
INT	Integer	16-bit data	✓	✓
DINT	Double integer	32-bit data.	✓	✓
LINT	Long Integer	64-bit data.	✓	✓
REAL	Real numbers	32-bit data; applicable to single precision floating-point instructions.	✓	✓
LREAL	Long reals	64-bit data; applicable to double precision floating-point instructions.	✓	✓
CNT	Counter	16-bit counter value or 32-bit counter value.	✓	✓
TMR	Timer	16-bit timer value.	✓	✓
ARRAY	Array	If a symbol is declared as an array, the size of an array and an array data type must be specified. (An array is composed of 256 elements/members at most.)	✓	✓
String	Character string	Variable-length single-byte data string. 8 bits as one ASCII character. The string length needs to be specified on declaration.	✓	✓

2.2.4 Using Devices, Symbols and Instructions

A device is assigned to a symbol according to the data type of the symbol. You can set the initial value of the device. After the program in a project is downloaded to a motion controller, the initial values will be written into the devices assigned to the symbols at the first scan cycle.

The principles of assigning devices to symbols are as follows.

- Devices can be assigned to the global symbols and the local symbols declared in the program POU by users, or assigned automatically by system.
- Local symbols (except the symbols of VAR class) declared in a function block can only be assigned with devices by the system.
- The system will assign applicable devices only. (You can set a range of devices which can be assigned automatically.)
- If a symbol is declared, the device assigned to the symbol, the data type of the symbol, and the initial value of the symbol must be compatible with one another.

The relation between the data types and the device types which can be assigned is described below.

Data type	AH Motion CPU	
	Device assigned by users	Device assigned by the system
BOOL	Contact M/SM or bit in the device X/Y (*3)	Contact M/SM
WORD	D	W

Data type	AH Motion CPU	
	Device assigned by users	Device assigned by the system
DWORD	D	W
LWORD	D	W
UINT	D	W
UDINT	D	W
INT	D	W
DINT	D	W
LINT	D	W
REAL	D	W
LREAL	D	W
CNT	C	C
TMR	T	T
ARRAY	`The devices assigned to a symbol whose data type is ARRAY depend on the array type specified. An array is composed of the devices starting from the device assigned by users or the system, and the number of devices in an array conforms to the size of the array.	
String	N/A	

*1. Please refer to *ISPSOft User Manual* for more information about setting a range of devices which can be assigned automatically.

*2. A symbol representing a function block definition has a special significance. Please refer to *ISPSOft User manual* for more information.

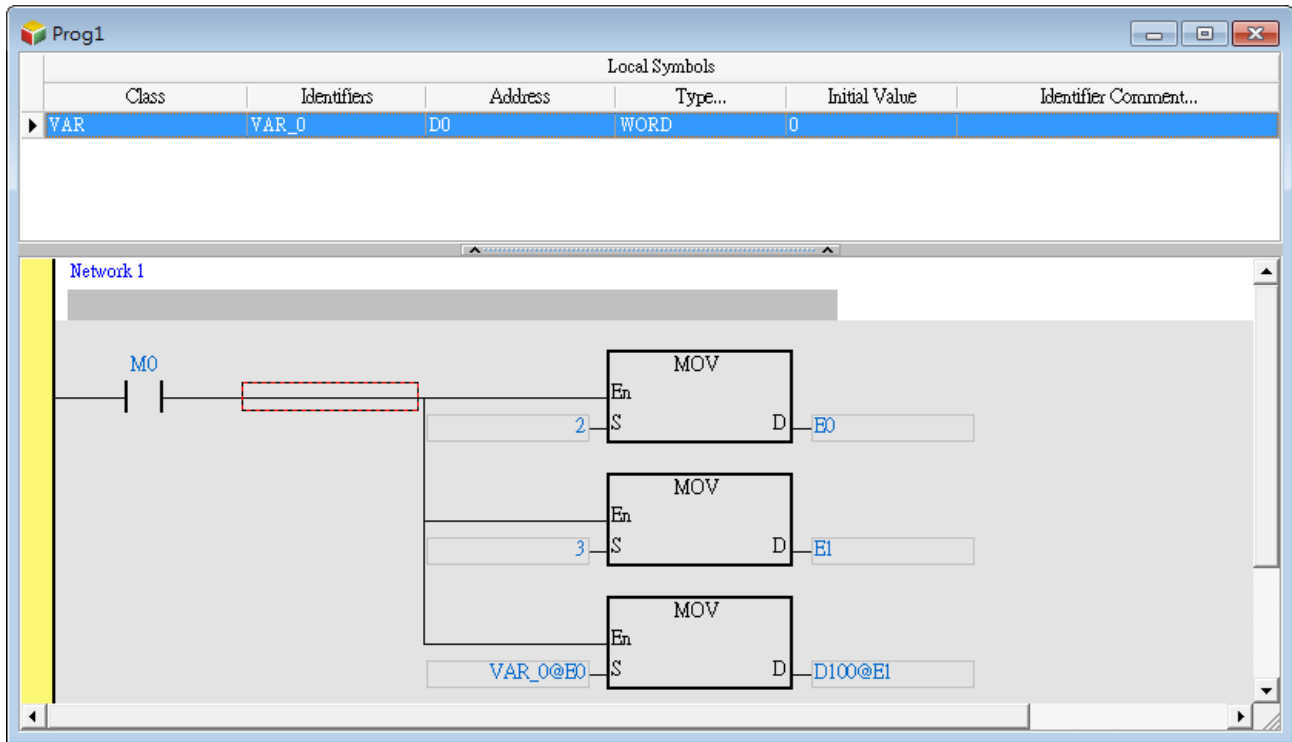
*3. X0.0 and Y0.1 are bits in the word devices X and Y. Please refer to *ISPSOft User manual* for more information.

2.2.5 Modifying a Symbol with an Index Register

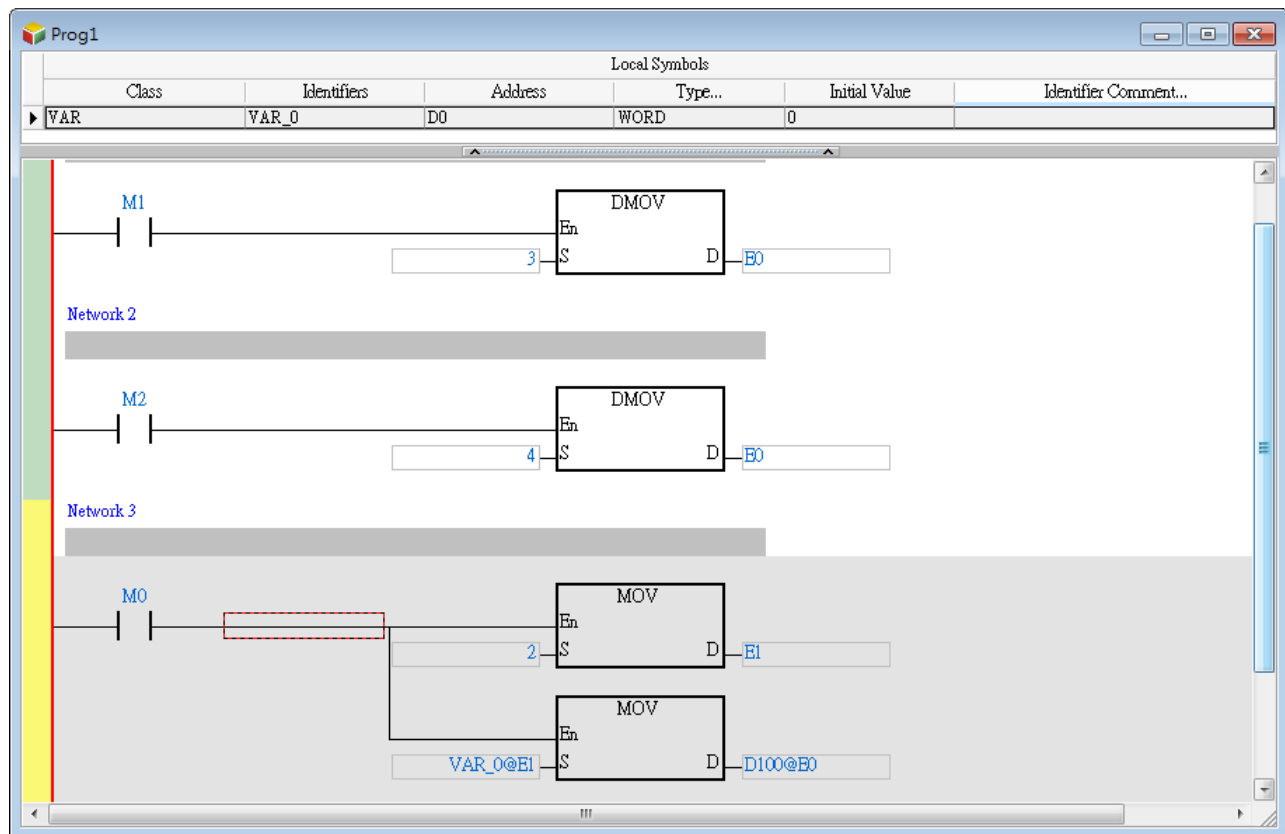
You are allowed to use index registers (E device) in ISPSOft to modify a symbol. The E devices are like general 16-bit data registers. You can write data into the E devices and read data from the E devices freely. If an E device is used as a general register, it can only be used in a 16-bit instruction. The modification of a symbol by an index register is represented by the format: **Identifier@Index register**. If an E device is used to modify an operand, it can be used in a 16-bit instruction or a 32-bit instruction.

Please refer to the program below. The device assigned to VAR_0 is D0. The data stored in an index register indicates the offset for the symbol which the index register modifies. If the value in the index register E0 is 2, VAR_0@ E0 indicates that 2 is added to the device address (D0) assigned to VAR_0, that is, VAR_0@E0 represents D2. If M0 is ON, the value in E0 will be 2, the value in E1 will be 3, and the value in D2 will be moved to D103.

2



Besides, if the value in an index register is changed, the device which actually operates differs from the original device. As a result, if the original device is not used in the program, the final value in the original device is retained. In the figure below, if the value in E0 is 3, the value in D2 will be moved to D103. When the value in E0 is changed from 3 to 4, the value in D2 will be moved to D104, and the value in D103 will remain unchanged.



*1. The data stored in an index register indicates the offset for the device which the index register modifies. If the system automatically assigns a device to a symbol, the use of an index register to modify the symbol will cause the program to be executed incorrectly because users do not know which device is assigned to the symbol.

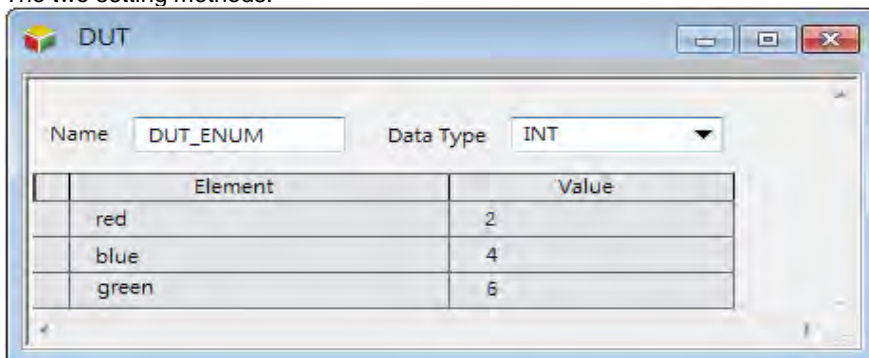
*2. If you want to assign index registers to symbols, you have to specify device addresses and data types.

2.3 Data Type Unit (DUT): ENUM

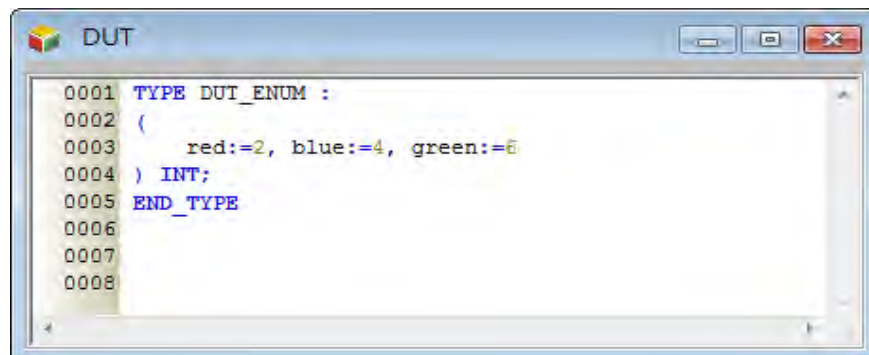
You can also use the data type unit (DUT) for enumeration (ENUM). ENUM is a derivative data type which defines the ENUM symbol with its elements and the associated values. You can specify the initial value to an element, and use one of the enumerated values in the associated elements list. The list defines an ordered set of values in series, starting with the first element and ending with the last one. You can use the same element in different ENUM symbols.

The rules for specifying the elements and the values for an ENUM symbol:

- If elements are not specified, the initial value will be 0, and following 1, and etc.
- When elements are specified with initial values, the initial value of the element will be changed. The enumerated values before the element is defined with an initial value remain unchanged. For example, 0, 1, 2 (not defined), 35(the defined initial value), 36, 37, and etc.
- Another example when the element is defined twice: 0, 1, 2 (not defined), 35(first defined value), 36, 70(second defined value), 71, and etc.
- The two setting methods:



Declare by the setting table



Declare by texts

You can refer to **Appendices** of this manual for a list of Enumerations. For more information about the software operation, refer to *ISPSoft User Manual*.

2.4 Instructions

Instructions used for the AH-Motion products include standard instructions and motion control instructions.

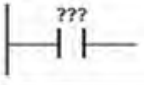
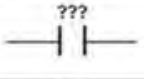
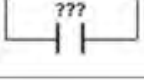
For information of motion control instructions, refer to **AH Motion – Motion Control Instructions Manual**. For a quick reference by alphabetic order, refer to appendix **A.4 Standard instructions (sort by Alphabet)**

2.4.1 Categories of Standard Instructions

Categories	Description
<u>Ladder Instructions</u>	Basic instructions frequently used in ladder diagram programming such as LD, LDP, AND, OR, OUT, and etc
<u>Comparison instructions</u>	Comparisons such as =, <>, >, >=, <, <=, and etc.
<u>Arithmetic instructions</u>	Using binary numbers or binary-coded decimal numbers to add, subtract, multiply, or divide.
<u>Data conversion instructions</u>	Converting the binary-coded decimal number into the binary number, and converting the binary number into the binary-coded decimal number
<u>Data transfer instructions</u>	Transfer the specified data
<u>Jump instructions</u>	The program jumps.
<u>Program execution instructions</u>	Enabling or disabling the interrupt
<u>I/O refreshing instructions</u>	Refreshing the I/O.
<u>Convenience instructions</u>	Instructions which are applied to the counters, the teaching timers, the special timers, and etc.
<u>Logic instructions</u>	Logical operations such as logical addition, logical multiplication, and etc.
<u>Rotation instructions</u>	Rotating/Shifting the specified data
<u>Basic instructions</u>	Timer instructions and counter instructions
<u>Shift instructions</u>	Shifting the specified data
<u>Data processing instructions</u>	16-bit data processing such as decoding and encoding.
<u>Structure creation instructions</u>	Nested loops
<u>Module instructions</u>	Reading the data from the special module and writing the data into the special module
<u>Floating-point number instructions</u>	Floating-point number operations
<u>Real-time clock instructions</u>	Reading/Writing, adding/subtracting and comparing the time
<u>Peripheral instructions</u>	I/O points connected to the peripheral
<u>Communication instructions</u>	Controlling the peripheral though communication
<u>Other instructions</u>	Instructions which are different from those mentioned above
<u>String processing instructions</u>	Conversion between binary/binary-coded decimal numbers and ASCII codes; conversion between binary numbers and strings; conversion between floating-point numbers and strings; string processing
<u>Ethernet instructions</u>	Controlling the Ethernet data exchange
<u>Memory card instructions</u>	Reading the data from the memory card and writing the data into the memory card
<u>Task control instructions</u>	Controlling the task in the program
<u>SFC control instructions</u>	Controlling the SFC operation

2.4.2 List of Standard Instructions

● Columns

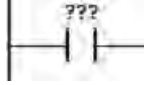
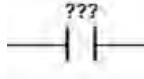
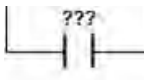
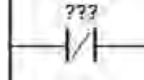
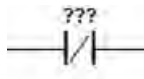
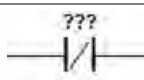
1	2	3	4	5
FB/FC	Instruction	Graphic expression	Description	Operand
FC	LD		Loading contact A	DX, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	AND		Connecting contact A in series	
FC	OR		Connecting contact A in parallel	

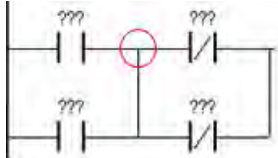
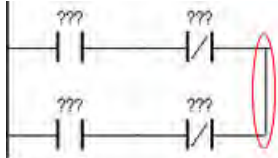
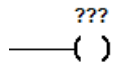

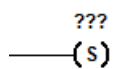

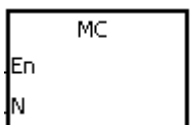

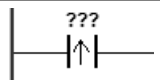

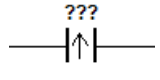
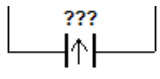
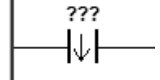

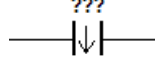
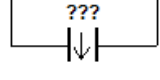

Items provided in the table

1	FB/FC	FB: Function block; FC: Function For further explanations of Program(PROG), Function block(FB), and Function (FC), refer to <i>ISPSOft User Manual</i> .
2	Instruction	The name of the instruction
3	Graphic expression	The graphic expression used in the ladder diagram in the software
4	Description	The function description of the instruction
5	Operand	The operands supported by the instruction

● Ladder instructions

For instruction details, refer to **3.2 Ladder Instructions**.

FB/FC	Instruction	Graphic expression	Description	Operand
FC	LD		Loading contact A	DX, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	AND		Connecting contact A in series	
FC	OR		Connecting contact A in parallel	
FC	LDI		Loading contact B	DX, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	ANI		Connecting contact B in series	
FC	ORI		Connecting contact B in parallel	

FB/FC	Instruction	Graphic expression	Description		Operand
FC	ANB		Connecting the loop blocks in series		—
FC	ORB		Connecting the loop blocks in parallel		—
FC	MPS	—	Storing the data in the stack		—
FC	MRD	—	Reading the data from the stack		—
FC	MPP	—	Popping the data from the stack		—
FC	OUT		Driving the coil	Execution condition: 	DY, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	SET		Keeping the device on	Execution condition: 	DY, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	MC		Setting the master control		N
FC	MCR		Resetting the master control		N
FC	LDP		Starting the rising-edge detection	Execution condition: 	DX, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	PED		Connecting the rising-edge detection in series		
FC	ORP		Connecting the rising-edge detection in parallel		
FC	LDF		Starting the falling-edge detection	Execution condition: 	DX, X, Y, M, S, T, C, HC, D, W, L, SM, and PR
FC	NED		Connecting the falling-edge detection in series		
FC	ANDF		Connecting the falling-edge detection in series		
FC	ONED		Connecting the falling-edge detection in parallel		

FB/FC	Instruction	Graphic expression	Description	Operand	
FC	PLS		Rising-edge output	Execution condition: 	DY, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	PLF		Falling-edge output	Execution condition: 	DY, X, Y, M, S, T, C, HC, D, L, SM, and PR
FC	INV		Inverting the logical operation result	—	—
FC	NOP	—	No operation	—	—
FC	NP		The circuit is rising edge-triggered.	—	—
FC	PN		The circuit is falling edge-triggered.	—	—
FC	FB_NP		The circuit is rising edge-triggered.	S	—
FC	FB_PN		The circuit is falling edge-triggered.	S	—
FC	PSTOP		Stopping executing the PLC program	—	—

● Columns

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	LD=	DLD=			Comparing the values ON: S1 = S2 OFF: S1 ≠ S2
FC	LD<>	DLD<>			Comparing the values ON: S1 ≠ S2 OFF: S1 = S2

Items provided in the table

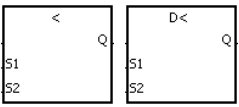
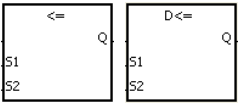
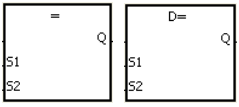
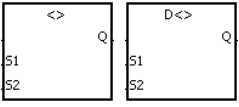
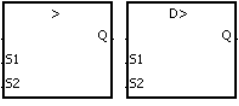
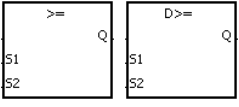
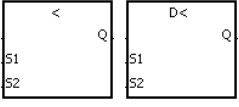
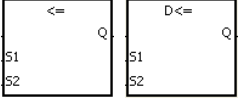
1	FB/FC	FB: Function block; FC: Function
2	Instruction	The name of the instruction (16-bit/32-bit/64-bit) 32-bit: If the 16-bit instruction can be used as a 32-bit instruction, a D is added in front of the 16-bit instruction to form the 32-bit instruction. 64-bit: If the 32-bit floating-point instruction can be used as a 64-bit floating-point number instruction, a D is added in front of the 32-bit floating-point instruction to form a 64-bit floating-point instruction.

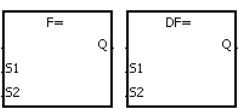
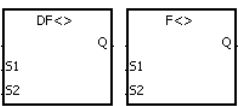
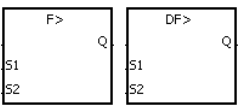
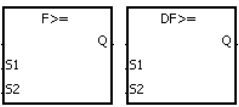
3	Pulse instruction	“√” indicates that the instruction can be used as a pulse instruction, whereas “—” indicates that it can not. If you want to use the pulse instruction, you only need to add a P in back of the instruction.
4	Graphic expression	The graphic expression used in the ladder diagram in the software
5	Description	The function description of the instruction

● **Comparison instructions**

For instruction details, refer to **3.3 Comparison Instructions**.

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	LD=	DLD=	—		Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	LD<>	DLD<>	—		Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	LD>	DLD>	—		Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	LD>=	DLD>=	—		Comparing the values ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	LD<	DLD<	—		Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	LD<=	DLD<=	—		Comparing the values ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	AND=	DAND=	—		Comparing the values in series connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	AND<>	DAND<>	—		Comparing the values in series connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	AND>	DAND>	—		Comparing the values in series connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	AND>=	DAND>=	—		Comparing the values in series connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$

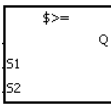
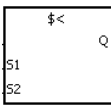
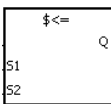
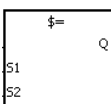
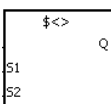
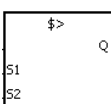
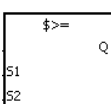
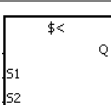
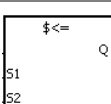
FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	AND<	DAND<	—		Comparing the values in series connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	AND<=	DAND<=	—		Comparing the values in series connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	OR=	DOR=	—		Comparing the values in parallel connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	OR<>	DOR<>	—		Comparing the values in parallel connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	OR>	DOR>	—		Comparing the values in parallel connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	OR>=	DOR>=	—		Comparing the values in parallel connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	OR<	DOR<	—		Comparing the values in parallel connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	OR<=	DOR<=	—		Comparing the values in parallel connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$

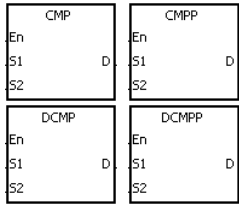
FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	FLD=	DFLD=	—		Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	FLD<>	DFLD<>	—		Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	FLD>	DFLD>	—		Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	FLD>=	DFLD>=	—		Comparing the floating-point numbers ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	FLD<	DFLD<	—		Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	FLD<=	DFLD<=	—		Comparing the floating-point numbers ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	FAND=	DFAND=	—		Comparing the floating-point numbers in series connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	FAND<>	DFAND<>	—		Comparing the floating-point numbers in series connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	FAND>	DFAND>	—		Comparing the floating-point numbers in series connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	FAND>=	DFAND>=	—		Comparing the floating-point numbers in series connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	FAND<	DFAND<	—		Comparing the floating-point numbers in series connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	FAND<=	DFAND<=	—		Comparing the floating-point numbers in series connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	FOR=	DFOR=	—		Comparing the floating-point numbers in parallel connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	FOR<>	DFOR<>	—		Comparing the floating-point numbers in parallel connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	FOR>	DFOR>	—		Comparing the floating-point numbers in parallel connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	FOR>=	DFOR>=	—		Comparing the floating-point numbers in parallel connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	FOR<	DFOR<	—		Comparing the floating-point numbers in parallel connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	FOR<=	DFOR<=	—		Comparing the floating-point numbers in parallel connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	LD\$=	—	—		Comparing the strings ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	LD\$<>	—	—		Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	LD\$>	—	—		Comparing the strings ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	LD\$>=	—	—		Comparing the strings ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	LD\$<	—	—		Comparing the strings ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	LD\$<=	—	—		Comparing the strings ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	AND\$=	—	—		Comparing the strings in series connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	AND\$<>	—	—		Comparing the strings in series connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	AND\$>	—	—		Comparing the strings in series connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$

2

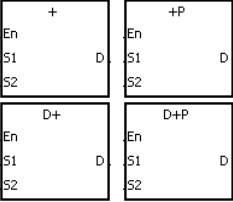
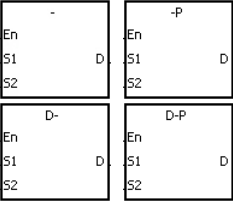
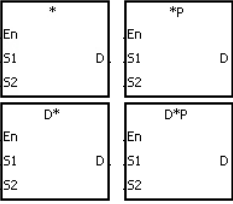
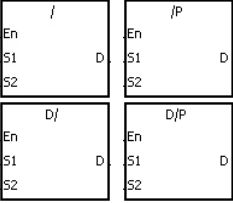
FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	AND\$>=	—	—		Comparing the strings in series connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	AND\$<	—	—		Comparing the strings in series connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	AND\$<=	—	—		Comparing the strings in series connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
FC	OR\$=	—	—		Comparing the strings in parallel connection ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
FC	OR\$<>	—	—		Comparing the strings in parallel connection ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
FC	OR\$>	—	—		Comparing the strings in parallel connection ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
FC	OR\$>=	—	—		Comparing the strings in parallel connection ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
FC	OR\$<	—	—		Comparing the strings in parallel connection ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
FC	OR\$<=	—	—		Comparing the strings in parallel connection ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$

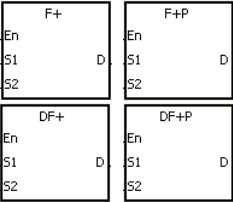
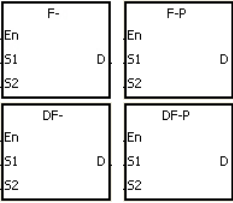

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	CMP	DCMP	✓		Comparing the values

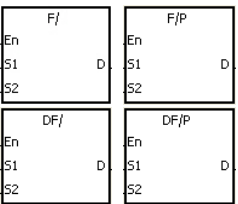
FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	ZCP	DZCP	✓		Zone comparison
FC	—	FCMP	✓		Comparing the floating-point numbers
FC	—	FZCP	✓		Floating-point zone comparison
FC	MCMP	—	✓		Matrix comparison
FC	CMPT=	—	✓		Comparing the tables ON: =
FC	CMPT<>	—	✓		Comparing the tables ON: ≠
FC	CMPT>	—	✓		Comparing the tables ON: >
FC	CMPT>=	—	✓		Comparing the tables ON: ≧
FC	CMPT<	—	✓		Comparing the tables ON: <
FC	CMPT<=	—	✓		Comparing the tables ON: ≦
FC	CHKADR	—	—		Checking the address of the contact type of pointer register

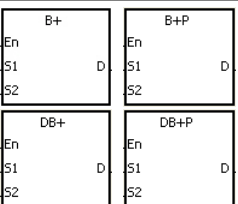
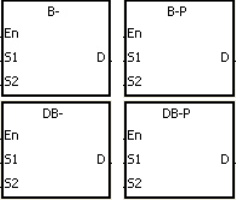
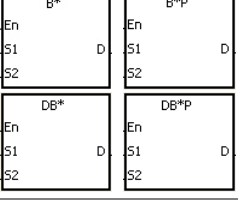
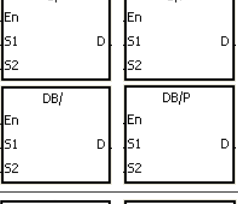
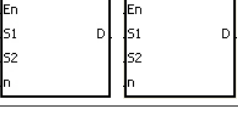
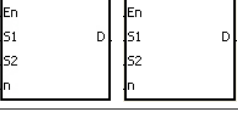
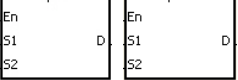
● Arithmetic instructions

For instruction details, refer to **3.4 Arithmetic Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	+	D+	✓		Addition of binary numbers $S1+S2=D$
FC	-	D-	✓		Subtraction of binary numbers $S1-S2=D$
FC	*	D*	✓		Multiplication of binary numbers $S1*S2=D$
FC	/	D/	✓		Division of binary numbers $S1/S2=D$

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	F+	DF+	✓		Addition of floating-point numbers $S1+S2=D$
FC	F-	DF-	✓		Subtraction of floating-point numbers $S1-S2=D$
FC	F*	DF*	✓		Multiplication of floating-point numbers $S1*S2=D$

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	F/	DF/	✓		Division of floating-point numbers $S1/S2=D$

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	B+	DB+	✓		Addition of binary-coded decimal numbers $S1+S2=D$
FC	B-	DB-	✓		Subtraction of binary-coded decimal numbers $S1-S2=D$
FC	B*	DB*	✓		Multiplication of binary-coded decimal numbers $S1*S2=D$
FC	B/	DB/	✓		Division of binary-coded decimal numbers $S1/S2=D$
FC	BK+	—	✓		Addition of binary numbers in blocks
FC	BK-	—	✓		Subtraction of binary numbers in blocks
FC	\$+	—	✓		Linking the strings

2

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	INC	DINC	✓		Adding one to the binary number
FC	DEC	DDEC	✓		Subtracting one from the binary number
FC	MUL16	MUL32	✓		16-bit binary multiplication/ 32-bit binary multiplication
FC	DIV16	DIV32	✓		16-bit binary division/ 32-bit binary division

● Data conversion instructions

For instruction details, refer to 3.5 Data Conversion Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	BCD	DBCD	✓		Converting the binary number into the binary-coded decimal number
FC	BIN	DBIN	✓		Converting the binary-coded decimal number into the binary number
FC	FLT	DFLT	✓		Converting the binary integer into the binary floating-point number
FC	FLTD	DFLTD	✓		Converting the binary integer into the 64-bit floating-point number

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	INT	DINT	✓		Converting the 32-bit floating-point number into the binary integer

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	FINT	DFINT	✓		Converting the 64-bit floating-point number into the binary integer

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	MMOV	—	✓		Converting the 16-bit value into the 32-bit value
FC	RMOV	—	✓		Converting the 32-bit value into the 16-bit value
FC	GRY	DGRY	✓		Converting the binary number into the Gray code
FC	GBIN	DGBIN	✓		Converting the Gray code into the binary number
FC	NEG	DNEG	✓		Two's complement
FC	—	FNEG	✓		Reversing the sign of the 32-bit floating-point number
FC	—	FBCD	✓		Converting the binary floating-point number into the decimal floating-point number
FC	—	FBIN	✓		Converting the decimal floating-point number into the binary floating-point number
FC	BKBCD	—	✓		Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks

2

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	BKBIN	—	✓		Converting the binary numbers in blocks into the binary-coded decimal numbers in blocks
FC	SCAL	—	✓		Scale value operation
FC	SCLP	—	✓		Parameter type of scale value operation
FC	LINE	DLINE	✓		Converting a column of data into a line of data
FC	COLM	DCOLM	✓		Converting a line of data into a column of data

● Data transfer instructions

For instruction details, refer to 3.6 Data Transfer Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	MOV	DMOV	✓		Transferring the data

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	32-bit	64-bit			
FC	—	DFMOV	✓		Transferring the 64-bit floating-point number

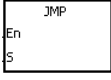

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	\$MOV	—	✓		Transferring the string

FB/FC	Instruction		Pulse Instruction	Graphic expression	Description
	16-bit	32-bit			
FC	CML	DCML	✓		Inverting the data
FC	BMOV	—	✓		Transferring all data
FC	NMOV	DNMOV	✓		Transferring the data to several devices
FC	XCH	DXCH	✓		Exchanging the data
FC	BXCH	—	✓		Exchanging all data
FC	SWAP	DSWAP	✓		Exchange the high byte with the low byte
FC	SMOV	—	✓		Transferring the digits
FC	MOVB	—	✓		Transferring several bits

● **Jump instructions**

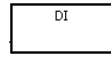
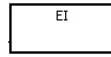
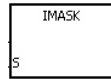
For instruction details, refer to **3.7 Jump Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	CJ	—	✓		Conditional jump

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	JMP	—	—		Unconditional jump
FC	GOEND	—	—		Jumping to END


● Program execution instructions

For instruction details, refer to 3.8 Program Execution Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	DI	—	—		Disabling the interrupt
FC	EI	—	—		Enabling the interrupt
FC	IMASK	—	—		Controlling the interrupt

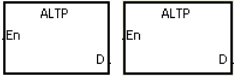

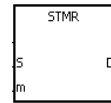
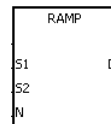
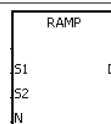
● I/O refreshing instructions

For instruction details, refer to 3.9 I/O Refreshing Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	REF	—	✓		Refreshing the I/O

● Convenience instructions

For instruction details, refer to 3.10 Convenience Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	ALT	—	✓		Alternating between ON and OFF
FC	TTMR	—	—		Teaching timer
FC	STMR	—	—		Special timer
FC	RAMP	—	—		Ramp signal
FC	MTR	—	—		Matrix input

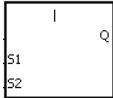

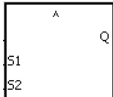

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	ABSD	DABSD	—		Absolute drum sequencer
FC	INCD	—	—		Incremental drum sequencer
FC	—	DPID	—		PID algorithm
FC	—	DPIDE	—		PID algorithm

● **Logic instructions**

For instruction details, refer to **3.11 Logic Instructions**.


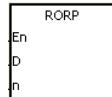

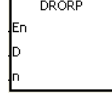

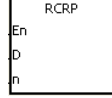
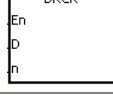

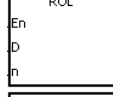
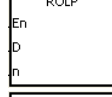
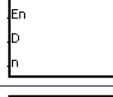
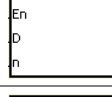

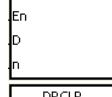


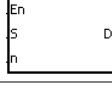
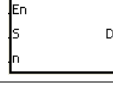
FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	WAND	DAND	✓		Logical AND operation

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	MAND	—	✓		Matrix AND operation
FC	WOR	DOR	✓		Logical OR operation
FC	MOR	—	✓		Matrix OR operation
FC	WXOR	DXOR	✓		Logical exclusive OR operation
FC	MXOR	—	✓		Matrix exclusive OR operation
FC	LD&	DLD&	—		ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$
FC	LD	DLD	—		ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2 = 0$
FC	LD^	DLD^	—		ON: $S_1 \wedge S_2 \neq 0$ OFF: $S_1 \wedge S_2 = 0$
FC	AND&	DAND&	—		ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$
FC	AND	DAND	—		ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2 = 0$
FC	AND^	DAND^	—		ON: $S_1 \wedge S_2 \neq 0$ OFF: $S_1 \wedge S_2 = 0$
FC	OR&	DOR&	—		ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	OR	DOR	—			ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2=0$
FC	OR^	DOR^	—			ON: $S_1^S_2 \neq 0$ OFF: $S_1^S_2=0$

- **Rotation instructions**


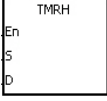
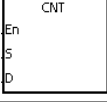
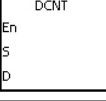
For instruction details, refer to **3.12 Rotation Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	ROR	DROR	✓			Rotating to the right
						
FC	RCR	DRCR	✓			Rotating to the right with the carry flag
						
FC	ROL	DROL	✓			Rotating to the left
						
FC	RCL	DRCL	✓			Rotating to the left with the carry flag
						
FC	MBR	—	✓			Rotating the matrix bits

- **Basic instructions**


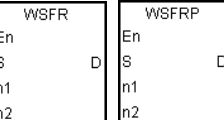
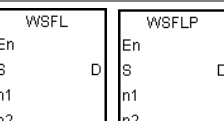
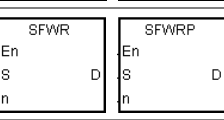

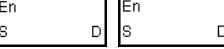
For instruction details, refer to **3.13 Basic Instructions**.



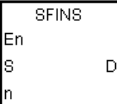

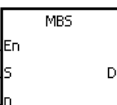
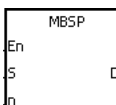
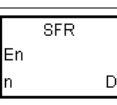
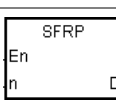
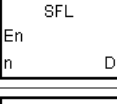

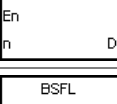
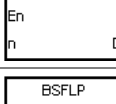
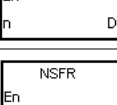
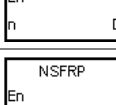




FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	RST	—	—	Device —(R)	Resetting the contact or clearing the register
FC	TMR	—	—		16-bit timer
FC	TMRH	—	—		16-bit timer
FC	CNT	—	—		16-bit counter
FC	—	DCNT	—		32-bit counter

● Shift instructions

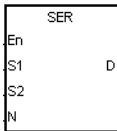

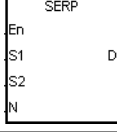
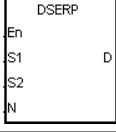
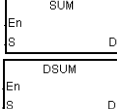
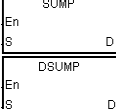
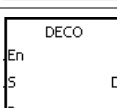
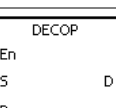
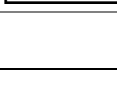
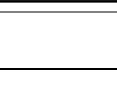
For instruction details, refer to 3.14 Shift Instructions.

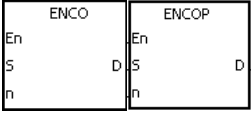
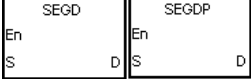
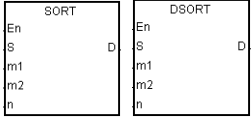


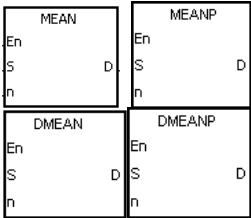

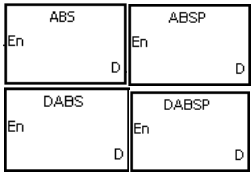

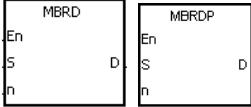

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	SFTR	—	✓		Shifting the states of the devices to the right
FC	SFTL	—	✓		Shifting the states of the devices to the left
FC	WSFR	—	✓		Shifting the data in the word devices to the right
FC	WSFL	—	✓		Shifting the data in the word devices to the left
FC	SFWR	—	✓		Shifting the data and writing it into the word device
FC	SFRD	—	✓		Shifting the data and reading it from the word device
FC	SFPO	—	✓		Reading the latest data from the data list

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	SFDEL	—	✓			Deleting the data from the data list
FC	SFINS	—	✓			Inserting the data into the data list
FC	MBS	—	✓			Shifting the matrix bits
FC	SFR	—	✓			Shifting the values of the bits in the 16-bit registers by n bits to the right
FC	SFL	—	✓			Shifting the values of the bits in the 16-bit registers by n bits to the left
FC	BSFR	—	✓			Shifting the states of the n bit devices by one bit to the right
FC	BSFL	—	✓			Shifting the states of the n bit devices by one bit to the left
FC	NSFR	—	✓			Shifting n registers to the right
FC	NSFL	—	✓			Shifting n registers to the left

- Data processing instructions

For instruction details, refer to 3.15 Data Processing Instructions.



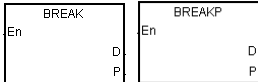
FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	SER	DSER	✓			Searching the data
						
FC	SUM	DSUM	✓			Number of bits whose states are ON
						
FC	DECO	—	✓			Decoder

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	ENCO	—	✓		Encoder
FC	SEGD	—	✓		Seven-segment decoding
FC	SORT	DSORT	—		Sorting the data
FC	ZRST	—	✓		Resetting the zone
FC	BON	DBON	✓		Checking the state of the bit
FC	MEAN	DMEAN	✓		Mean
FC	CCD	—	✓		Sum check
FC	ABS	DABS	✓		Absolute value
FC	MINV	—	✓		Inverting the matrix bits
FC	MBRD	—	✓		Reading the matrix bit
FC	MBWR	—	✓		Writing the matrix bit

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	MBC	—	✓			Counting the bits with the value 0 or 1
FC	DIS	—	✓			Disuniting the 16-bit data
FC	UNI	—	✓			Uniting the 16-bit data
FC	WSUM	DWSUM	✓			Getting the sum
FC	BSET	—	✓			Setting the bit in the word device to ON
FC	BRST	—	✓			Resetting the bit in the word device
FC	BKRST	—	✓			Resetting the specified zone
FC	LIMIT	DLIMIT	✓			Confining the value within the bounds
FC	BAND	DBAND	✓			Deadband control
FC	ZONE	DZONE	✓			Controlling the zone

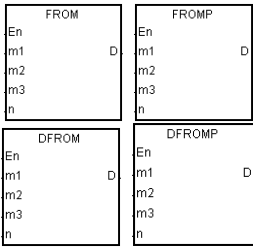
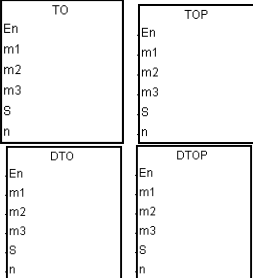
● **Structure creation instructions**

For instruction details, refer to **3.16 Structure Creation Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	FOR	—	—		Start of the nested loop
FC	NEXT	—	—		End of the nested loop
FC	BREAK	—	✓		Terminating the FOR-NEXT loop

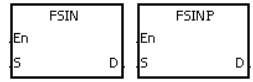
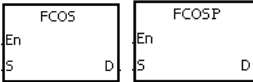
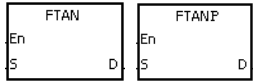
● **Module instructions**

For instruction details, refer to **3.17 Module Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	FROM	DFROM	✓		Reading the data from the control register in the special module
FC	TO	DTO	✓		Writing the data into the control register in the special module

● **Floating-point number instructions**

For instruction details, refer to **3.18 Floating-point Number Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	—	FSIN	✓		Sine of the floating-point number
FC	—	FCOS	✓		Cosine of the floating-point number
FC	—	FTAN	✓		Tangent of the floating-point number

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	—	FASIN	✓			Arcsine of the floating-point number
FC	—	FACOS	✓			Arccosine of the floating-point number
FC	—	FATAN	✓			Arctangent of the floating-point number
FC	—	FSINH	✓			Hyperbolic sine of the floating-point number
FC	—	FCOSH	✓			Hyperbolic cosine of the floating-point number
FC	—	FTANH	✓			Hyperbolic tangent of the floating-point number
FC	—	FRAD	✓			Converting the degree to the radian
FC	—	FDEG	✓			Converting the radian to the degree
FC	SQR	DSQR	✓			Square root of the binary number
FC	—	FSQR	✓			Square root of the floating-point number
FC	—	FEXP	✓			An exponent of the floating-point number
FC	—	FLOG	✓			Logarithm of the floating-point number
FC	—	FLN	✓			Natural logarithm of the binary floating-point number
FC	—	FPOW	✓			A power of the floating-point number
FC	RAND	—	✓			Random number
FC	BSQR	DBSQR	✓			Square root of the binary-coded decimal number

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	—	BSIN	✓		Sine of the binary-coded decimal number
FC	—	BCOS	✓		Cosine of the binary-coded decimal number
FC	—	BTAN	✓		Tangent of the binary-coded decimal number
FC	—	BASIN	✓		Arcsine of the binary-coded decimal number
FC	—	BACOS	✓		Arccosine of the binary-coded decimal number
FC	—	BATAN	✓		Arctangent of the binary-coded decimal number



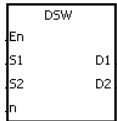
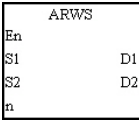

● **Real-time clock instructions**

For instruction details, refer to **3.19 Real-time Clock Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	TRD	—	✓		Reading the time
FC	TWR	—	✓		Writing the time
FC	T+	—	✓		Adding the time
FC	T-	—	✓		Subtracting the time
FC	HOUR	DHOUR	—		Running-time meter
FC	TCMP	—	✓		Comparing the time
FC	TZCP	—	✓		Time zone comparison

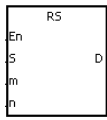

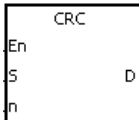
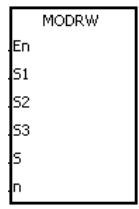

● **Peripheral instructions**

For instruction details, refer to **3.20 Peripheral Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	TKY	DTKY	—		Ten-key keypad
FC	HKY	DHKY	—		Sixteen-key keypad
FC	DSW	—	—		DIP switch
FC	ARWS	—	—		Arrow keys
FC	SEGL	—	—		Seven-segment display with latches

- **Communication instructions**

For instruction details, refer to **3.21 Communication Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	RS	—	—		Transmitting the user-defined communication command
FC	LRC	—	✓		Longitudinal parity check
FC	CRC	—	✓		Cyclic Redundancy Check
FC	MODRW	—	—		Reading/Writing the Modbus data
FC	COMRS	—	—		Sending and receiving communication data

● Other instructions

For instruction details, refer to 3.22 Other Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	WDT	—	✓	 	Watchdog timer
FC	DELAY	—	✓	 	Delaying the execution of the program
FC	GPWM	—	—		General pulse width modulation
FC	TIMCHK	—	—		Checking time
FC	EPUSH	—	✓	 	Storing the contents of the index registers
FC	EPOP	—	✓	 	Reading the data into the index registers

● String processing instructions

For instruction details, refer to 3.23 String Processing Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	BINDA	DBINDA	✓	 	Converting the signed decimal number into the ASCII code
FC	BINHA	DBINHA	✓	 	Converting the binary hexadecimal number into the hexadecimal ASCII code

FB/FC	Instruction		Pulse instruction	Graphic expression		Description																			
	16-bit	32-bit																							
FC	BCDDA	DBCDDA	✓	<table border="1"> <tr><td>BCDDA</td><td>BCDDAP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table> <table border="1"> <tr><td>DBCDDA</td><td>DBCDDAP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	BCDDA	BCDDAP	En	En	S	S	D	D	DBCDDA	DBCDDAP	En	En	S	S	D	D	Converting the binary-coded decimal number into the ASCII code				
BCDDA	BCDDAP																								
En	En																								
S	S																								
D	D																								
DBCDDA	DBCDDAP																								
En	En																								
S	S																								
D	D																								
FC	DABIN	DDABIN	✓	<table border="1"> <tr><td>DABIN</td><td>DABINP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table> <table border="1"> <tr><td>DDABIN</td><td>DDABINP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	DABIN	DABINP	En	En	S	S	D	D	DDABIN	DDABINP	En	En	S	S	D	D	Converting the signed decimal ASCII code into the signed decimal binary number				
DABIN	DABINP																								
En	En																								
S	S																								
D	D																								
DDABIN	DDABINP																								
En	En																								
S	S																								
D	D																								
FC	HABIN	DHABIN	✓	<table border="1"> <tr><td>HABIN</td><td>HABINP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table> <table border="1"> <tr><td>DHABIN</td><td>DHABINP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	HABIN	HABINP	En	En	S	S	D	D	DHABIN	DHABINP	En	En	S	S	D	D	Converting the hexadecimal ASCII code into the hexadecimal binary number				
HABIN	HABINP																								
En	En																								
S	S																								
D	D																								
DHABIN	DHABINP																								
En	En																								
S	S																								
D	D																								
FC	DABCD	DDABCD	✓	<table border="1"> <tr><td>DABCD</td><td>DABCDP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table> <table border="1"> <tr><td>DDABCD</td><td>DDABCDP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	DABCD	DABCDP	En	En	S	S	D	D	DDABCD	DDABCDP	En	En	S	S	D	D	Converting the ASCII code into the binary-coded decimal number				
DABCD	DABCDP																								
En	En																								
S	S																								
D	D																								
DDABCD	DDABCDP																								
En	En																								
S	S																								
D	D																								
FC	\$LEN	—	✓	<table border="1"> <tr><td>\$LEN</td><td>\$LENP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	\$LEN	\$LENP	En	En	S	S	D	D	Calculating the length of the string												
\$LEN	\$LENP																								
En	En																								
S	S																								
D	D																								
FC	\$STR	\$DSTR	✓	<table border="1"> <tr><td>\$STR</td><td>\$STRP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S1</td><td>S1</td></tr> <tr><td>S2</td><td>S2</td></tr> <tr><td>D</td><td>D</td></tr> </table> <table border="1"> <tr><td>D\$STR</td><td>D\$STRP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S1</td><td>S1</td></tr> <tr><td>S2</td><td>S2</td></tr> <tr><td>D</td><td>D</td></tr> </table>	\$STR	\$STRP	En	En	S1	S1	S2	S2	D	D	D\$STR	D\$STRP	En	En	S1	S1	S2	S2	D	D	Converting the binary number into the string
\$STR	\$STRP																								
En	En																								
S1	S1																								
S2	S2																								
D	D																								
D\$STR	D\$STRP																								
En	En																								
S1	S1																								
S2	S2																								
D	D																								
FC	\$VAL	\$DVAL	✓	<table border="1"> <tr><td>\$VAL</td><td>\$VALP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D1</td><td>D1</td></tr> <tr><td>D2</td><td>D2</td></tr> </table> <table border="1"> <tr><td>D\$VAL</td><td>D\$VALP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D1</td><td>D1</td></tr> <tr><td>D2</td><td>D2</td></tr> </table>	\$VAL	\$VALP	En	En	S	S	D1	D1	D2	D2	D\$VAL	D\$VALP	En	En	S	S	D1	D1	D2	D2	Converting the string into the binary number
\$VAL	\$VALP																								
En	En																								
S	S																								
D1	D1																								
D2	D2																								
D\$VAL	D\$VALP																								
En	En																								
S	S																								
D1	D1																								
D2	D2																								
FC	\$FSTR	—	✓	<table border="1"> <tr><td>\$FSTR</td><td>\$FSTRP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S1</td><td>S1</td></tr> <tr><td>S2</td><td>S2</td></tr> <tr><td>D</td><td>D</td></tr> </table>	\$FSTR	\$FSTRP	En	En	S1	S1	S2	S2	D	D	Converting the floating-point number into the string										
\$FSTR	\$FSTRP																								
En	En																								
S1	S1																								
S2	S2																								
D	D																								
FC	\$FVAL	—	✓	<table border="1"> <tr><td>\$FVAL</td><td>\$FVALP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> </table>	\$FVAL	\$FVALP	En	En	S	S	D	D	Converting the string into the floating-point number												
\$FVAL	\$FVALP																								
En	En																								
S	S																								
D	D																								
FC	\$RIGHT	—	✓	<table border="1"> <tr><td>\$RIGHT</td><td>\$RIGHTP</td></tr> <tr><td>En</td><td>En</td></tr> <tr><td>S</td><td>S</td></tr> <tr><td>D</td><td>D</td></tr> <tr><td>n</td><td>n</td></tr> </table>	\$RIGHT	\$RIGHTP	En	En	S	S	D	D	n	n	The retrieve of the characters in the string begins from the right.										
\$RIGHT	\$RIGHTP																								
En	En																								
S	S																								
D	D																								
n	n																								

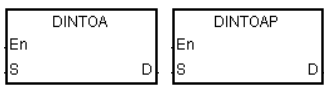
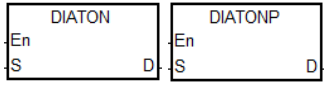
2

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	\$LEFT	—	✓			The retrieve of the characters in the string begins from the left.
FC	\$MIDR	—	✓			Retrieving a part of the string
FC	\$MIDW	—	✓			Replacing a part of the string
FC	\$SER	—	✓			Searching the string
FC	\$RPLC	—	✓			Replacing the characters in the string
FC	\$DEL	—	✓			Deleting the characters in the string
FC	\$CLR	—	✓			Clearing the string
FC	\$INS	—	✓			Inserting the string
FC	\$FMOD	—	✓			Converting the floating-point number into the binary-coded decimal floating-point number
FC	\$FREXP	—	✓			Converting the Binary-coded decimal floating-point number into the floating-point number

● Ethernet instructions

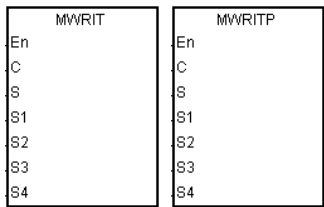
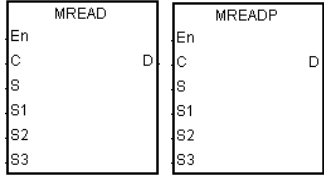
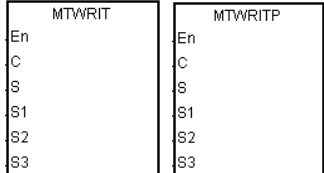
For instruction details, refer to 3.24 Ethernet Instructions.

FB/FC	Instruction		Pulse instruction	Graphic expression		Description
	16-bit	32-bit				
FC	EMDRW	—	✓			Reading/Writing the Modbus TCP data

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	—	DINTOA	✓		Converting the IP address of the integer type into the IP address of the string type
FC	—	DIATON	✓		Converting the IP address of the string type into the IP address of the integer type

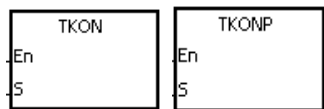
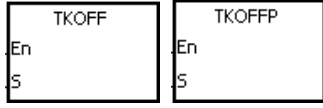
- **Memory card instructions**

For instruction details, refer to **3.25 Memory Card Instructions**.

FB/FC	Instruction code		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	MWRIT	—	✓		Writing the data from the PLC into the memory card
FC	MREAD	—	✓		Reading the data from the memory card into the PLC
FC	MTWRIT	—	✓		Writing the string into the memory card

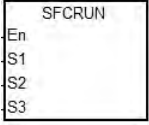
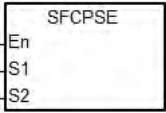

- **Task control instructions**

For instruction details, refer to **3.26 Task Control Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	TKON	—	✓		Enabling the cyclic task
FC	TKOFF	—	✓		Disabling the cyclic task

● **SFC control instructions**

For instruction details, refer to **3.27 SFC Control Instructions**.

FB/FC	Instruction		Pulse instruction	Graphic expression	Description
	16-bit	32-bit			
FC	SFCRUN	—	—		SFC Run
FC	SFCPSE	—	—		SFC Pause
FC	SFCSTP	—	—		SFC Stop

2

Chapter 3 Standard Instructions

Table of Contents

3.1	Applying This Chapter.....	3-3
3.1.1	Items of Standard Instructions	3-3
3.1.2	Restrictions on the Use of Standard Instructions	3-8
3.2	Ladder Instructions	3-10
3.3	Comparison Instructions	3-34
3.4	Arithmetic Instructions	3-70
3.5	Data Conversion Instructions	3-109
3.6	Data Transfer Instructions	3-154
3.7	Jump Instructions	3-180
3.8	Program Execution Instructions	3-189
3.9	I/O Refreshing Instructions	' ! 197
3.10	Convenience Instructions	' ! 200
3.11	Logic Instructions.....	' ! 237
3.12	Rotation Instructions	". ! 261
3.13	Basic Instructions.....	' ! 272
3.14	Shift Instructions	3-280
3.15	Data Processing Instructions.....	3-311
3.16	Structure Creation Instructions	3-360
3.17	Module Instructions	3-368
3.18	Floating Point Instructions	3-374
3.19	Real-time Clock Instructions	3-419
3.20	Peripheral Instructions.....	3-435
3.21	Communication Instructions	3-451
3.22	Other Instructions	3-477
3.23	String Processing Instructions	3-489

3.24	Ethernet Instructions	3-554
3.25	Memory Card Instructions	' !563
3.26	Task Control Instructions	' !576
3.27	SFC Control Instructions	' !579

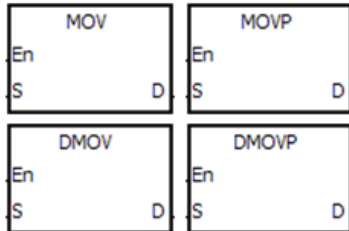
3.1 Applying This Chapter

The columns and notation used for describing standard instructions are explained in this section. In addition, some restrictions on using specific instructions are also specified.

3.1.1 Items of Standard Instructions

1	2	3	4																		
5	FB/FC		Instruction			Operand					Description										
	FC	D*	MOV	P	S, D					Transferring the data											
	Data type	BOOL	WORD	DWORD	LWORD	LINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING						
	S		●	●*				●	●*												
	D		●	●*				●	●*												
6	Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF			
	S	●	●			●	●	●	●	●		●	○		○	○			○		
	D	●	●			●	●	●	●	●		●	○								
	Pulse instruction		16-bit instruction					32-bit instruction													
	AH Motion CPU		AH Motion CPU					AH Motion CPU													

7 ← Graphic expression:



S : Data source

D : Data destination

10

8 ← Explanation:

- When the instruction is executed, the data in S is transferred to D. When the instruction is not executed, the data in D is unchanged.
- Only the data in S which is used in the 32-bit instruction can be the floating-point number.
- Only the 32-bit instructions can use the 32-bit counter.

9 ← Example:

- To transfer the 16-bit data, you should use the instruction MOV.

Items provided in the standard instructions		
1	FB/FC	FB: Function block; FC: Function FC instructions can be called from programs, function blocks, and functions. FB instructions can only be called from programs and function blocks.
2	Instruction	The name of the instruction The name of the instruction varies according to the length of the operands, and may have prefix D* (identifier of 32-bit or 64-bit instruction), F/DF (floating-point instruction), and suffix

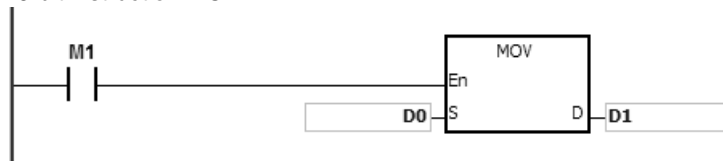
P (pulse instruction).

D*: identifier of 32-bit/64-bit instruction

The values of the operands can be 16-bit values or 32-bit values, and accordingly require 16-bit instructions and 32-bit instructions to operate the data of different data length. To identify 32-bit instructions, a **D*** is added in front of the name of a 16-bit instruction. The rule also applies on indentifying a 64-bit instruction from a 32-bit one.

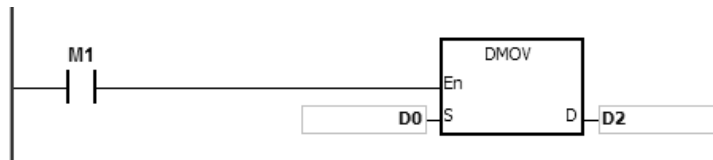
Note: The upper right “*” sign is a reference to the data type of the 32/64-bit instruction. Refer to item 5: **Data Type** for further explanation.

16-bit instruction MOV



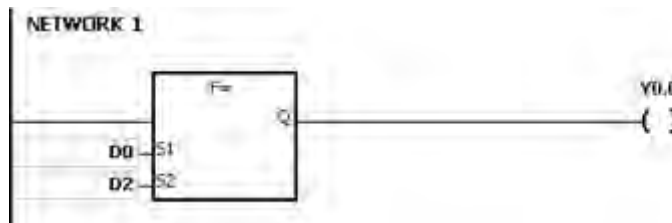
When M1 is ON, the data in D0 is transferred to D1.

32-bit instruction DMOV

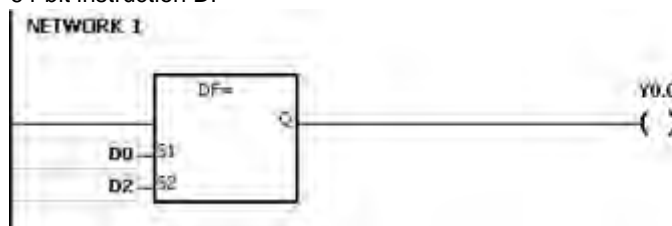


When M1 is ON, the data in (D1, D0) is transferred to (D3, D2).

32-bit instruction F=



64-bit instruction DF=



Note: in some instructions a prefix **W** is added together with **D*** to differentiate between 16-bit and 32-bit instructions.

e.g.

FB/FC	Instruction		
FC	W D*	XOR	P

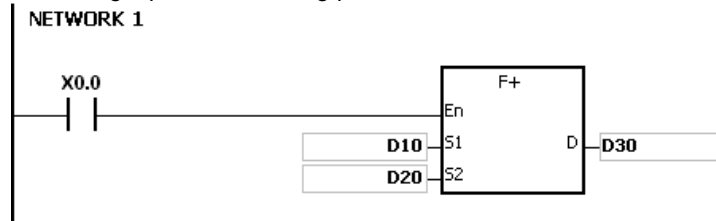
3

F/DF: floating-point instruction

The floating-point instructions can be 32-bit, single-precision floating-point instructions, or 64-bit, double-precision floating-point instructions. For more information on floating-point value, refer to **Chapter 2: Devices, Symbols and Instructions** of this manual.

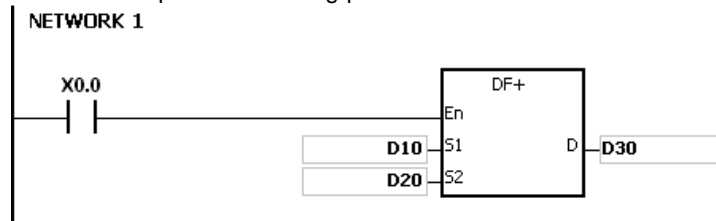
To identify 32-bit single precision floating-point instructions, an **F** is added in front of the name of the 32-bit instruction. To identify 64-bit double precision floating-point instructions, a **DF** is added in front of the name of the 32-bit instruction.

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



When X0.0 is ON, the 32-bit data in (D11, D10) is added to (D21, D20) and stored to (D31, D30).

64-bit double-precision floating-point value instruction DF+



When X0.0 is ON, the data in (D13, D12, D11, D10) is added to (D23, D22, D21, D20) and stored to (D33, D32, D31, D30).

P: pulse execution instruction

The continuous execution and the pulse execution:

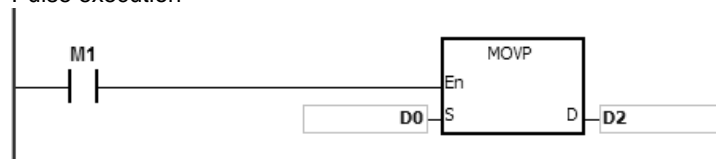
The execution of the instructions can be continuous execution or pulse execution. When a program contains less continuous executed instructions, the time needed to execute the program could be less because of a shorter scan cycle. You can obtain a shorter scan cycle by using pulse instruction appropriately in the program.

When the driving contact of the instruction is ON, the pulse function allows the instruction to execute once in one scan cycle.

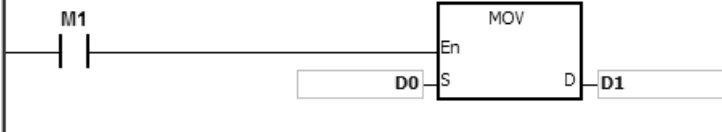
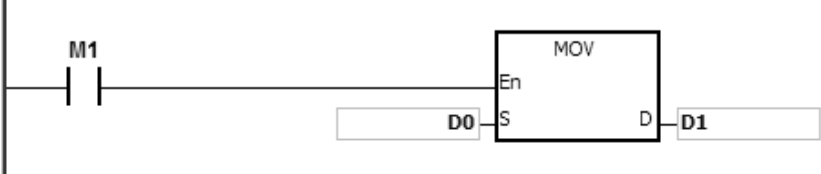
The instruction will execute again only when the driving contact is switched from OFF to ON again.

Examples:

Pulse execution



When M1 is switched from OFF to ON, the instruction MOVP is executed once. The instruction will not be executed any more within the scan cycle. Therefore, it is called the pulse instruction.

		<p>Continuous execution</p>  <p>Whenever M1 is ON, the instruction MOV is executed in every scan cycle. Therefore, the instruction is called the continuous instruction.</p> <p>When the driving contact M1 is OFF, the instruction is not executed, and the value in the destination operand D will not change.</p>						
<p style="text-align: center;">3</p>	<p style="text-align: center;">Operand</p>	<p>The operands supported by the instruction</p> <p>The operands used by the instruction are listed in the operand column. S, D, n, and m are used as the operands according to their functions.</p> <table border="1" data-bbox="469 721 1402 1151"> <tr> <td style="text-align: center;">S</td> <td>Source operand If there is more than one source operand, these source operands are represented by S₁, S₂, and etc.</td> </tr> <tr> <td style="text-align: center;">D</td> <td>Destination operand If there is more than one destination operand, these destination operand is represented by D₁, D₂, and etc.</td> </tr> <tr> <td style="text-align: center;">N</td> <td>Level of the nested program structure</td> </tr> </table> <p>If the operand only can designate the constant K/H or the register, it is represented by m, m₁, m₂, n, n₁, or n₂.</p> <p>Take MOV (transferring the data) for example, it comes with operands S and D which are used to designate the data source and the destination for the data moving operation.</p>  <p>Some standard instructions do not need operands to perform their function. For example, the instructions WDT, and etc. However, most applied instructions consist of several operands required to perform their function.</p> <p>If you want to use an instruction in a function block, and timers (T), 16-bit counters (C), or 32-bit counters (HC/AC) are supported among the operands, you have to use pointer registers of timers (T_Pointer)(TR), 16-bit counters (C_Pointer)(CR), or 32-bit counter (HC/AC_Pointer)(HCR). Refer to Chapter 2: Devices, Symbols and Instructions for more information.</p>	S	Source operand If there is more than one source operand, these source operands are represented by S₁ , S₂ , and etc.	D	Destination operand If there is more than one destination operand, these destination operand is represented by D₁ , D₂ , and etc.	N	Level of the nested program structure
S	Source operand If there is more than one source operand, these source operands are represented by S₁ , S₂ , and etc.							
D	Destination operand If there is more than one destination operand, these destination operand is represented by D₁ , D₂ , and etc.							
N	Level of the nested program structure							
<p style="text-align: center;">4</p>	<p style="text-align: center;">Description</p>	<p>The function description of the instruction</p>						
<p style="text-align: center;">5</p>	<p style="text-align: center;">Data type</p>	<p>The supported data types for this instruction can be identified by a solid circle ●</p> <p>The Array data type is indicated by parenthesis following the operand. For example, operand D(array, 3) with a solid circle ● in the column "Word" indicates an array with 3 words.</p>						

		<table border="1" data-bbox="491 235 774 398"> <tr> <th>Data type</th> <th>BOOL</th> <th>WORD</th> </tr> <tr> <td>S (array, 3)</td> <td></td> <td>●</td> </tr> <tr> <td>D (array, 3)</td> <td></td> <td>●</td> </tr> </table> <p data-bbox="478 414 1476 526">The data type WORD/UINT/INT is only available for 16-bit instruction and DWORD/UDINT/DINT is only available for 32-bit instruction. A star sign is attached for indicating the difference as below:</p> <table border="1" data-bbox="491 526 845 604"> <tr> <th>FB/FC</th> <th colspan="3">Instruction</th> </tr> <tr> <td>FC</td> <td>D*</td> <td>MOV</td> <td>P</td> </tr> </table> <table border="1" data-bbox="491 616 845 784"> <tr> <th>Data type</th> <th>BOOL</th> <th>WORD</th> <th>DWORD</th> </tr> <tr> <td>S</td> <td></td> <td>●</td> <td>●*</td> </tr> <tr> <td>D</td> <td></td> <td>●</td> <td>●*</td> </tr> </table> <p data-bbox="478 795 1476 862">The data type REAL is only available for 32-bit instruction and LWORD/LINT/LREAL is only available for 64-bit instruction. A star sign is attached for indicating the difference as below:</p> <table border="1" data-bbox="491 873 1436 974"> <tr> <th>FB/FC</th> <th colspan="2">Instruction</th> <th>Operand</th> <th>Description</th> </tr> <tr> <td>FC</td> <td>D*</td> <td>FLD✕</td> <td>S₁, S₂</td> <td>Comparing the floatin</td> </tr> </table> <table border="1" data-bbox="491 985 1436 1131"> <tr> <th>Data type</th> <th>BOOL</th> <th>WORD</th> <th>DWORD</th> <th>LWORD</th> <th>UINT</th> <th>UDINT</th> <th>INT</th> <th>DINT</th> <th>LINT</th> <th>REAL</th> <th>LREAL</th> </tr> <tr> <td>S₁, S₂</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>●</td> <td>●*</td> </tr> </table>	Data type	BOOL	WORD	S (array, 3)		●	D (array, 3)		●	FB/FC	Instruction			FC	D*	MOV	P	Data type	BOOL	WORD	DWORD	S		●	●*	D		●	●*	FB/FC	Instruction		Operand	Description	FC	D*	FLD✕	S ₁ , S ₂	Comparing the floatin	Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	S ₁ , S ₂										●	●*
Data type	BOOL	WORD																																																															
S (array, 3)		●																																																															
D (array, 3)		●																																																															
FB/FC	Instruction																																																																
FC	D*	MOV	P																																																														
Data type	BOOL	WORD	DWORD																																																														
S		●	●*																																																														
D		●	●*																																																														
FB/FC	Instruction		Operand	Description																																																													
FC	D*	FLD✕	S ₁ , S ₂	Comparing the floatin																																																													
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL																																																						
S ₁ , S ₂										●	●*																																																						
6	Device	<p data-bbox="478 1176 925 1209">The supported devices for this instruction</p> <p data-bbox="478 1220 1476 1288">The solid circle ● indicates that the device can be modified by an index register, and the hollow circle ○ indicates that the device can not be modified by an index register.</p> <p data-bbox="478 1299 957 1332">The devices that the operands can designate:</p> <p data-bbox="478 1344 925 1377">X: Input relay: X0.0~X127.15 or X0~X127</p> <p data-bbox="478 1388 925 1422">Y: Output relay: Y0.0~Y127.15 or Y0~Y127</p> <p data-bbox="478 1433 766 1467">M: Internal relay: M0~M4095</p> <p data-bbox="478 1478 766 1512">S: Stepping relay: S0~S2047</p> <p data-bbox="478 1523 670 1556">T: Timer: T0~T255</p> <p data-bbox="478 1568 766 1601">C: 16-bit counter: C0~C199</p> <p data-bbox="478 1612 829 1646">HC/AC: 32-bit counter: AC0~AC55</p> <p data-bbox="478 1657 989 1691">D: Data register: D0~D65535 or D0.0~D65535.15</p> <p data-bbox="478 1702 989 1736">L: Link registers: L0~L65535 or L0.0~D65535.15</p> <p data-bbox="478 1747 1476 1780">SM/AM/AR: Special auxiliary flag: SM0~SM2047, AM0~AM16383 or AR0.15~AR65535.15</p> <p data-bbox="478 1792 1149 1825">SR/AR: Special data register: SR0~SR2047 or AR0~AR16383</p> <p data-bbox="478 1836 766 1870">E: Index register: E0~E15</p> <p data-bbox="478 1881 829 1915">PR: Pointer register: PR0~PR15</p> <p data-bbox="526 1926 893 1960"> Pointer register of timer: TR0~TR7</p> <p data-bbox="526 1971 989 2004"> Pointer register of 16-bit counter: CR0~CR7</p> <p data-bbox="526 2016 989 2049"> Pointer register of 32-bit counter: HCR0~HCR7</p> <p data-bbox="478 2060 702 2094">K: Decimal constant</p>																																																															

		<p>16#: Hexadecimal constant</p> <p>"\$": String</p> <p>DF: Single-precision floating-point value; F; double-precision floating-point value: DF</p> <p>Note:</p> <ol style="list-style-type: none"> 1. The data length of a register is usually 16 bits. If you want to store the 32-bit data in the register, you have to designate two consecutive registers. 2. If the operand used in the 32-bit instruction designates D0, the 32-bit data register composed of (D1, D0) is occupied. 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. 3. When the 32-bit counter HC/AC is used as the data register, it only can be designated by the operand used in the 32-bit instruction. <p>You can refer to Chapter 2: Devices, Symbols and Instructions for more information about devices.</p>				
7	Graphic expression	<p>The graphic expression used in the ladder diagram in the software</p> <p>Explanations on the operands are also given</p> <p>En: Enable</p> <p>S: The data source</p> <p>D: The data destination</p> <p>Q: State output of a contact type instruction. Can be used to drive a coil or connect to a contact in series</p>				
8	Explanantion	Detailed explanation of this instruction				
9	Example	Application examples of this instruction				
10	Applicable models	<p>Indicating whether the instrcution can be used as a pulse, 16-bit or 32-bit instruction and its applicable models.</p> <table border="1"> <thead> <tr> <th>Applicable models</th> <th>Model name</th> </tr> </thead> <tbody> <tr> <td>AH Motion CPU</td> <td>AHxxEMC-5A</td> </tr> </tbody> </table>	Applicable models	Model name	AH Motion CPU	AHxxEMC-5A
Applicable models	Model name					
AH Motion CPU	AHxxEMC-5A					

3.1.2 Restrictions on the Use of Standard Instructions

- The instructions which can only be used in the function blocks
CHKADR, FB_NP, FB_PN, NED, ANED, ONED, PED, APED, and OPED
- The instructions which can not be used in the interrupt tasks
GOEND
- The instructions which are not supported in the function blocks
LDP, ANDP, ORP, LDF, ANDF, ORF, PLS, PLF, NP, PN, MC/MCR, GOEND, and all pulse instructions in applied commands

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

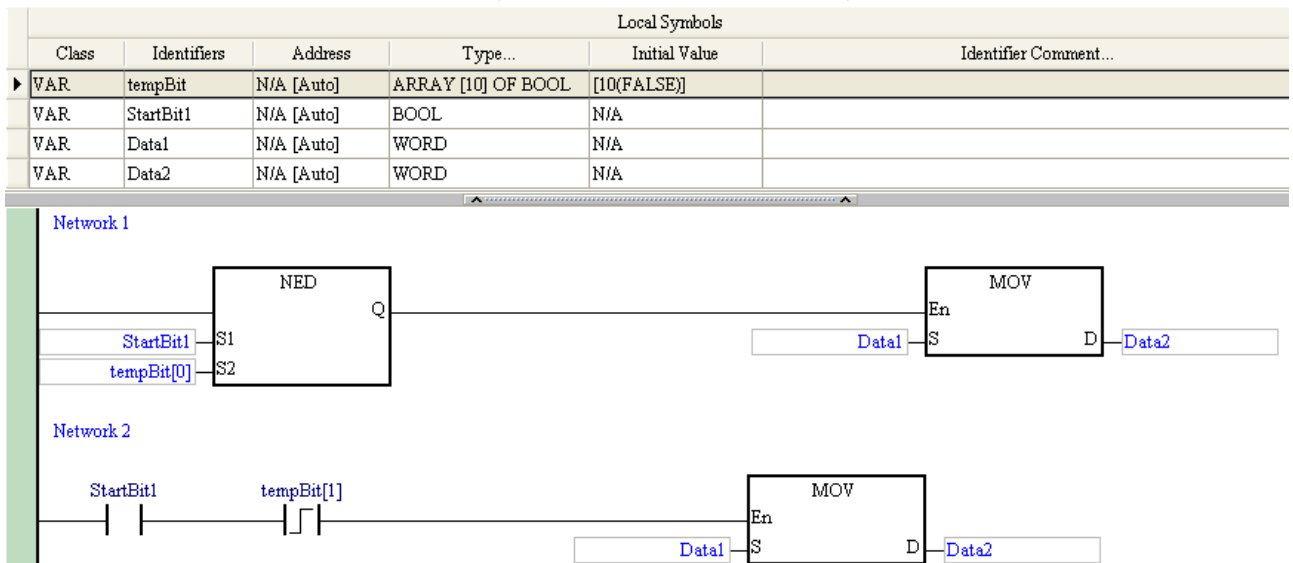
Instruction which can not be used in the function block	Substitute instruction in the function block
LDP/ANDP/ORP	PED/APED/OPED
LDF/ANDF/ORF	NED/ANED/ONED

Instruction which can not be used in the function block	Substitute instruction in the function block
PLS	-
PLF	-
NP	FB_NP
PN	FB_PN
MC	-
MCR	-
All pulse instructions in applied commands	Note 1

Note 1: Pulse instructions can not be used in the function blocks. If you want to apply pulse execution with the function block, you can refer to the following example.

Example:

1. First, declare 10 bits for the variable tempBit[10] which is used in the system.
2. When StartBit1 is switched from ON to OFF, network 1 executes the instruction MOV once. When StartBit1 is switched from OFF to ON, network 2 executes the instruction MOV once. Both network 1 and 2 works for a pulse execution with function blocks.
3. The variable tempBit used in the system can not be used repeatedly.



3.2 Ladder Instructions

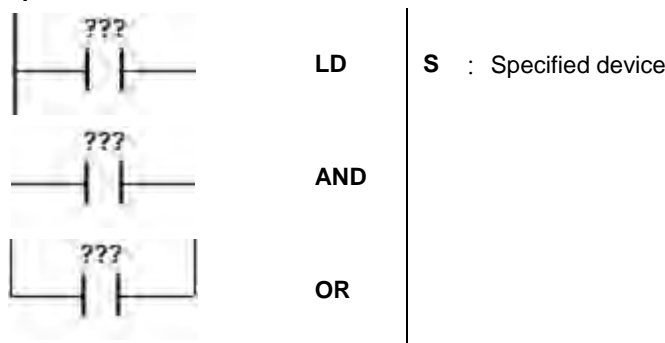
FB/FC	Instruction	Description	Device	Step
FC	<u>LD/AND/OR</u>	Loading contact A/Connecting contact A in series/Connecting contact A in parallel	DX, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>LDI/ANI/ORI</u>	Loading contact B/Connecting contact B in series/Connecting contact B in parallel	DX, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>ANB/ORB</u>	Connecting the loop blocks in series/parallel	–	1
FC	<u>MPS/MRD/MPP</u>	Storing the data in the stack/Reading the data from the stack/Popping the data from the stack	–	1
FC	<u>OUT</u>	Driving the coil	DY, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>SET</u>	Keeping the device on	DY, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>MC/MCR</u>	Setting/Resetting the master control	N	1
FC	<u>LDP/ANDP/ORP</u>	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel	DX, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>LDF/ANDF/ORF</u>	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	DX, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>PED/APED/OPED</u>	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel	X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	5
FC	<u>NED/ANED/ONED</u>	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel	X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	5
FC	<u>PLS</u>	Rising-edge output	DY, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>PLF</u>	Falling-edge output	DY, X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>INV</u>	Inverting the logical operation result	–	1
FC	<u>NOP</u>	No operation	–	1
FC	<u>NP</u>	The circuit is rising edge-triggered.	–	1
FC	<u>PN</u>	The circuit is falling edge-triggered.	–	1
FC	<u>FB_NP</u>	The circuit is rising edge-triggered.	X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>FB_PN</u>	The circuit is falling edge-triggered.	X, Y, M, SM/AM/AR, S, T, C, HC/AC, D, L, and PR	1-2
FC	<u>PSTOP</u>	Stopping executing the program in the PLC	–	1

FB/FC	Instruction	Operand	Description
FC	LD/AND/OR	S	Loading contact A/Connecting contact A in series/Connecting contact A in parallel

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S	•		•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

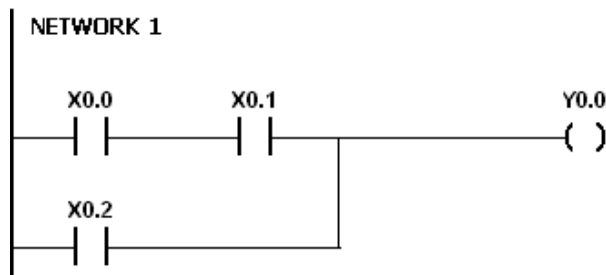


Explanation:

1. The instruction LD applies to contact A which starts from the mother line or contact A which is the start of a contact circuit. It functions to reserve the current contents, and store the contact state which is acquired in the accumulative register.
2. The instruction AND is used to connect contact A in series. It functions to read the state of the contact which is specified to be connected in series, and perform the AND operation with the previous logical operation result. The final result is stored in the accumulative register.
3. The instruction OR is used to connect contact A in parallel. It functions to read the state of the contact which is specified to be connected in parallel, and perform the OR operation with the previous logical operation result. The final result is stored in the accumulative register.

Example:

1. Contact A of X0.0 is loaded, contact A of X0.1 is connected in series, contact A of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are ON, or when X0.2 is ON, Y0.0 is ON.

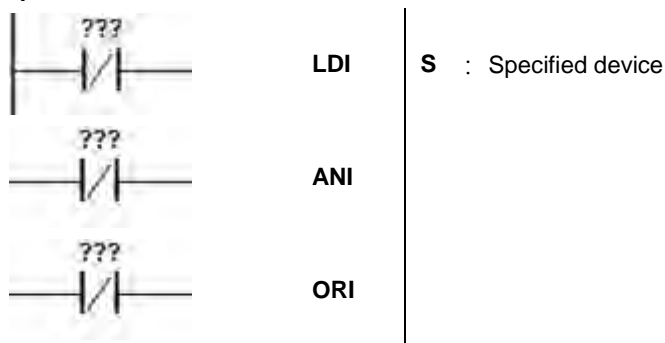


FB/FC	Instruction	Operand	Description
FC	LDI/ANI/ORI	S	Loading contact B/Connecting contact B in series/Connecting contact B in parallel

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S	•		•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

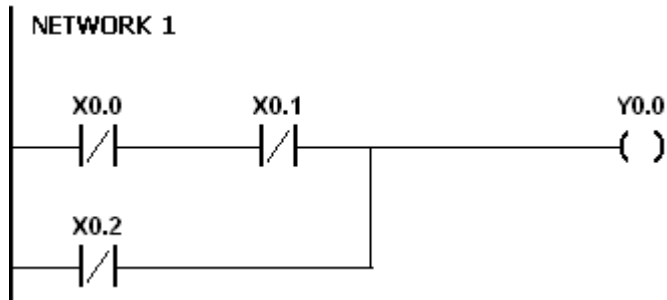


Explanation:

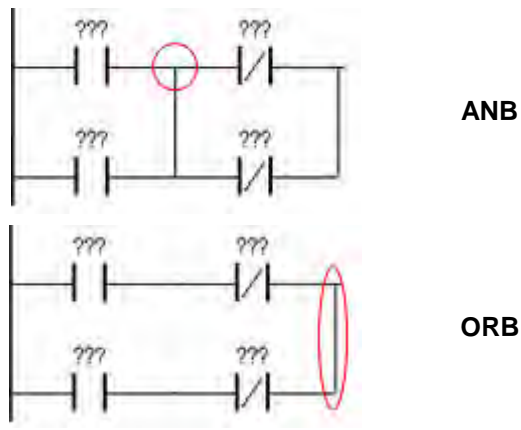
1. The instruction LDI applies to contact B which starts from the mother line or contact B which is the start of a contact circuit. It functions to reserve the current contents, and store the contact state which is acquired in the accumulative register.
2. The instruction ANI is used to connect contact B in series. It functions to read the state of the contact which is specified to be connected in series, and perform the AND operation with the previous logical operation result. The final result is stored in the accumulative register.
3. The instruction ORI is used to connect contact B in parallel. It functions to read the state of the contact which is specified to be connected in parallel, and perform the OR operation with the previous logical operation result. The final result is stored in the accumulative register.

Example:

1. Contact B of X0.0 is loaded, contact B of X0.1 is connected in series, contact B of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are ON, or when X0.2 is ON, Y0.0 is ON.



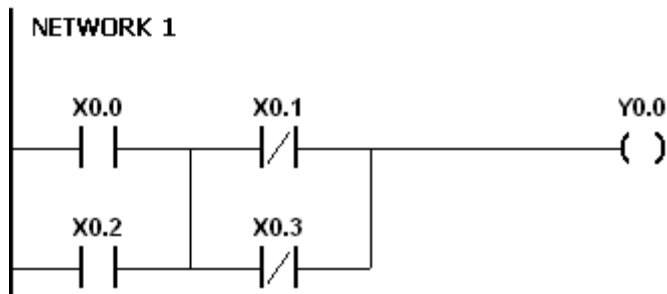
FB/FC	Instruction	Operand	Description
FC	ANB/ORB	-	Connecting the circuit blocks in series/parallel

Graphic Expression:**Explanation:**

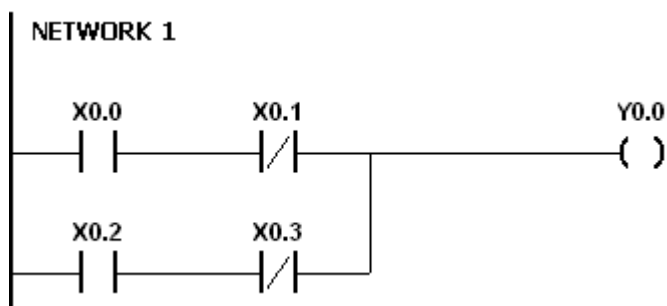
1. The instruction ANB is used to perform the AND operation between the reserved logical operation result and the contents of the accumulative register.
2. The instruction ANB is used to perform the OR operation between the reserved logical operation result and the contents of the accumulative register.

Example:

1. Contact A of X0.0 is loaded, contact A of X0.2 is connected in parallel, contact B of X0.1 is loaded, contact B of X0.3 is connected in parallel, the circuit blocks are connected in series, and the coil Y0.0 is driven.



2. Contact A of X0.0 is loaded, contact B of X0.1 is connected in series, contact A of X0.2 is loaded, contact B of X0.3 is connected in series, the circuit blocks are connected in parallel, and the coil Y0.0 is driven.



FB/FC	Instruction	Operand	Description
FC	MPS/MRD/MPP	-	Storing the data in the stack/Reading the data from the stack/Popping the data from the stack

Explanation:

1. The instruction MPS is used to store the data in the accumulative register in the stack (the value of the stack pointer increases by one).
2. The instruction MRD is used to read the data from the stack and store it in the accumulative register (the value of the stack pointer remains the same).
3. The instruction MPP is used to pop the previous logical operation result from the stack, and store it in the accumulative register (the value of the stack pointer decreases by one).

Example:

1. Contact A of X0 is loaded, and the data in the accumulative register is stored in the stack.
2. Contact A of X1 is connected in series, the coil Y1 is driven, and the data is read from the stack (the value of the stack pointer remains the same).
3. Contact A of X2 is connected in series, the coil M0 is driven, and the previous logical operation result is popped from the stack.

Instruction: Operation:

LD X0 Contact A of X0 is loaded.
MPS The data in the accumulative register is stored in the stack.
AND X1 Contact A of X1 is connected in series.
OUT Y1 The coil Y1 is driven.
MRD The data is read from the stack.
AND X2 Contact A of X2 is connected in series.
OUT M0 The coil M0 is driven.
MPP The previous logical operation result is popped from the stack.
OUT Y2 The coil Y2 is driven.
END The program ends.

Note:

1. The number of MPS instructions must be equal to that of MPP instructions.
2. The instruction MPS can be used at most 31 times.

FB/FC	Instruction	Operand	Description
FC	OUT	S	Driving the coil

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S		•	•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:



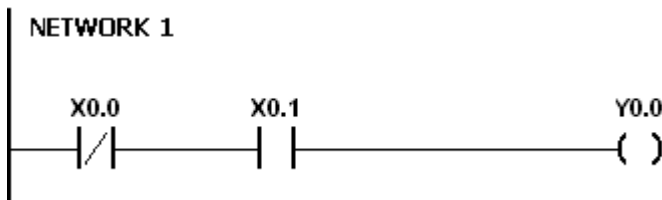
Explanation:

1. The logical operation result prior to the application of the instruction OUT is output into the specified device.
2. 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 X0.0 is loaded, contact A of X0.1 is connected in series, and the coil Y0.0 is driven.
2. When X0.0 is OFF, and X0.1 is ON, Y0.0 is ON.



FB/FC	Instruction	Operand	Description
FC	SET	S	Keeping the device On

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S		•	•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

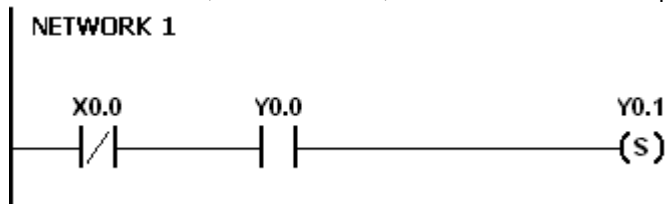


3 Explanation:

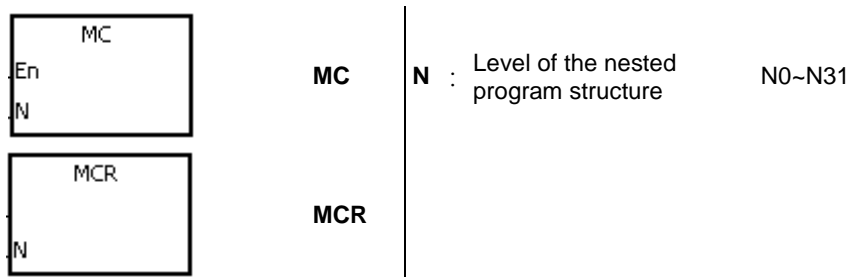
When the instruction SET is driven, the specified device is set to ON. No matter the instruction SET is still driven, the specified device keeps ON. You can set the specified device to OFF by means of the instruction RST.

Example:

1. Contact B of X0.0 is loaded, contact A of Y0.0 is connected in series, and Y0.1 keeps ON.
2. When X0.0 is OFF, and Y0.0 is ON, Y0.1 is ON. Even if the operation result changes, Y0.1 still keeps ON.



FB/FC	Instruction	Operand	Description
FC	MC/MCR	N	Setting/Resetting the master control

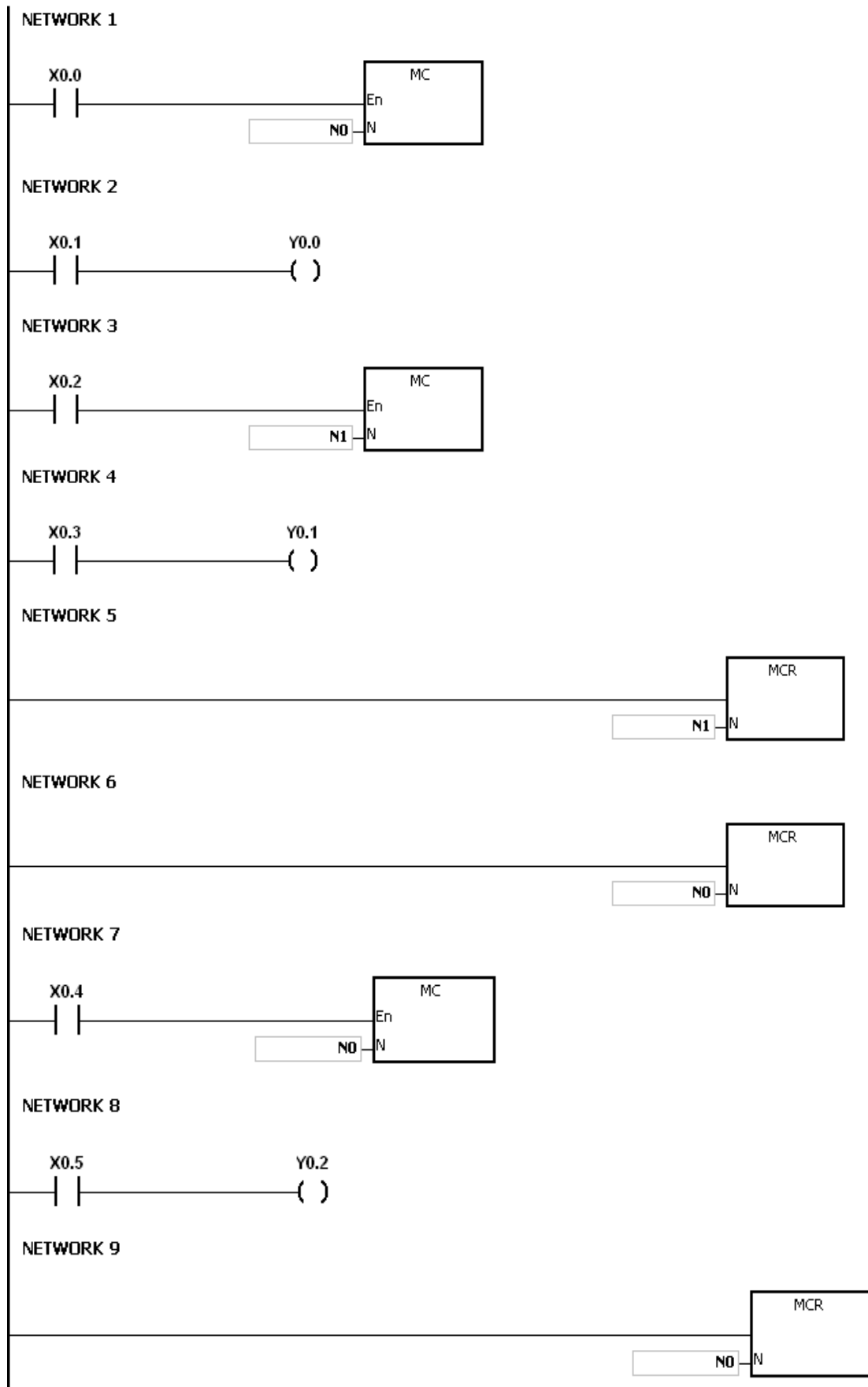
Graphic Expression:**Explanation:**

1. The instruction MCR is used to set the master control. When the instruction MC is executed, the instructions between MC and MCR are executed as usual. When the instruction MC is OFF, the actions of the instructions between MC and MCR are as follows.

Instruction type	Description
General-purpose timer	The timer value is reset to zero. The coil and the contact are OFF.
Timer used 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 remains the same.
Counter	The coil is OFF. The timer value and the state of the contact remains 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. The instruction MCR is used to reset 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). Please refer to the example below.

Example:

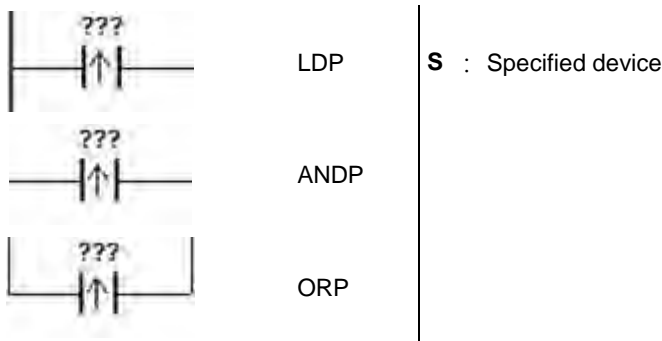


FB/FC	Instruction	Operand	Description
FC	LDP/ANDP/ORP	S	Starting the rising-edge detection/Connecting the rising-edge detection in series/Connecting the rising-edge detection in parallel

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S	•		•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

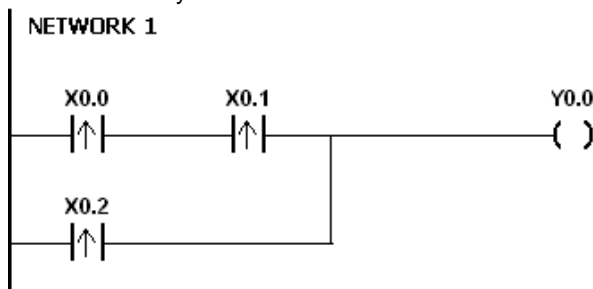


Explanation:

1. The instruction LDP functions to reserve the current contents, and store the rising-edge detection of the contact in the accumulative register.
2. The instruction ANDP is used to connect the rising-edge detection of the contact in series.
3. The instruction ORP is used to connect the rising-edge detection of the contact in parallel.
4. Only when LDP/ANDP/ORP is scanned can the state of the device be gotten, and not until LDP/ANDP/ORP is scanned next time can whether the state of the device changes be judged.
5. Please use the instructions PED, APED, and OPED in the subroutine.

Example:

1. The rising-edge detection of X0.0 starts, the rising-edge detection of X0.1 is connected in series, the rising-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

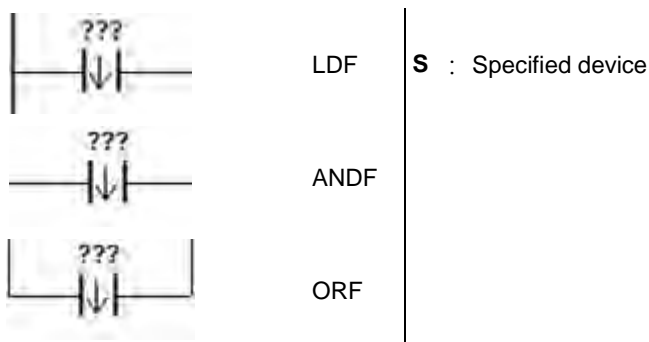


FB/FC	Instruction	Operand	Description
FC	LDF/ANDF/ORF	S	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

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

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S	•		•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

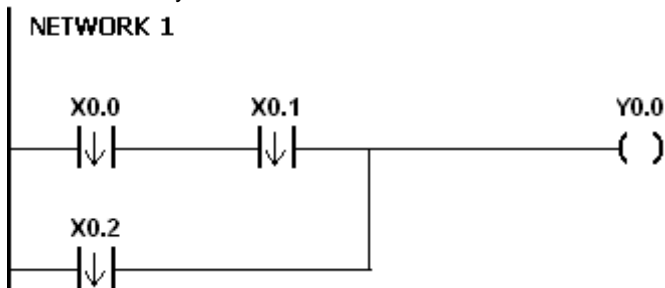


Explanation:

1. The instruction LDF functions to reserve the current contents, and store the falling-edge detection of the contact in the accumulative register.
2. The instruction ANDF is used to connect the falling-edge detection of the contact in series.
3. The instruction ORP is used to connect the falling-edge detection of the contact in parallel.
4. Only when LDF/ANDF/ORF is scanned can the state of the device be gotten, and not until LDF/ANDF/ORF is scanned next time can whether the state of the device changes be judged.
5. Please use the instructions NED, ANED, and ONED in the subroutine.

Example:

1. The falling-edge detection of X0.0 starts, the falling-edge detection of X0.1 is connected in series, the falling-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

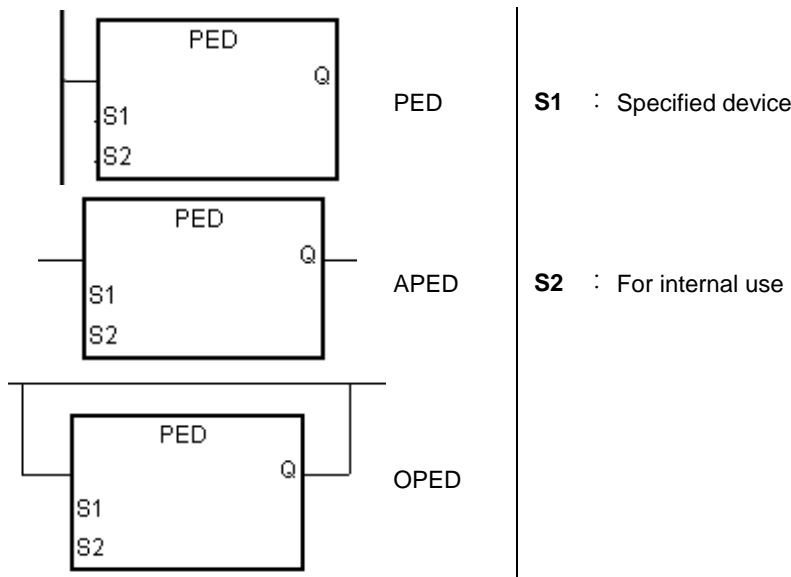


FB/FC	Instruction	Operand	Description
FC	PED/APED/OPED	S1, S2	Starting the rising-edge detection/Connecting the rising edge-detection in series/Connecting the rising-edge detection in parallel

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S1, S2	•													

Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S1, S2			•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:

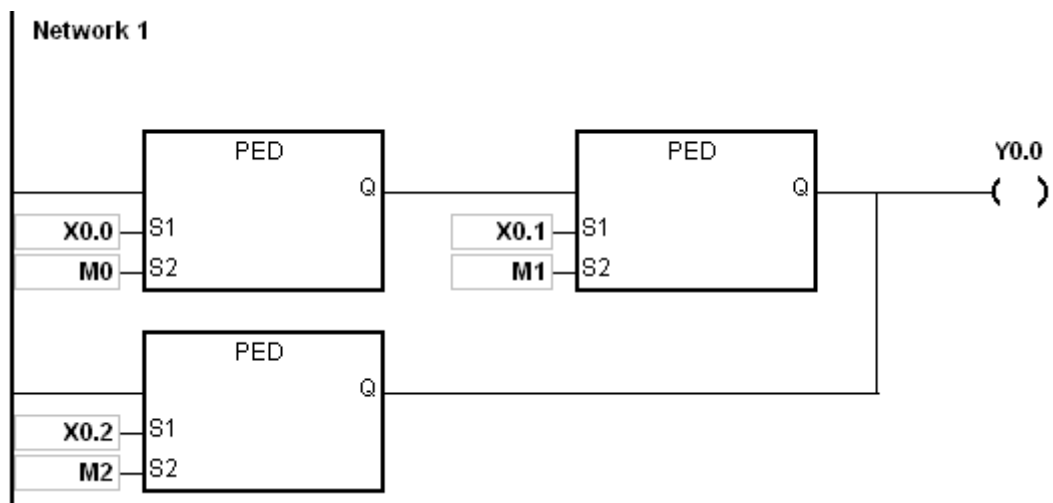


Explanation:

1. PED/APED/OPED corresponds to LDP/ANDP/ORP. The only difference between PED/APED/OPED and LDP/ANDP/ORP lies in that you need to specify the bit device **S2** in which the previous state of the contact is stored when PED/APED/OPED is executed. Please do not use the device **S2** repeatedly in the program. Otherwise, the wrong execution result will appear.
2. The instruction APED is used to connect the rising-edge detection of the contact in series.
3. The instruction OPED is used to connect the rising-edge detection of the contact in parallel.
4. Only when PED/APED/OPED is scanned can the state of the device be gotten, and not until PED/APED/OPED is scanned next time can whether the state of the device changes be judged.
5. PED/APED/OPED only can be used in the function block.

Example:

1. The rising-edge detection of X0.0 starts, the rising-edge detection of X0.1 is connected in series, the rising-edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.

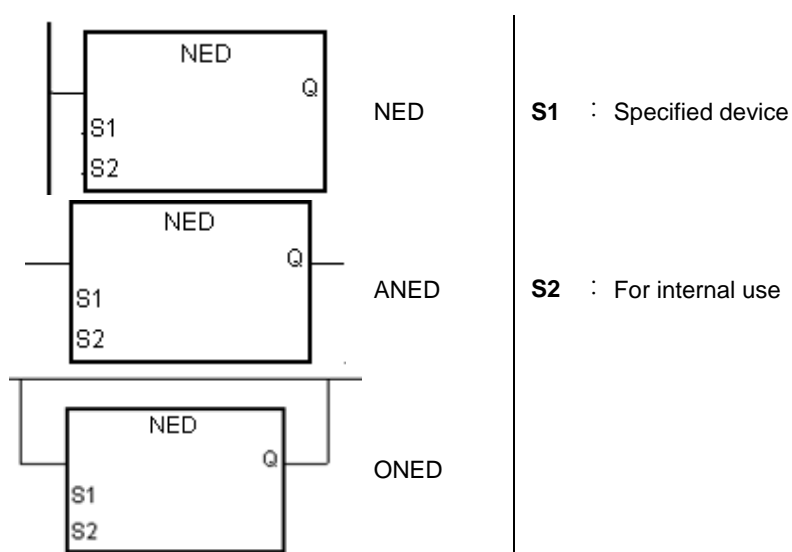


3

FB/FC	Instruction	Operand	Description
FC	NED/ANED/ONED	S1, S2	Starting the falling-edge detection/Connecting the falling-edge detection in series/Connecting the falling-edge detection in parallel

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S1, S2	•													

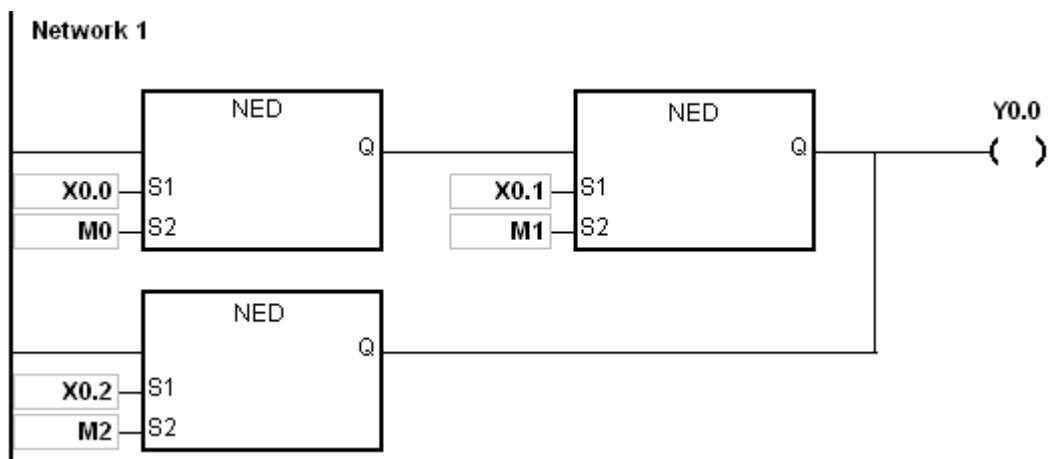
Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR
S1, S2		•	•	•	•	•	•	•	•	•	•	•	•

Graphic Expression:**Explanation:**

1. NED/ANED/ONED corresponds to LDF/ANDF/ORF. The only difference between NED/ANED/ONED and LDF/ANDF/ORF lies in that you need to specify the bit device **S2** in which the previous state of the contact is stored when NED/ANED/ONED is executed. Please do not use the device **S2** repeatedly in the program. Otherwise, the wrong execution result will appear.
2. The instruction ANED is used to connect the falling-edge detection of the contact in series.
3. The instruction ONED is used to connect the falling-edge detection of the contact in parallel.
4. Only when NED/ANED/ONED is scanned can the state of the device be gotten, and not until NED/ANED/ONED is scanned next time can whether the state of the device changes be judged.
5. NED/ANED/ONED only can be used in the function block.

Example:

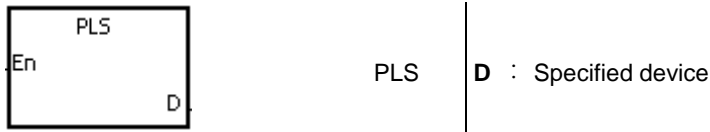
1. The falling -edge detection of X0.0 starts, the falling -edge detection of X0.1 is connected in series, the falling -edge detection of X0.2 is connected in parallel, and the coil Y0.0 is driven.
2. When both X0.0 and X0.1 are switched from OFF to ON, or when X0.2 is switched from OFF to ON, Y0.0 is ON for a scan cycle.



3

FB/FC	Instruction		Operand					Description						
FC	PLS		D					Rising-edge output						
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	•													
Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR	
D		•	•	•	•	•	•	•	•	•	•	•	•	

Graphic Expression:

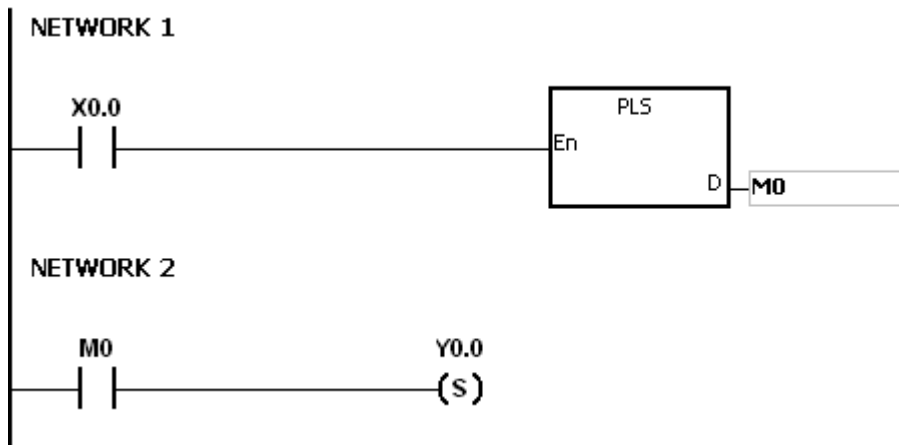


Explanation:

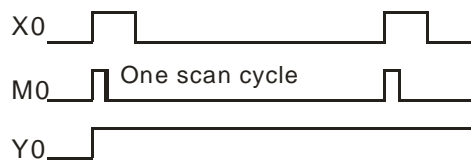
1. When the conditional contact is switched from OFF to ON, the instruction PLS is executed, and the device D sends out a pulse for a scan cycle.
2. Please do not use the instruction PLS in the function block.

Example:

When X0.0 is ON, M0 is ON for a pulse time. When M0 is ON, Y0.0 is set to ON.



Timing diagram:



FB/FC	Instruction		Operand				Description							
FC	PLF		D				Falling-edge output							
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D	•													
Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR	
D		•	•	•	•	•	•	•	•	•	•	•	•	

Graphic Expression:

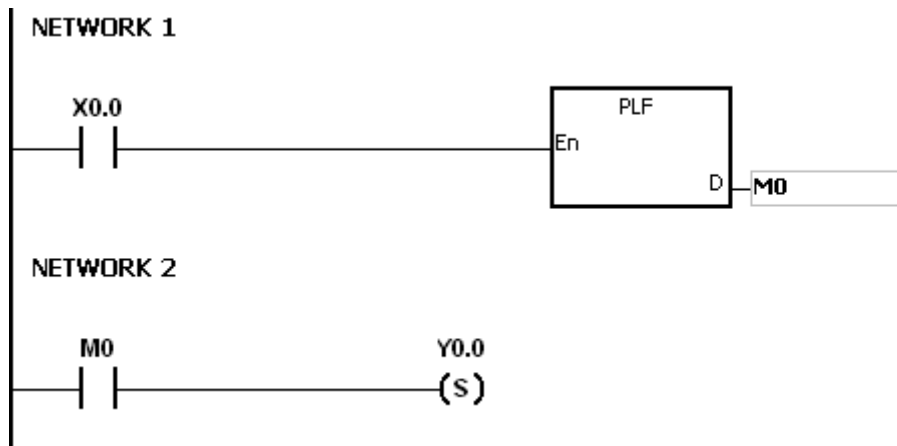


Explanation:

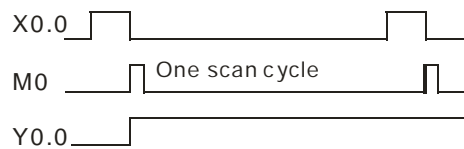
1. When the conditional contact is switched from ON to OFF, the instruction PLF is executed, and the device D sends out a pulse for a scan cycle.
2. Please do not use the instruction PLS in the function block.

Example:

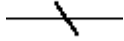
When X0.0 is ON, M0 is ON for a pulse time. When M0 is ON, Y0.0 is set to ON.



Timing chart:



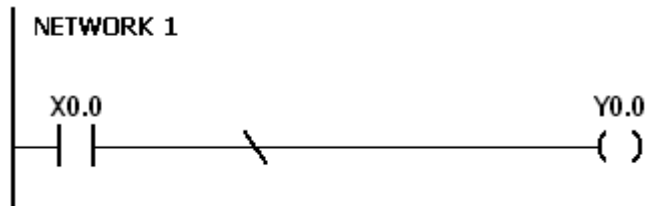
FB/FC	Instruction	Operand	Description
FC	INV	-	Inverting the logical operation result

Graphic Expression:**Explanation:**

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

Example:

When X0.0 is ON, Y0.0 is OFF. When X0.0 is OFF, Y0.0 is ON.



FB/FC	Instruction	Operand	Description
FC	NOP	-	No operation

Explanation:

The instruction NOP does not perform any operation in the program. Therefore, the original logical operation result is retained after NOP is executed. If you want to delete a certain instruction without changing the length of the program, you can use NOP instead.

The instruction NOP only supports the instruction list in ISPSOft. It does not support ladder diagrams.

Example:

The instruction list in ISPSOft:

Instruction:		Operation:
LD	X0.0	Contact A of X0 is loaded.
NOP		No action
OUT	Y1.0	The coil Y1.0 is driven.

FB/FC	Instruction	Operand	Description
FC	NP	-	The circuit is rising edge-triggered

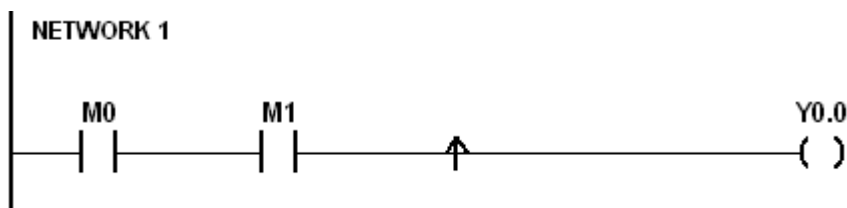
Graphic Expression:



Explanation:

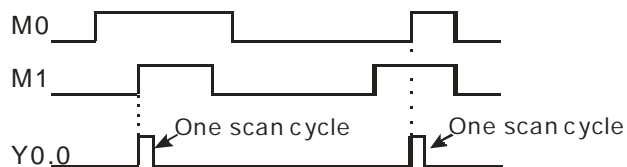
1. When the value in the accumulative register turns from 0 to 1, the instruction NP keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Please use the instruction FB_NP in the function block.

Example:



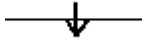
Instruction:	Operation:
LD M0	Contact A of M0 is loaded.
AND M1	Contact A of M1 is connected in series.
NP	The circuit is rising edge-triggered.
OUT Y0.0	The coil Y0.0 is driven.

Timing diagram:



FB/FC	Instruction	Operand	Description
FC	PN	-	The circuit is falling edge-triggered

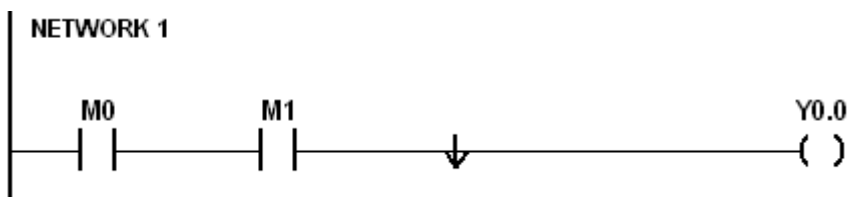
Graphic Expression:



Explanation:

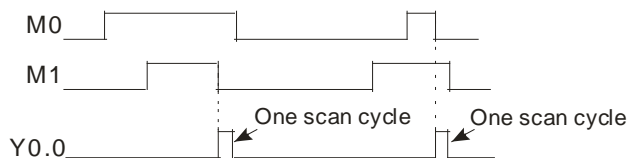
1. When the value in the accumulative register turns from 1 to 0, the instruction PN keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. Please use the instruction FB_PN in the function block.

Example:



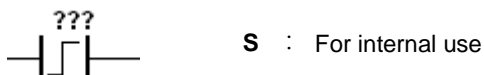
Instruction:	Operation: :
LD M0	Contact A of M0 is loaded.
AND M1	Contact A of M1 is connected in series.
PN	The circuit is falling edge-triggered.
OUT Y0.0	The coil Y0.0 is driven.

Timing diagram:



FB/FC	Instruction		Operand					Description						
FC	FB_NP		S					The circuit is rising edge-triggered.						
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	•													
Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR	
S			•	•	•	•	•	•	•	•	•	•	•	

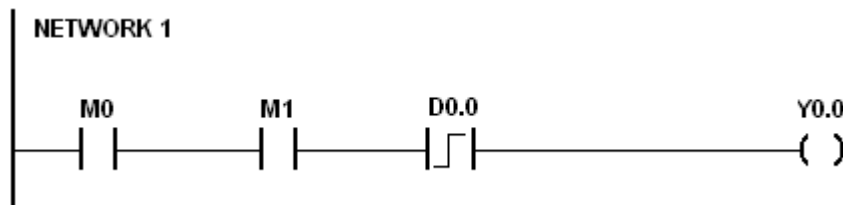
Graphic Expression:



Explanation:

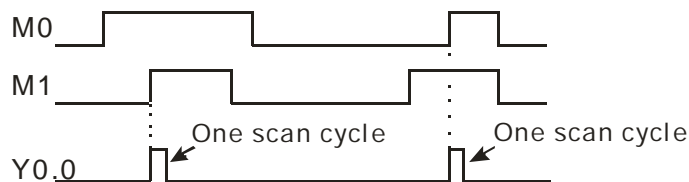
1. When the value in the accumulative register turns from 0 to 1, the instruction FB_NP keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Please do not use **S** repeatedly in the program. Otherwise, the wrong execution result will appear.
3. The instruction FB_NP only can be used in the function block.

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 Y0.0	The coil Y0.0 is driven.

Timing diagram:



FB/FC	Instruction		Operand					Description						
FC	FB_PN		S					The circuit is falling edge-triggered						
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	•													
Device	DX	DY	X	Y	M	SM/AM/AR	S	T	C	HC/AC	D	L	PR	
S			•	•	•	•	•	•	•	•	•	•	•	

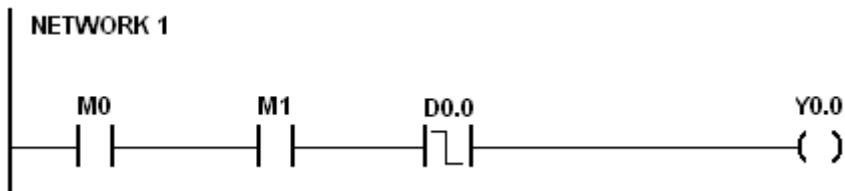
Graphic Expression:



Explanation:

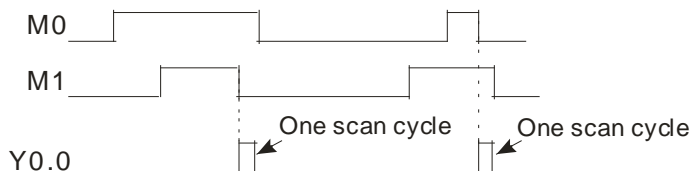
1. When the value in the accumulative register turns from 1 to 0, the instruction FB_PN keeps the value 1 in the accumulative register for a scan cycle. After the second scan cycle is finished, the value in the accumulative register changes to 0.
2. The previous state of the contact is stored in the bit device **S**. Please do not use **S** repeatedly in the program. Otherwise, the wrong execution result will appear.
3. The instruction FB_PN only can be used in the function block.

Example:

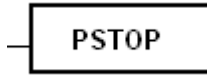


Instruction:		Operation:
LD	M0	Contact A of M0 is loaded.
AND	M1	Contact A of M1 is connected in series.
FB_PN	D0.0	The circuit is falling edge-triggered.
OUT	Y0.0	The coil Y0.0 is driven.

Timing diagram:



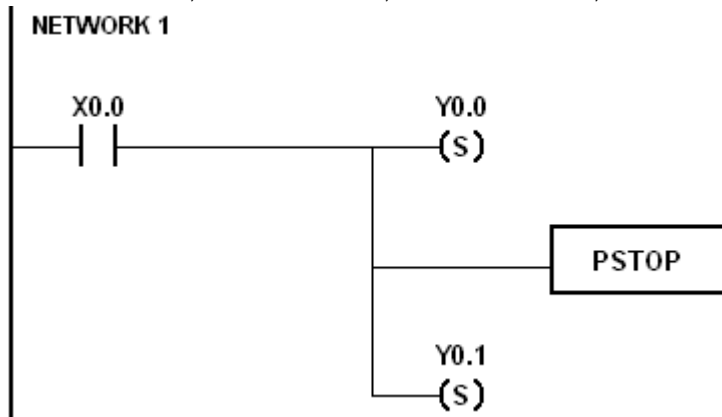
FB/FC	Instruction	Operand	Description
FC	PSTOP	-	Stopping executing the program in the PLC

Graphic Expression:**Explanation:**

When the conditional contact is enabled, the instruction PSTOP stops the execution of the program, and the PLC stops running.

Example:

When X0.0 is ON, Y0.0 is set to ON, Y0.1 remains OFF, and the PLC stops running.



3.3 Comparison Instructions

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>LD=</u>	<u>DLD=</u>	–	–	$S_1 = S_2$	5
FC	<u>LD<></u>	<u>DLD<></u>	–	–	$S_1 \neq S_2$	5
FC	<u>LD></u>	<u>DLD></u>	–	–	$S_1 > S_2$	5
FC	<u>LD>=</u>	<u>DLD>=</u>	–	–	$S_1 \geq S_2$	5
FC	<u>LD<</u>	<u>DLD<</u>	–	–	$S_1 < S_2$	5
FC	<u>LD<=</u>	<u>DLD<=</u>	–	–	$S_1 \leq S_2$	5
FC	<u>AND=</u>	<u>DAND=</u>	–	–	$S_1 = S_2$	5
FC	<u>AND<></u>	<u>DAND<></u>	–	–	$S_1 \neq S_2$	5
FC	<u>AND></u>	<u>DAND></u>	–	–	$S_1 > S_2$	5
FC	<u>AND>=</u>	<u>DAND>=</u>	–	–	$S_1 \geq S_2$	5
FC	<u>AND<</u>	<u>DAND<</u>	–	–	$S_1 < S_2$	5
FC	<u>AND<=</u>	<u>DAND<=</u>	–	–	$S_1 \leq S_2$	5
FC	<u>OR=</u>	<u>DOR=</u>	–	–	$S_1 = S_2$	5
FC	<u>OR<></u>	<u>DOR<></u>	–	–	$S_1 \neq S_2$	5
FC	<u>OR></u>	<u>DOR></u>	–	–	$S_1 > S_2$	5
FC	<u>OR>=</u>	<u>DOR>=</u>	–	–	$S_1 \geq S_2$	5
FC	<u>OR<</u>	<u>DOR<</u>	–	–	$S_1 < S_2$	5
FC	<u>OR<=</u>	<u>DOR<=</u>	–	–	$S_1 \leq S_2$	5
FC	–	<u>FLD=</u>	<u>DFLD=</u>	–	$S_1 = S_2$	5-7
FC	–	<u>FLD<></u>	<u>DFLD<></u>	–	$S_1 \neq S_2$	5-7
FC	–	<u>FLD></u>	<u>DFLD></u>	–	$S_1 > S_2$	5-7
FC	–	<u>FLD>=</u>	<u>DFLD>=</u>	–	$S_1 \geq S_2$	5-7
FC	–	<u>FLD<</u>	<u>DFLD<</u>	–	$S_1 < S_2$	5-7
FC	–	<u>FLD<=</u>	<u>DFLD<=</u>	–	$S_1 \leq S_2$	5-7
FC	–	<u>FAND=</u>	<u>DFAND=</u>	–	$S_1 = S_2$	5-7
FC	–	<u>FAND<></u>	<u>DFAND<></u>	–	$S_1 \neq S_2$	5-7
FC	–	<u>FAND></u>	<u>DFAND></u>	–	$S_1 > S_2$	5-7
FC	–	<u>FAND>=</u>	<u>DFAND>=</u>	–	$S_1 \geq S_2$	5-7
FC	–	<u>FAND<</u>	<u>DFAND<</u>	–	$S_1 < S_2$	5-7
FC	–	<u>FAND<=</u>	<u>DFAND<=</u>	–	$S_1 \leq S_2$	5-7
FC	–	<u>FOR=</u>	<u>DFOR=</u>	–	$S_1 = S_2$	5-7

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	–	<u>FOR<></u>	<u>DFOR<></u>	–	$S_1 \neq S_2$	5-7
FC	–	<u>FOR></u>	<u>DFOR></u>	–	$S_1 > S_2$	5-7
FC	–	<u>FOR>=</u>	<u>DFOR>=</u>	–	$S_1 \geq S_2$	5-7
FC	–	<u>FOR<</u>	<u>DFOR<</u>	–	$S_1 < S_2$	5-7
FC	–	<u>FOR<=</u>	<u>DFOR<=</u>	–	$S_1 \leq S_2$	5-7
FC	<u>LD\$=</u>	–	–	–	$S_1 = S_2$	5-17
FC	<u>LD\$<></u>	–	–	–	$S_1 \neq S_2$	5-17
FC	<u>LD\$></u>	–	–	–	$S_1 > S_2$	5-17
FC	<u>LD\$>=</u>	–	–	–	$S_1 \geq S_2$	5-17
FC	<u>LD\$<</u>	–	–	–	$S_1 < S_2$	5-17
FC	<u>LD\$<=</u>	–	–	–	$S_1 \leq S_2$	5-17
FC	<u>AND\$=</u>	–	–	–	$S_1 = S_2$	5-17
FC	<u>AND\$<></u>	–	–	–	$S_1 \neq S_2$	5-17
FC	<u>AND\$></u>	–	–	–	$S_1 > S_2$	5-17
FC	<u>AND\$>=</u>	–	–	–	$S_1 \geq S_2$	5-17
FC	<u>AND\$<</u>	–	–	–	$S_1 < S_2$	5-17
FC	<u>AND\$<=</u>	–	–	–	$S_1 \leq S_2$	5-17
FC	<u>OR\$=</u>	–	–	–	$S_1 = S_2$	5-17
FC	<u>OR\$<></u>	–	–	–	$S_1 \neq S_2$	5-17
FC	<u>OR\$></u>	–	–	–	$S_1 > S_2$	5-17
FC	<u>OR\$>=</u>	–	–	–	$S_1 \geq S_2$	5-17
FC	<u>OR\$<</u>	–	–	–	$S_1 < S_2$	5-17
FC	<u>OR\$<=</u>	–	–	–	$S_1 \leq S_2$	5-17
FC	<u>CMP</u>	<u>DCMP</u>	–	✓	Comparing the values	7
FC	<u>ZCP</u>	<u>DZCP</u>	–	✓	Zone comparison	9
FC	–	<u>FCMP</u>	–	✓	Comparing the floating-point values	7-9
FC	–	<u>FZCP</u>	–	✓	Floating-point zone comparison	9-12
FC	<u>MCMP</u>	–	–	✓	Matrix comparison	9
FC	<u>CMPT=</u>	–	–	✓	Comparing the tables ON: =	9
FC	<u>CMPT<></u>	–	–	✓	Comparing the tables ON: ≠	9
FC	<u>CMPT></u>	–	–	✓	Comparing the tables ON: >	9

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>CMPT>=</u>	–	–	✓	Comparing the tables ON: \geq	9
FC	<u>CMPT<</u>	–	–	✓	Comparing the tables ON: <	9
FC	<u>CMPT<=</u>	–	–	✓	Comparing the tables ON: \leq	9
FC	<u>CHKADR</u>	–	–	–	Checking the address of the contact type of pointer register	7

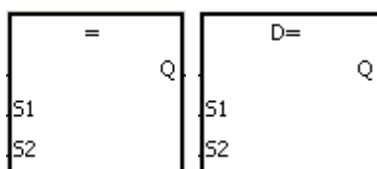
FB/FC	Instruction			Operand	Description
FC	D*	LD※		S ₁ , S ₂	Comparing the values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking LD= and DLD= for example

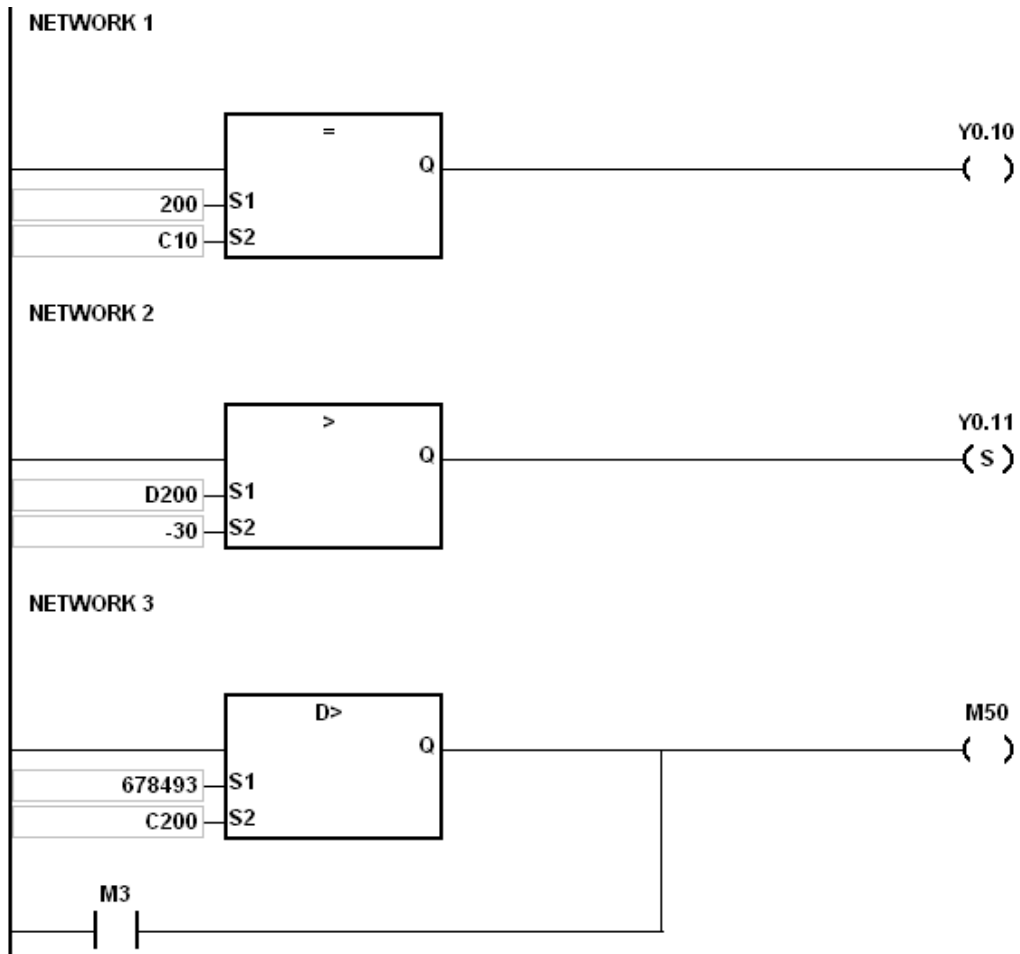
Explanation:

- The instructions are used to compare the value in S₁ with that in S₂. Take the instruction LD= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.
- Only the 32-bit instruction can use the 32-bit counter.

16-bit instruction	32-bit instruction	Comparison operation results	
		ON	OFF
LD =	DLD =	S ₁ = S ₂	S ₁ ≠ S ₂
LD < >	DLD < >	S ₁ ≠ S ₂	S ₁ = S ₂
LD >	DLD >	S ₁ > S ₂	S ₁ ≤ S ₂
LD > =	DLD > =	S ₁ ≥ S ₂	S ₁ < S ₂
LD <	DLD <	S ₁ < S ₂	S ₁ ≥ S ₂
LD < =	DLD < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

- When the value in C10 is equal to 200, Y0.10 is ON.
- When the value in D200 is greater than -30, Y0.11 keeps ON.
- When the value in (C201, C200) is less than 678,493, or when M3 is ON, M50 is ON.



3

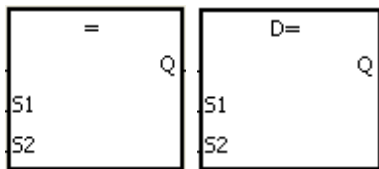
FB/FC	Instruction			Operand	Description
FC	D*	AND※		S ₁ , S ₂	Comparing the values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking AND= and DAND= for example

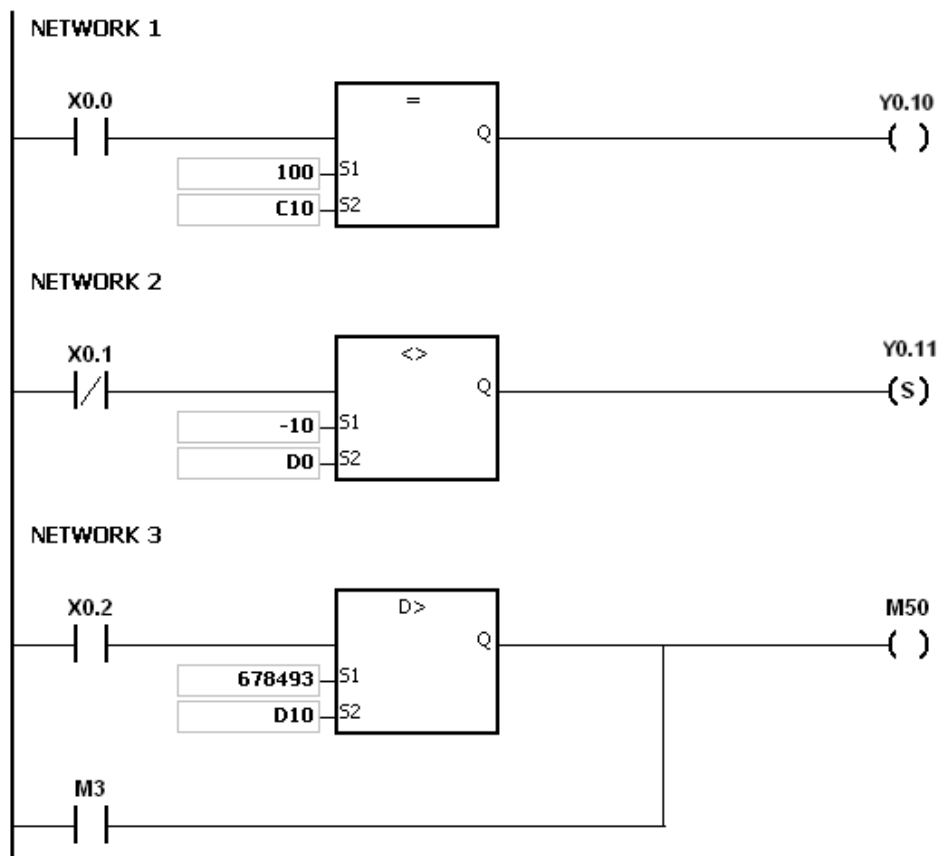
Explanation:

- The instructions are used to compare the value in S₁ with that in S₂. Take the instruction AND= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.
- Only the 32-bit instruction can use the 32-bit counter.

16-bit instruction	32-bit instruction	Comparison operation results	
		ON	OFF
AND =	DAND =	S ₁ = S ₂	S ₁ ≠ S ₂
AND < >	DAND < >	S ₁ ≠ S ₂	S ₁ = S ₂
AND >	DAND >	S ₁ > S ₂	S ₁ ≤ S ₂
AND > =	DAND > =	S ₁ ≥ S ₂	S ₁ < S ₂
AND <	DAND <	S ₁ < S ₂	S ₁ ≥ S ₂
AND < =	DAND < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

- When X0.0 is ON and the current value in C10 is equal to 100, Y0.10 is ON.
- When X0.1 is OFF and the value in D0 is not equal to -10, Y0.11 keeps ON.
- When X0.2 is ON and the value in (D11, D10) is less than 678,493, or when M3 is ON, M50 is ON.



3

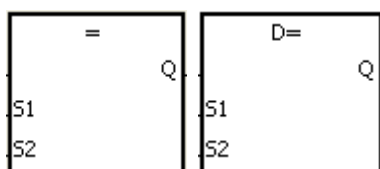
FB/FC	Instruction			Operand	Description
FC	D*	OR※		S ₁ , S ₂	Comparing the values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking OR= and DOR= for example

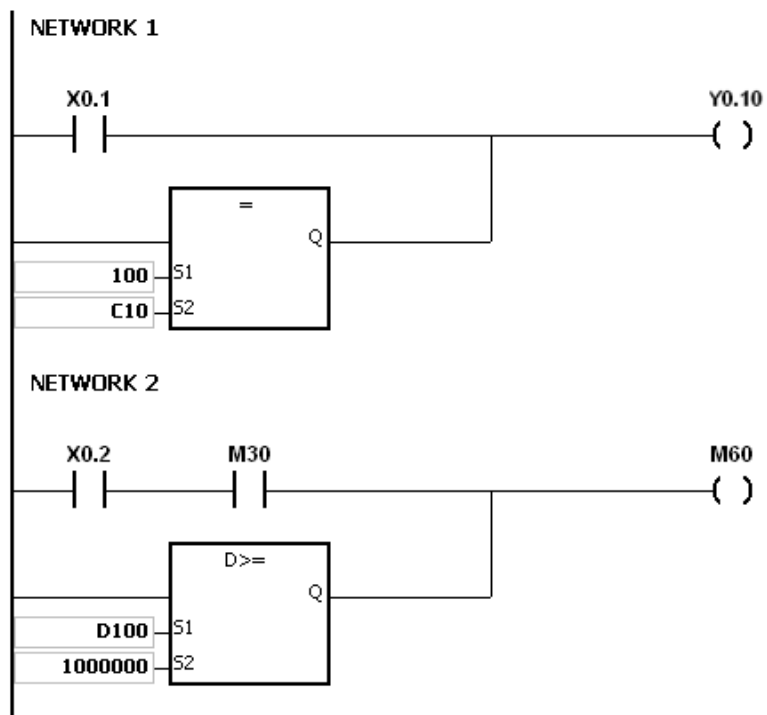
Explanation:

- The instructions are used to compare the value in S₁ with that in S₂. Take the instruction OR= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.
- Only the 32-bit instruction can use the 32-bit counter.

16-bit instruction	32-bit instruction	Comparison operation result	
		ON	OFF
OR =	DOR =	S ₁ = S ₂	S ₁ ≠ S ₂
OR < >	DOR < >	S ₁ ≠ S ₂	S ₁ = S ₂
OR >	DOR >	S ₁ > S ₂	S ₁ ≤ S ₂
OR > =	DOR > =	S ₁ ≥ S ₂	S ₁ < S ₂
OR <	DOR <	S ₁ < S ₂	S ₁ ≥ S ₂
OR < =	DOR < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

- When X0.1 is ON, or when the current value in C10 is equal to 100, Y0.10 is ON.
- When both X0.2 and M30 are ON, or when the value in (D101, D100) is greater than or equal to 1000,000, M60 is ON.



3

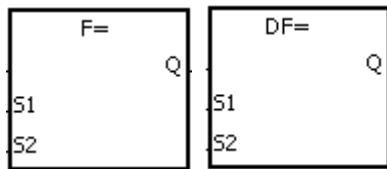
FB/FC	Instruction			Operand				Description				
FC	D*	FLD※		S ₁ , S ₂				Comparing the floating-point values				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking FLD= and DFLD= for example

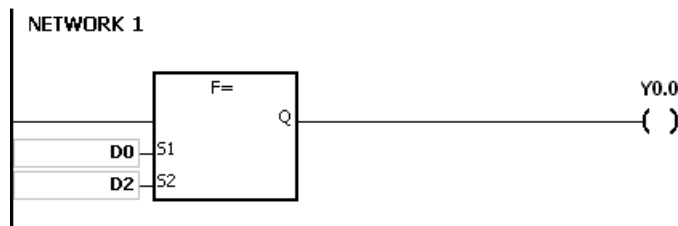
Explanation:

The instructions are used to compare the value in S₁ with that in S₂, and the values compared are floating-point values. Take the instruction FLD= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.

32-bit instruction	64-bit instruction	Comparison operation result	
		ON	OFF
FLD =	DFLD =	S ₁ = S ₂	S ₁ ≠ S ₂
FLD < >	DFLD < >	S ₁ ≠ S ₂	S ₁ = S ₂
FLD >	DFLD >	S ₁ > S ₂	S ₁ ≤ S ₂
FLD > =	DFLD > =	S ₁ ≥ S ₂	S ₁ < S ₂
FLD <	DFLD <	S ₁ < S ₂	S ₁ ≥ S ₂
FLD < =	DFLD < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

Take the instruction FLD = for example. When the value in D0 is equal to that in D2, Y0.0 is ON.



Additional remark:

If the value in S₁ or S₂ is out of the range of values which can be represented by the floating-point values, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

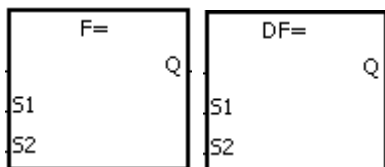
FB/FC	Instruction			Operand			Description									
FC	D*	FAND※		S ₁ , S ₂			Comparing the floating-point values									

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking FAND= and DFAND= for example

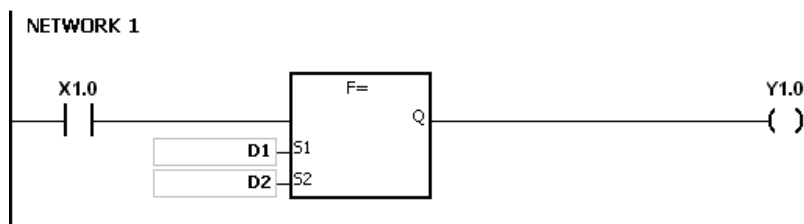
Explanation:

The instructions are used to compare the value in S₁ with that in S₂, and the values compared are floating-point values. Take the instruction FAND= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.

32-bit instruction	64-bit instruction	Comparison operation result	
		ON	OFF
FAND =	DFAND =	S ₁ = S ₂	S ₁ ≠ S ₂
FAND < >	DFAND < >	S ₁ ≠ S ₂	S ₁ = S ₂
FAND >	DFAND >	S ₁ > S ₂	S ₁ ≤ S ₂
FAND > =	DFAND > =	S ₁ ≥ S ₂	S ₁ < S ₂
FAND <	DFAND <	S ₁ < S ₂	S ₁ ≥ S ₂
FAND < =	DFAND < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

Take the instruction FAND = for example. When X1.0 is ON and the value in D1 is equal to that in D2, Y1.0 is ON.



Additional remark:

If the value in S1 or S2 is out of the range of values which can be represented by the floating-point values, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

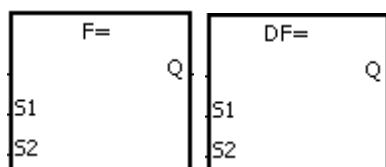
FB/FC	Instruction			Operand			Description		
FC	D*	FOR※		S ₁ , S ₂			Comparing the floating-point values		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



Taking FOR= and DFOR= for example

S₁ : Data source 1

S₂ : Data source 2

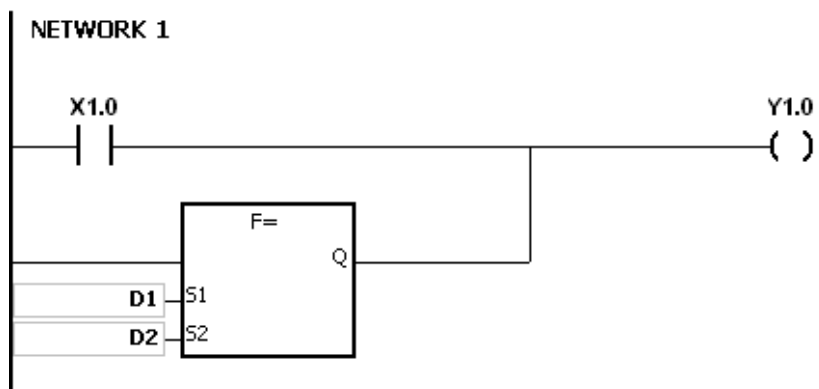
Explanation:

The instructions are used to compare the value in S₁ with that in S₂, and the values compared are floating-point values. Take the instruction FOR= for example. When the comparison result is that the value in S₁ is equal to that in S₂, the condition of the instruction is met. When the comparison result is that the value in S₁ is not equal to that in S₂, the condition of the instruction is not met.

32-bit instruction	64-bit instruction	Comparison operation result	
		ON	OFF
FOR =	DFOR =	S ₁ = S ₂	S ₁ ≠ S ₂
FOR < >	DFOR < >	S ₁ ≠ S ₂	S ₁ = S ₂
FOR >	DFOR >	S ₁ > S ₂	S ₁ ≤ S ₂
FOR > =	DFOR > =	S ₁ ≥ S ₂	S ₁ < S ₂
FOR <	DFOR <	S ₁ < S ₂	S ₁ ≥ S ₂
FOR < =	DFOR < =	S ₁ ≤ S ₂	S ₁ > S ₂

Example:

When X1.0 is ON, or when the value in D1 is equal to that in D2, Y1.0 is ON.

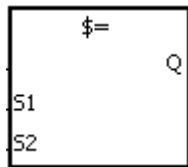


Additional remark:

If the value in **S₁** or **S₂** is out of the range of values which can be represented by the floating-point values, the contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

FB/FC	Instruction				Operand				Description								
FC		LD\$※			S₁, S₂				Comparing the strings								
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING			
S₁, S₂														●			
Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●		●	●		●	○	●			○	
				Pulse instruction				16-bit instruction				32-bit instruction					
				-				AH Motion CPU				-					

Graphic expression:



Taking LD\$= for example

S₁ : Data source 1

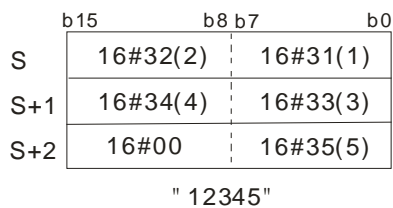
S₂ : Data source 2

Explanation:

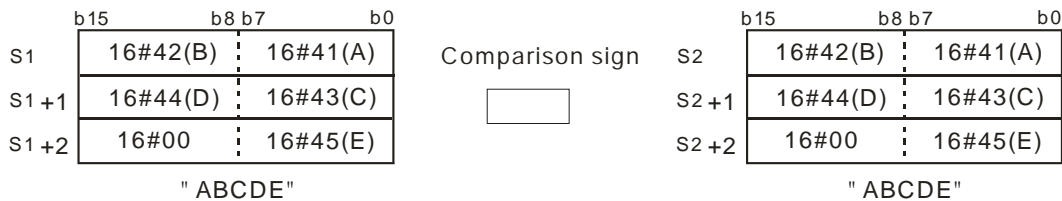
- The instructions are used to compare the data in **S₁** with that in **S₂**, and the data compared is strings. Take the instruction LD\$= for example. When the comparison result is that the data in **S₁** is equal to that in **S₂**, the condition of the contact is met. When the comparison result is that the data in **S₁** is not equal to that in **S₂**, the condition of the contact is not met.

Instruction	Comparison operation result	
	ON	OFF
LD\$ =	S₁ = S₂	S₁ ≠ S₂
LD\$ < >	S₁ ≠ S₂	S₁ = S₂
LD\$ >	S₁ > S₂	S₁ ≤ S₂
LD\$ > =	S₁ ≥ S₂	S₁ < S₂
LD\$ <	S₁ < S₂	S₁ ≥ S₂
LD\$ < =	S₁ ≤ S₂	S₁ > S₂

- Only when the data in **S-S+n** (n indicates the nth device) includes 16#00 can the data be judged as a complete string.

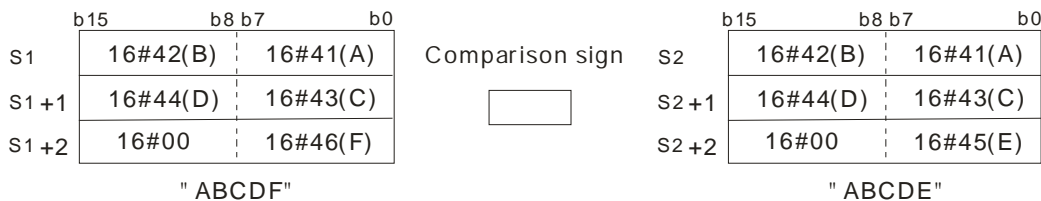


- When two strings are the same, the corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	ON
\$ < >	OFF
\$ >	OFF
\$ > =	ON
\$ <	OFF
\$ < =	ON

4. When the lengths of the strings are the same, but their contents are different, the first different values (ASCII codes) met in the strings are compared. For example, the string in **S₁** is "ABCDF", and the string in **S₁** is "ABCDE". The first different values met in the strings are "F" (16#46) and "E" (16#45). Owing to the fact that 16#46 is greater than 16#45, the string in **S₁** is greater than that in **S₁**. The corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	OFF
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

5. When the lengths of the strings are different, the string whose length is longer is greater than the string whose length is shorter. For example, the string in **S₁** is "1234567", and the string in **S₂** is "99999". Owing to the fact that the string in **S₁** is composed of 7 characters, and the string in **S₂** is composed of 5 characters, the string in **S₁** is greater than the string in **S₂**. The corresponding comparison operation results of the instructions are listed below.

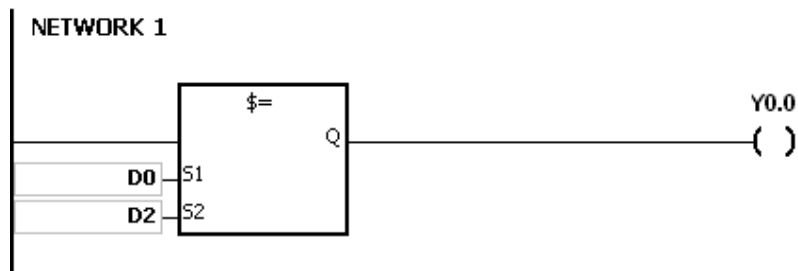


Comparison symbol	Comparison operation result
\$ =	OFF

Comparison symbol	Comparison operation result
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

Example:

When the string starting with the data in D0 is equal to the string starting with D2, Y0.0 is ON.

**Additional remark:**

If 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.

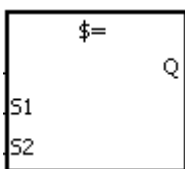
FB/FC	Instruction		Operand	Description
FC		AND\$※	S₁, S₂	Comparing the strings

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂														●

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁, S₂	●	●			●	●		●			●	○	●			○	

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	

Graphic expression:



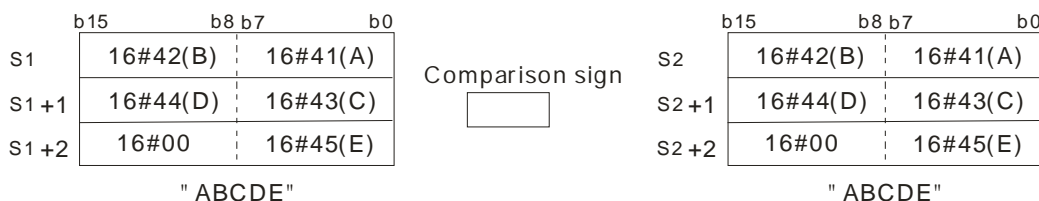
S₁ : Data source 1

S₂ : Data source 2

Taking AND\$= for example

Explanation:

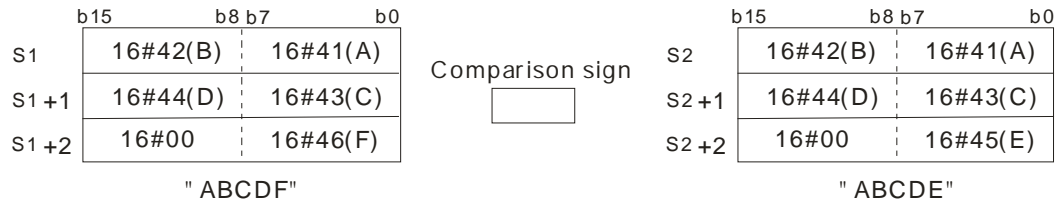
- The instructions are used to compare the data in **S₁** with that in **S₂**, and the data compared is strings. Take the instruction AND\$= for example. When the comparison result is that the data in **S₁** is equal to that in **S₂**, the condition of the contact is met. When the comparison result is that the data in **S₁** is not equal to that in **S₂**, the condition of the contact is not met.
- Only when the data in **S~S+n** (n indicates the nth device) includes 16#00 can the data be judged as a complete string.
- When the strings are completely the same, the corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	ON
\$ < >	OFF
\$ >	OFF
\$ > =	ON
\$ <	OFF
\$ < =	ON

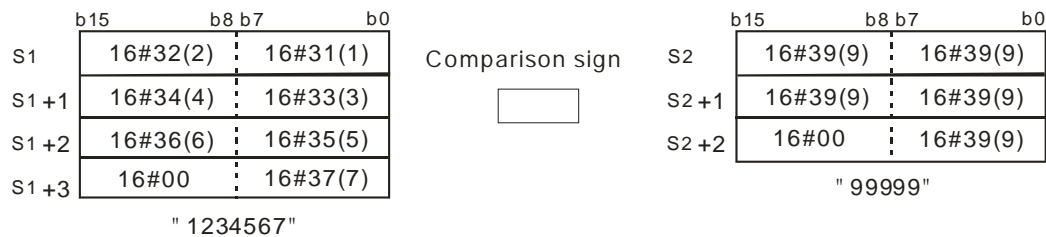
- When the lengths of the strings are the same, but their contents are different, the first different values (ASCII codes) met in the strings are compared. For example, the string in **S₁** is "ABCDF", and the string in **S₁** is

"ABCDE". The first different values met in the strings are "F" (16#46) and "E" (16#45). Owing to the fact that 16#46 is greater than 16#45, the string in **S₁** is greater than that in **S₁**. The corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	OFF
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

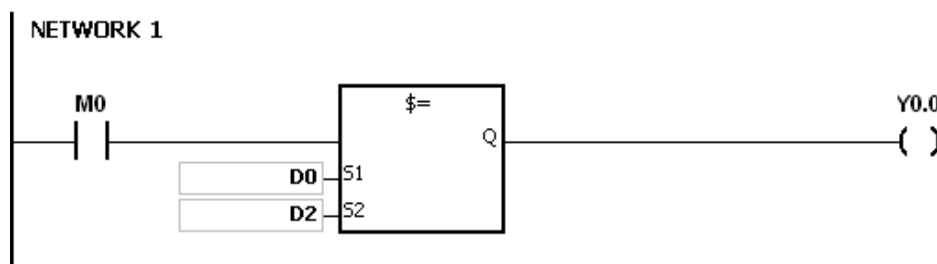
- When the lengths of the strings are different, the string whose length is longer is greater than the string whose length is shorter. For example, the string in **S₁** is "1234567", and the string in **S₂** is "99999". Owing to the fact that the string in **S₁** is composed of 7 characters, and the string in **S₂** is composed of 5 characters, the string in **S₁** is greater than the string in **S₂**. The corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	OFF
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

Example:

When M0 is ON and the string starting with the data in D0 is equal to the string starting with D2, Y0.0 is ON.



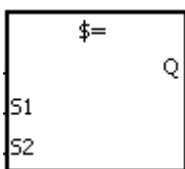
FB/FC	Instruction		Operand	Description
FC		OR\$※	S₁, S₂	Comparing the strings

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂														●

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁, S₂	●	●			●	●		●	●		●	○	●			○	

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



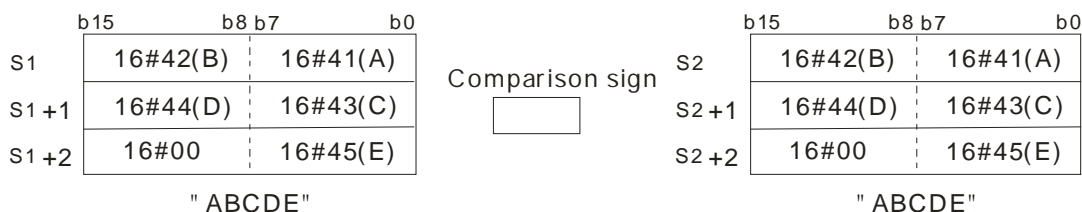
S₁ : Data source 1

S₂ : Data source 2

Taking OR\$= for example

Explanation:

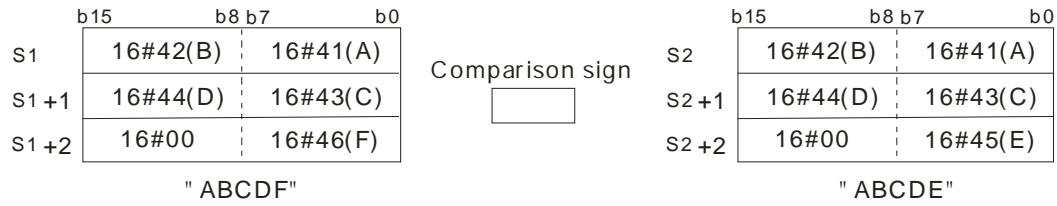
- The instructions are used to compare the data in **S₁** with that in **S₂**, and the data compared is strings. Take the instruction OR\$= for example. When the comparison result is that the data in **S₁** is equal to that in **S₂**, the condition of the contact is met. When the comparison result is that the data in **S₁** is not equal to that in **S₂**, the condition of the contact is not met.
- Only when the data in **S~S+n** (n indicates the nth device) includes 16#00 can the data be judged as a complete string.
- When the strings are completely the same, the corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	ON
\$ < >	OFF
\$ >	OFF
\$ > =	ON
\$ <	OFF
\$ < =	ON

- When the lengths of the strings are the same, but their contents are different, the first different values (ASCII codes) met in the strings are compared. For example, the string in **S₁** is "ABCDF", and the string in **S₁** is

"ABCDE". The first different values met in the strings are "F" (16#46) and "E" (16#45). Owing to the fact that 16#46 is greater than 16#45, the string in **S₁** is greater than that in **S₁**. The corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	OFF
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

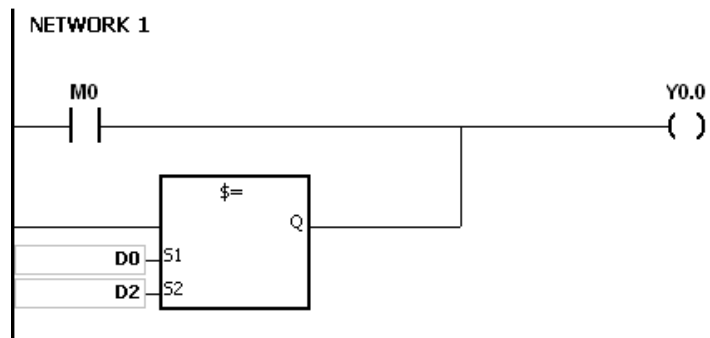
- When the lengths of the strings are different, the string whose length is longer is greater than the string whose length is shorter. For example, the string in **S₁** is "1234567", and the string in **S₂** is "99999". Owing to the fact that the string in **S₁** is composed of 7 characters, and the string in **S₂** is composed of 5 characters, the string in **S₁** is greater than the string in **S₂**. The corresponding comparison operation results of the instructions are listed below.



Comparison symbol	Comparison operation result
\$ =	OFF
\$ < >	ON
\$ >	ON
\$ > =	ON
\$ <	OFF
\$ < =	OFF

Example:

When M0 is ON, or when the string starting with the data in D0 is equal to the string starting with D2, Y0.0 is ON.



3

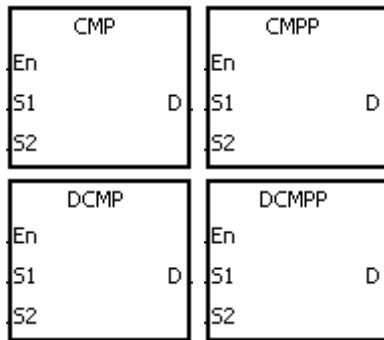
FB/FC	Instruction			Operand	Description
FC	D*	CMP	P	S ₁ , S ₂ , D	Comparing the values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D(array, 3)	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Comparison value 1

S₂ : Comparison value 2

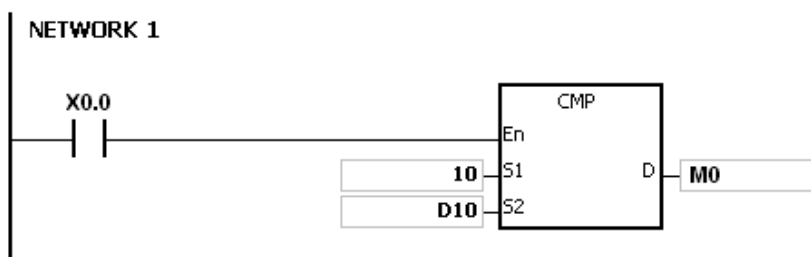
D : Comparison results

Explanation:

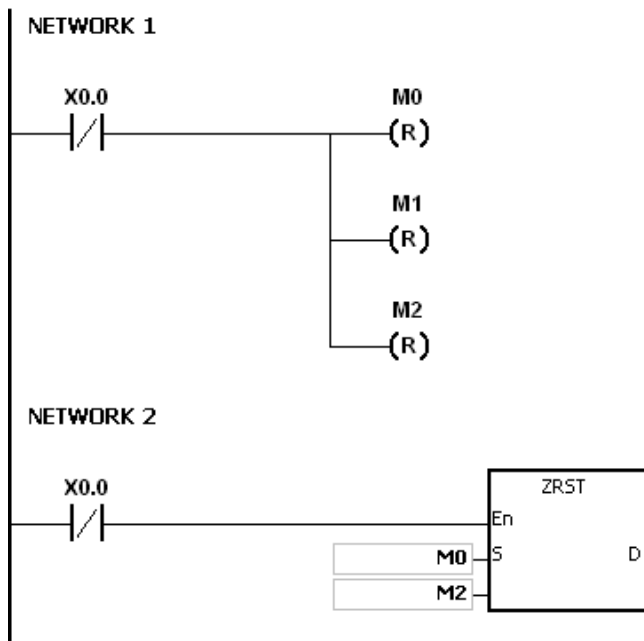
- The instruction is used to compare the value in S₁ with that in S₂, and the values compared are signed decimal numbers. The comparison results are stored in D.
- The operand D occupies three consecutive devices. The comparison results are stored in D, D+1, and D+2. If the comparison value in S₁ is greater than the comparison value in S₂, D will be ON. If the comparison value in S₁ is equal to the comparison value in S₂, D+1 is ON. If the comparison value in S₁ is less than the comparison value in S₂, D+2 will be ON.
- Only the instructions DCOMP and DCMPP can use the 32-bit counter.

Example:

- If the operand D is M0, the comparison results will be stored in M0, M1 and M2, as shown below.
- When X0.0 is ON, the instruction CMP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the execution of the instruction CMP stops. The state of M0, the state of M1, and the state of M1 remain unchanged.



- If you want to clear the comparison result, you can use the instruction ZRST.



Additional remark:

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

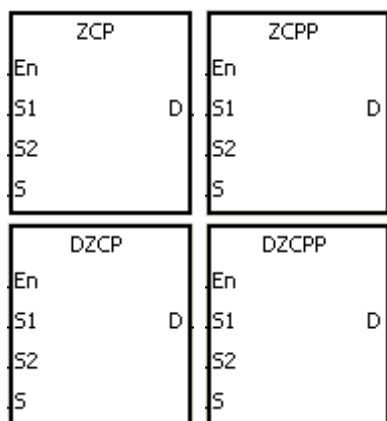
FB/FC	Instruction			Operand	Description
FC	D*	ZCP	P	S ₁ , S ₂ , S, D	Zone comparison

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂ , S		●	●*				●	●*						
D(array, 3)	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂ , S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Minimum value of the zone comparison

S₂ : Maximum value of the zone comparison

S : Comparison value

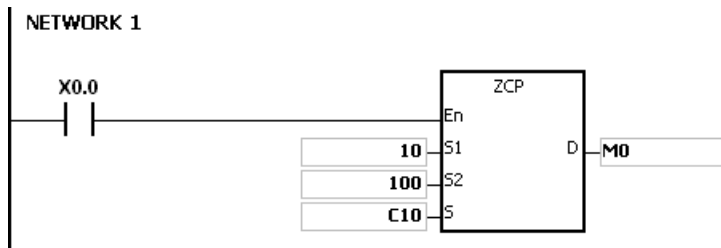
D : Comparison results

Explanation:

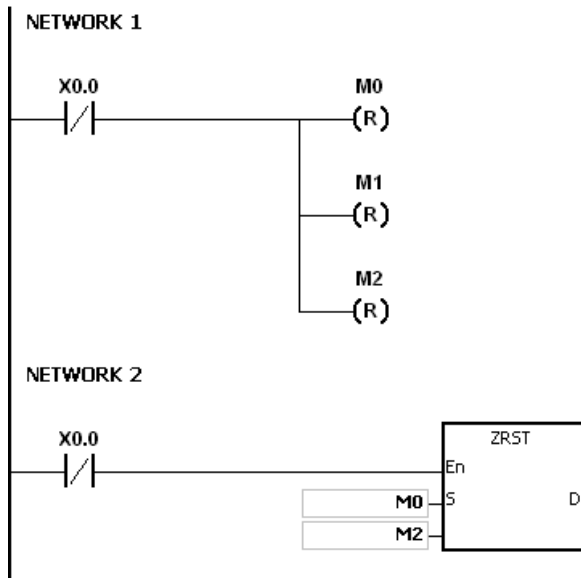
- The instruction is used to compare the value in **S** with that in **S₁**, and compare the value in **S** with that in **S₂**. The values compared are signed decimal numbers, and the comparison results are stored in **D**.
- The value in **S₁** must be less than that in **S₂**. If the value in **S₁** is larger than that in **S₂**, **S₁** will be taken as the maximum/minimum value during the execution of the instruction ZCP.
- The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D+1**, and **D+2**. If the comparison value in **S₁** is less than the comparison value in **S**, **D** will be ON. If the comparison value in **S** is within the range between the value in **S₁** and the value in **S₂**, **D+1** will be ON. If the comparison value in **S** is greater than the value in **S₂**, **D+2** will be ON.
- Only the instructions **DZCP** and **DZCPP** can use the 32-bit counter.

Example:

- If the operand **D** is M0, the comparison results will be stored in M0, M1 and M2, as shown below.
- When X0.0 is ON, the instruction ZCP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the instruction ZCP is not executed. The state of M0, the state of M1, and the state of M2 remain the same as those before X0.0's being OFF.



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



Additional remark:

1. If you declare the operand **D** in ISPSOft, the data type will be 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.

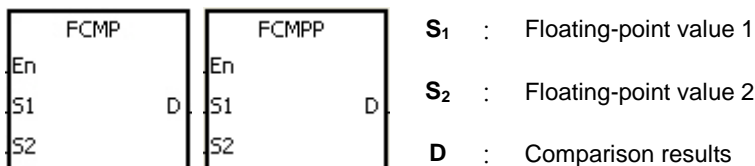
FB/FC	Instruction		Operand				Description					
FC		FCMP	P	S₁, S₂, D				Comparing the floating-point values				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂										●				
D(array, 3)	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●	●	●	●		●	○	●				○
D	●	●	●	●				●	●	●		●					

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

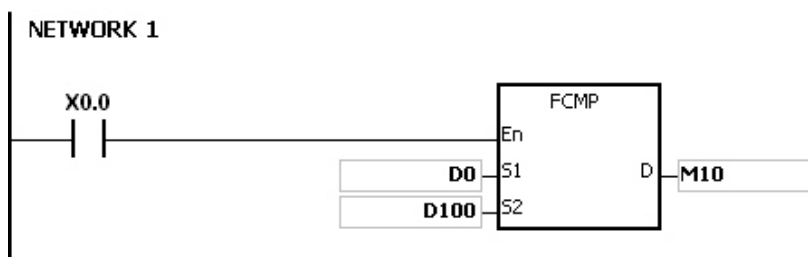


Explanation:

1. The instruction FCMP is used to compare the floating-point value in **S₁** with the floating-point value in **S₂**. The comparison results are stored in **D**.
2. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D+1**, and **D+2**. If the comparison value in **S₁** is greater than the comparison value in **S₂**, **D** will be ON. If the comparison value in **S₁** is equal to the value in **S₂**, **D+1** will ON. If the comparison value in **S₁** is less than the value in **S₂**, **D+2** will be ON.

Example:

1. If the operand **D** is M10, the comparison results will be stored in M10, M11 and M12, as shown below.
2. When X0.0 is ON, the instruction FCMP is executed. M10, M11, or M12 is ON. When X0.0 is OFF, the instruction FCMP is not executed. The state of M10, the state of M11, and the state of M12 remain the same as those before X0.0's being OFF.
3. If you want to get the comparison result \geq , \leq , or \neq , you can connect M10~M12 is series or in parallel.
4. If you want to clear the comparison result, you can use the instruction RST or ZRST.



Additional remark:

1. If the value in **S₁** or **S₂** is out of the range of values which can be represented by the floating-point values, the

contact is OFF, SM is ON, and the error code in SR0 is 16#2013.

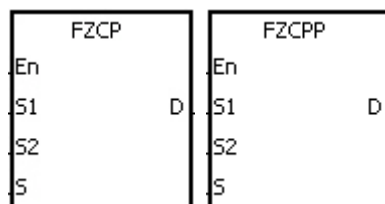
2. If you declare the operand **D** in ISPSOft, the data type will ARRAY [3] of BOOL.
3. If **D+2** exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction		Operand				Description					
FC		FZCP	P	S₁, S₂, S, D				Floating-point zone comparison				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂, S										●				
D(array, 3)	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂, S	●	●			●	●	●	●	●		●	○	●				○
D	○	●	●	●				●	●	●			●				

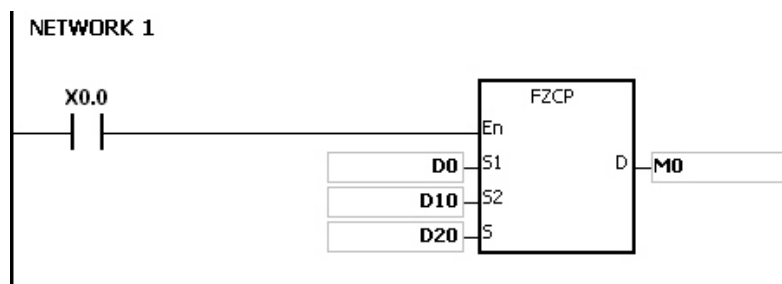
Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:**S₁** : Minimum value of the zone comparison**S₂** : Maximum value of the zone comparison**S** : Comparison value**D** : Comparison result**Explanation:**

1. The instruction is used to compare the value in **S** with that in **S₁**, and compare the value in **S** with that in **S₂**. The values compared are floating-point values, and the comparison results are stored in **D**.
2. The value in **S₁** must be less than that in **S₂**. If the value in **S₁** is larger than that in **S₂**, **S₁** will be taken as the maximum/minimum value during the execution of the instruction FZCP.
3. The operand **D** occupies three consecutive devices. The comparison results are stored in **D**, **D+1**, and **D+2**. If the comparison value in **S₁** is greater than the comparison value in **S**, **D** will be ON. If the comparison value in **S** is within the range between the value in **S₁** and the value in **S₂**, **D+1** will be ON. If the comparison value in **S₂** is less than the value in **S**, **D+2** will be ON.

Example:

1. If the operand **D** is M0, the comparison results will be stored in M0, M1 and M2.
2. When X0.0 is ON, the instruction FZCP is executed. M0, M1, or M2 is ON. When X0.0 is OFF, the instruction FZCP is not executed. The state of M0, the state of M1, and the state of M2 remain the same as those before X0.0's being OFF.
3. If you want to clear the comparison result, you can use the instruction RST or ZRST.



Additional remark:

1. If the value in **S₁** or **S₂** or **S** is out of the range of values which can be represented by the floating-point values, 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 will be 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.

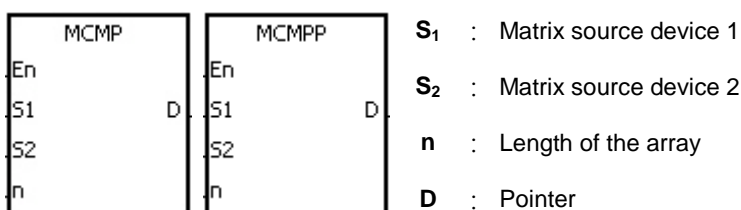
FB/FC	Instruction		Operand				Description					
FC		MCMP	P	S₁, S₂, n, D				Matrix comparison				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
n		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●		●	○	○		
D	●	●			●	●		●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

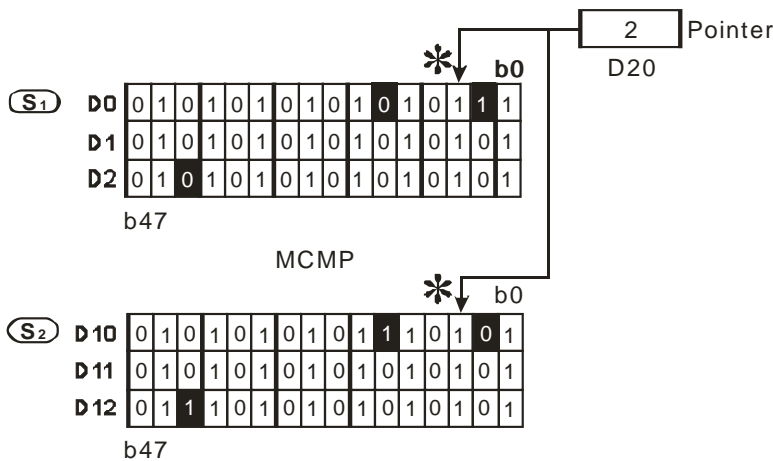
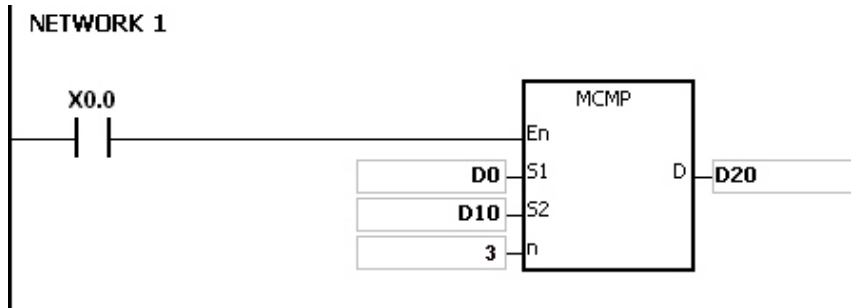
1. The search for the bits whose states are different starts from the bits specified by the number gotten from the addition of one to the current value in **D**. After the bits whose states are different are found, the bit number is stored in **D**, and the comparison is finished.
2. The operand **n** should be within the range between 1 and 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 compared, 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 0th bits in the next scan cycle, and SM609 is ON. When the value in **D** is out of the range, SM611 is ON.
4. When the instruction MCMP is executed, you need a 16-bit register to specify a certain bit among the 16n bits in the matrix for the operation. The register is called the pointer, and is specified by users. The value in the register is within the range between 0 and 16n-1, and corresponds to the bit within the range between b0 and b16n-1. During the operation, you should be prevented from altering the value of the pointer in case the search for the matching bits is affected. If the value of the pointer is out of the range, SM611 will be ON, and the instruction MCMP will not be executed.
5. If SM608 and SM610 occur simultaneously, they will be ON simultaneously.

Example:

1. When X0.0 is switched from OFF to ON, SM609 is OFF. The search for the bits whose states are different (SM607 is OFF) starts from the bits specified by the number gotten from the addition of one to the current value of the pointer.

2. Suppose the current value in D20 is 2. When X0.0 is switched from OFF to ON four times, you can 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 remark:

1. The description of the operation error code:

If the devices S_1+n-1 and S_2+n-1 exceed the range, the instruction MCMP is not executed, SM is ON, and the error code in SR0 is 16#2003.

If the value in the operand n is not within the range between 1 and 256, the instruction MCMP is not executed, SM is ON, and the error code in SR0 is 16#200B.

2. The description of the flags:

SM607:	It is the matrix comparison flag. ON: Comparing the equivalent values OFF: Comparing the different values
SM608:	The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.
SM609:	When SM609 is ON, the comparison starts from bit 0.
SM610:	It is the matrix bit search flag. When the matching bits are compared, the comparison stops immediately, and SM610 is ON.
SM611:	It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.

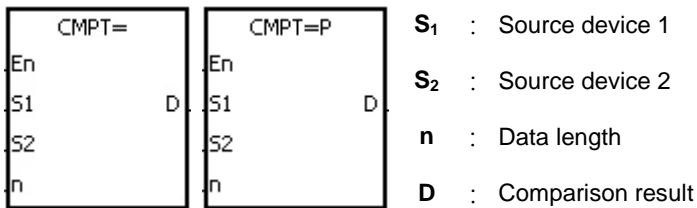
FB/FC	Instruction		Operand				Description					
FC		CMPT#	P	S₁, S₂, n, D				Comparing the tables				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							
n		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●		●	○	●	○	○		○
S ₂	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●	●	●				●	●	●		●	●				

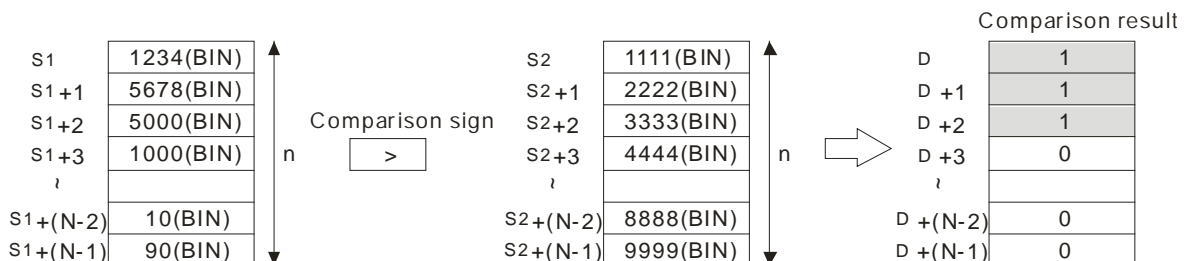
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

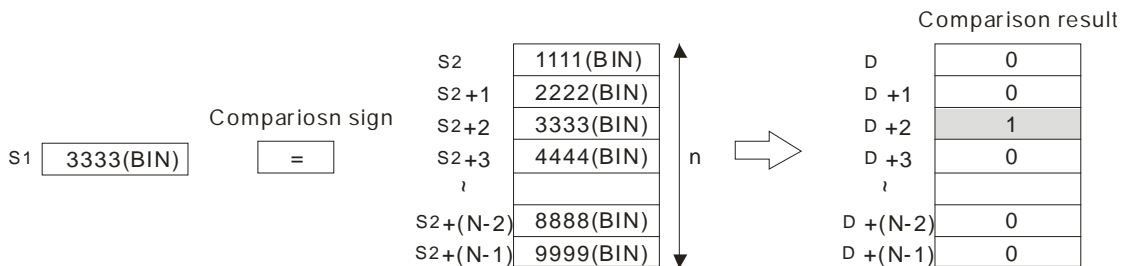


Explanation:

- The instruction is used to compare **n** pieces of data in devices starting from **S₁** with those in devices starting from **S₂**. The values compared are signed decimal numbers, and the comparison results are stored in **D**.
- The operand **n** should be within the range between 1 and 256.
- The value which is written into the operand **D** is a one-bit value.
- When the results gotten from the comparison by using the instruction CMPT# are that all devices are ON, SM620 is ON. Otherwise, SM620 is OFF.
- If the operand **S₁** is a device, the comparison will be as shown below.



- If the operand **S₁** is a constant within the range between -32768 and 32767, the comparison will be as shown below.

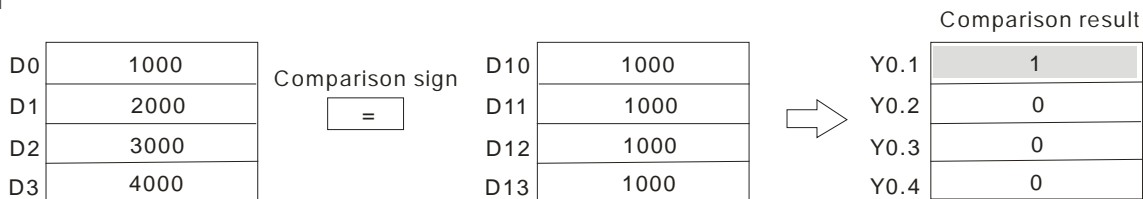
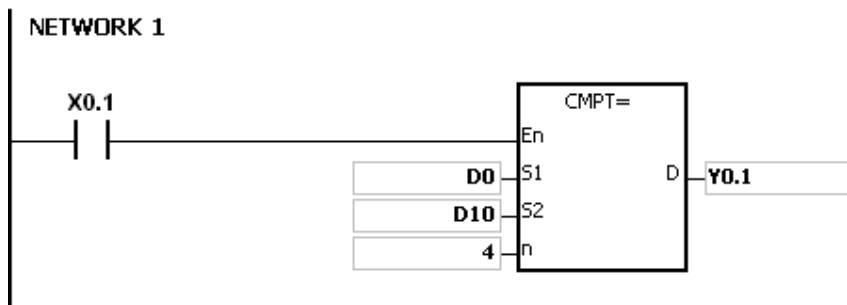


7. The corresponding comparison operation results of the instructions are listed below.

16-bit instruction	Comparison operation result	
	ON	OFF
CMPT =	$S_1 = S_2$	$S_1 \neq S_2$
CMPT < >	$S_1 \neq S_2$	$S_1 = S_2$
CMPT >	$S_1 > S_2$	$S_1 \leq S_2$
CMPT > =	$S_1 \geq S_2$	$S_1 < S_2$
CMPT <	$S_1 < S_2$	$S_1 \geq S_2$
CMPT < =	$S_1 \leq S_2$	$S_1 > S_2$

Example:

The data in D0~D3 are compared with that in D10~D13. If the comparison result is that the data in D0~D3 is the same as that in D10~D13, Y0.1~Y0.4 will be ON.



Additional remark:

1. If the value in the operand **n** is not within the range between 1 and 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 $S_1 \sim S_1+n$, $S_2 \sim S_2+n$, or **D** is insufficient, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

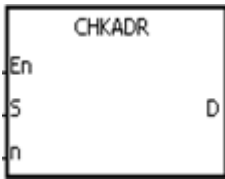
FB/FC	Instruction			Operand			Description		
FC	CHKADR			S, n, D			Checking the address of the contact type of pointer register		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S													●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



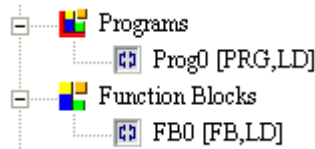
- S** : Pointer register Pointer / T_Pointer / C_Pointer / HC/AC_Pointer
- n** : Number of devices
- D** : Check result

Explanation:

- The instruction CHKADR is used to check whether the value in **S** and (the value in **S**)+**n**-1 exceed the device range. If the check result is that the value in **S** and (the value in **S**)+**n**-1 do not exceed the device range, the device **D** will be ON. Otherwise, it will be OFF.
- S** supports the pointer registers PR, TR, CR, and HCR.
- The operand **n** should be within the range between 1 and 1024.
- The instruction CHKADR only can be used in the function block.

Example:

- Establish a program and a function block in ISPSOft.



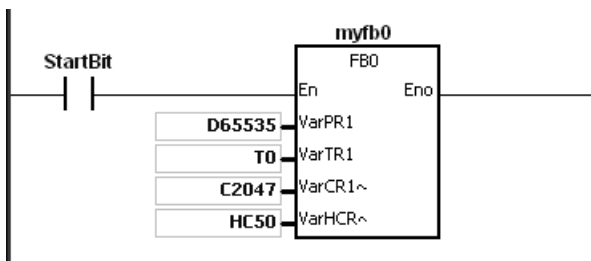
Declare two variables in the program.

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

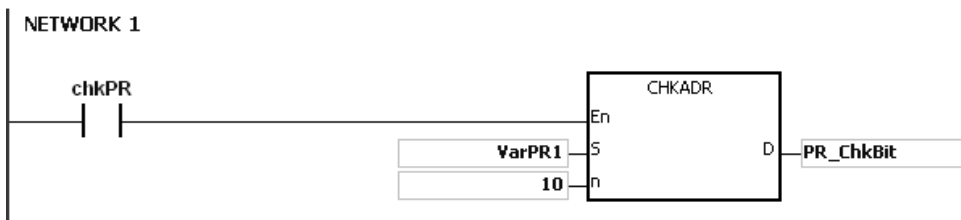
- Declare VarPR1, VarTR1, VarCR1, and VarHCR1 in the function block, and assign the data types POINTER, T_POINTER, C_POINTER, and HC/AC_POINTER to them respectively.

Local Symbols					
Class	Identifiers	Address	Type...	Initial Value	Identifier Comment...
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	

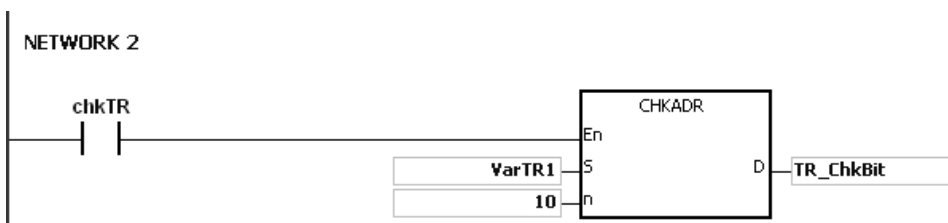
- Call the function block FB0 in the program, and assign D65535, T0, C2047, and HC50 to VarPR1, VarTR1, VarCR1, and VarHCR1 in FB0 respectively.



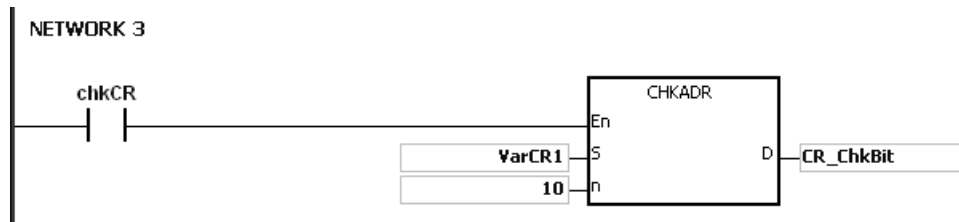
- Use the instruction CHKADR to check whether VarPR1, VarTR1, VarCR1, and VarHCR1 exceed the range.
- When chkPR is ON, the practical device represented by VarPR1 is D65535. Since the legal range of devices is from D0 to D65535, and $D65535+10-1=D65544$, which is out of the range, PR_ChkBit is OFF.



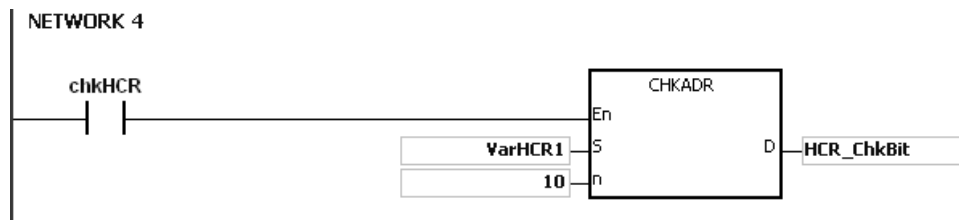
- When chkTR is ON, the practical device represented by VarTR1 is T0. Since the legal range of devices is from T0 to T2047, and $T0+10-1=T9$, which does not exceed the range, TR_ChkBit is ON.



- When chkCR is ON, the practical device represented by C2047. Since the legal range of devices is from C0 to C2047, and $C2047+10-1=C2056$, which is out of the range, CR_ChkBit is OFF.



8. When chkHCR is ON, the practical device represented by HC50 is VarHCR1. Since the legal range of deices is from HC0 to HC63, and $HC50+10-1=HC59$, which does not exceed the range, HCR_ChkBit is ON.



Additional remark:

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

3.4 Arithmetic Instructions

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>±</u>	<u>D+</u>	–	✓	Addition of binary values	7
FC	<u>±</u>	<u>D-</u>	–	✓	Subtraction of binary values	7
FC	<u>*</u>	<u>D*</u>	–	✓	Multiplication of binary values	7
FC	<u>/</u>	<u>D/</u>	–	✓	Division of binary values	7
FC	–	<u>F+</u>	<u>DF+</u>	✓	Addition of floating-point values	7-9
FC	–	<u>F-</u>	<u>DF-</u>	✓	Subtraction of floating-point values	7-9
FC	–	<u>F*</u>	<u>DF*</u>	✓	Multiplication of floating-point values	7-9
FC	–	<u>F/</u>	<u>DF/</u>	✓	Division of floating-point values	7-9
FC	<u>B+</u>	<u>DB+</u>	–	✓	Addition of binary-coded decimal numbers	7
FC	<u>B-</u>	<u>DB-</u>	–	✓	Subtraction of binary-coded decimal numbers	7
FC	<u>B*</u>	<u>DB*</u>	–	✓	Multiplication of binary-coded decimal numbers	7
FC	<u>B/</u>	<u>DB/</u>	–	✓	Division of binary-coded decimal numbers	7
FC	<u>BK+</u>	–	–	✓	Binary number block addition	9
FC	<u>BK-</u>	–	–	✓	Binary number block subtraction	9
FC	<u>\$+</u>	–	–	✓	Linking the strings	7-19
FC	<u>INC</u>	<u>DINC</u>	–	✓	Adding one to the binary number	3
FC	<u>DEC</u>	<u>DDEC</u>	–	✓	Subtracting one from the binary number	3
FC	<u>MUL16</u>	<u>MUL32</u>	–	✓	16-bit binary multiplication/ 32-bit binary multiplication	7
FC	<u>DIV16</u>	<u>DIV32</u>	–	✓	16-bit binary division/ 32-bit binary division	7

3

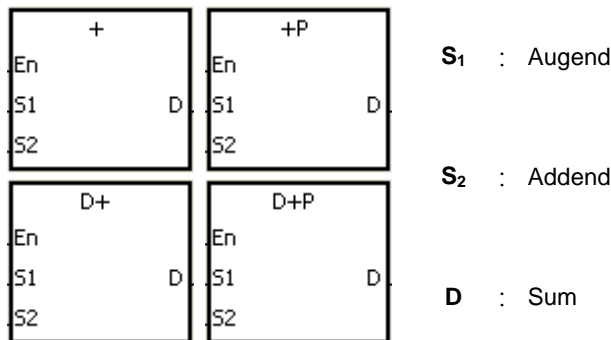
FB/FC	Instruction			Operand	Description
FC	D*	+	P	S ₁ , S ₂ , D	Addition of binary values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

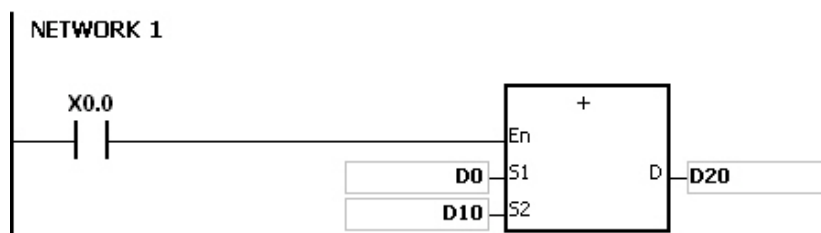


Explanation:

1. The binary value in **S₂** is added to the binary value in **S₁**, and the sum is stored in **D**.
2. Only the 32-bit instructions can use the 32-bit counter.
3. The 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. The addition of 16-bit binary values:
When the operation result is out of the range of 16-bit binary values, SM602 is ON. Otherwise, it is OFF.
6. The addition of 32-bit binary values:
When the operation result is out of the range of 32-bit binary values, SM602 is ON. Otherwise, it is OFF.

Example 1:

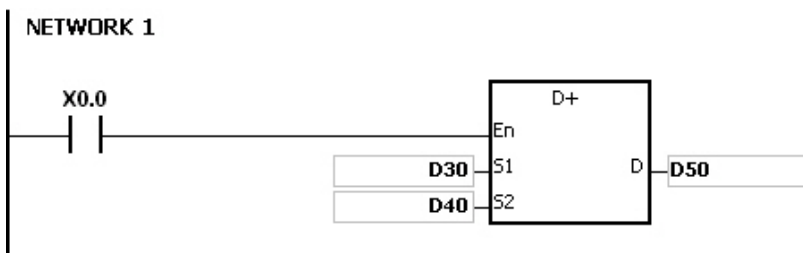
The addition of 16-bit binary values: When X0.0 is ON, the addend in D10 is added to the augend in D0, and sum is stored in D20.



- 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 is out of the range of 16-bit binary values, SM602 is ON, and the value stored in D20 is 16#0. Besides, since the operation result is 16#0, SM600 is ON.

Example 2:

The addition of 32-bit binary values: When X0.0 is ON, the addend in (D41, D40) is added to the augend in (D31, D30), and sum is stored 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).

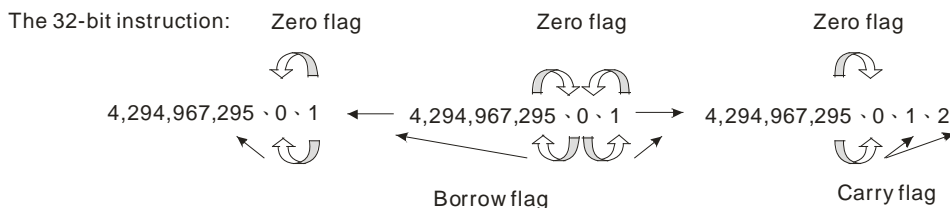
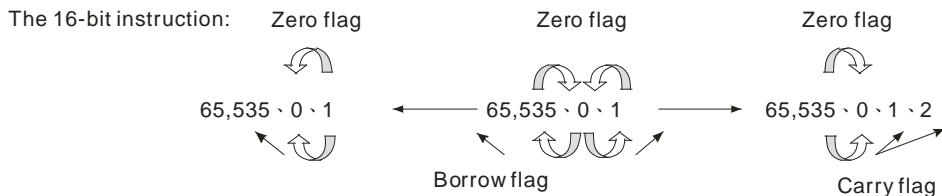


- When the values in (D31, D30) and (D41, D40) are 11111111 and 44444444 respectively, (D31, D30) plus (D41, D40) equals 55555555, and 55555555 is stored in (D51, D50).
- When the values in (D31, D30) and (D41, D40) are 16#80000000 and 16#FFFFFFFF respectively, (D31, D30) plus (D41, D40) equals 16#17FFFFFFF.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

Additional remarks:

Operation of flags:

- The 16-bit instruction
 - If the operation result is zero, SM600 will be set to ON.
 - If the operation result exceeds 65,535, SM602 will be set to ON.
- The 32-bit instruction
 - If the operation result is zero, SM600 will be set to ON.
 - If the operation result exceeds 4,294,967,295, SM602 will be set to ON.



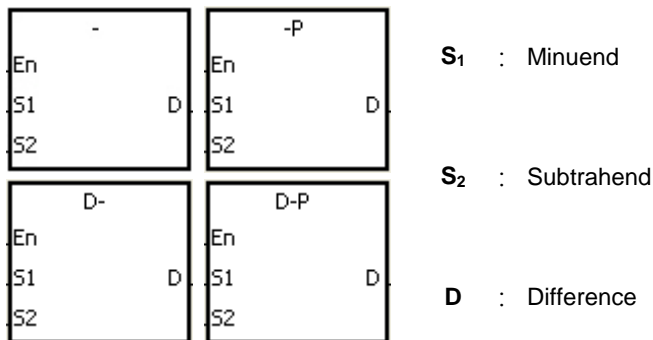
FB/FC	Instruction			Operand	Description
FC	D*	-	P	S ₁ , S ₂ , D	Subtraction of binary values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●			●	○		○	○		
D	●	●			●	●	●	●			●	○					

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

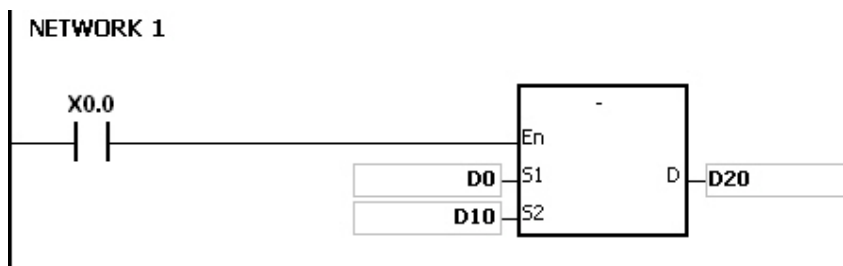


Explanation:

1. The binary value in **S₂** is subtracted from the binary value in **S₁**, and the difference is stored in **D**.
2. Only the 32-bit instructions can use the 32-bit counter.
3. The 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 a borrow occurs during the arithmetic, SM601 is ON. Otherwise, it is OFF.

Example 1:

The subtraction of 16-bit binary values: When X0.0 is ON, the subtrahend in D10 is subtracted from the minuend in D0, and the difference is stored in D20.

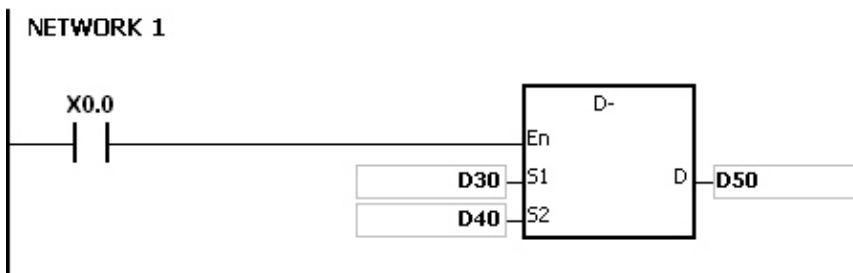


- When the values in D0 and D10 are 100 and 10 respectively, D0 minus D10 leaves 90, and 90 is stored in D20.

- 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 the borrow occurs during the arithmetic, 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 the borrow occurs during the arithmetic, SM601 is ON, and the value stored in D20 is 16#1.

Example 2:

The addition of 32-bit binary values: When X0.0 is ON, the subtrahend in (D41, D40) is subtracted from the minuend in (D31, D30), and sum is stored 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) 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#F8000001. Since the borrow occurs during the arithmetic, SM601 is ON, and the value stored in (D51, D50) is 16#80000001.

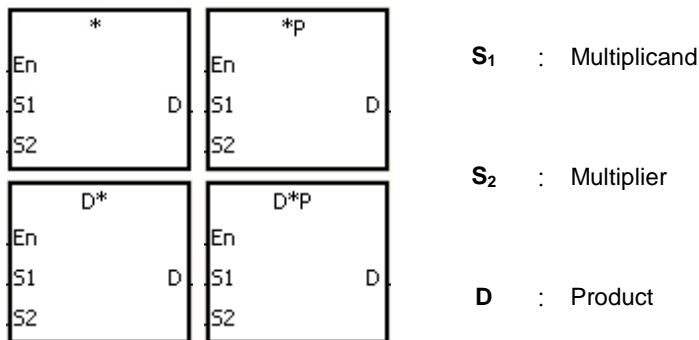
FB/FC	Instruction			Operand	Description
FC	D*	*	P	S ₁ , S ₂ , D	Multiplication of binary values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D			●	●*				●	●*					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		○
D	●	●			●	●	●	●	●		●	○	●				

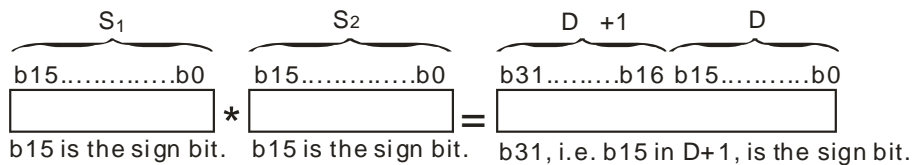
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



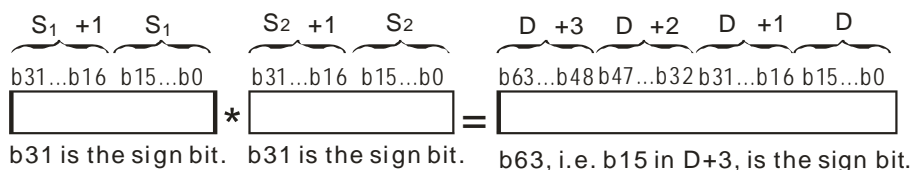
Explanation:

- The signed binary value in **S₁** is multiplied by the signed binary value in **S₂**, and the product is stored in **D**.
- Only the instruction **D*** can use the 32-bit counter.
- The multiplication of 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.

- The multiplication of 32-bit binary values:

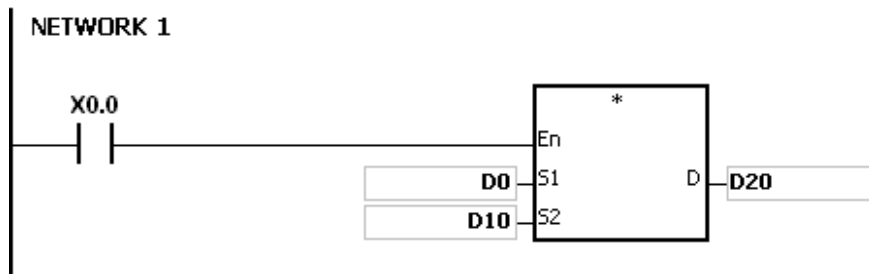


The product is a 64-bit value, and is stored in the register (D+3, D+2, D+1, D0), which is composed of 64 bits. When the sign bit b63 is 0, the product is a positive value. When the sign bit b63 is 1, the product is a negative value.

value.

Example:

The 16-bit value in D0 is multiplied by the 16-bit value in D10, and the 32-bit product is stored 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.



$D0 \times D10 = (D21, D20)$

16-bit value \times 16-bit value = 32-bit value

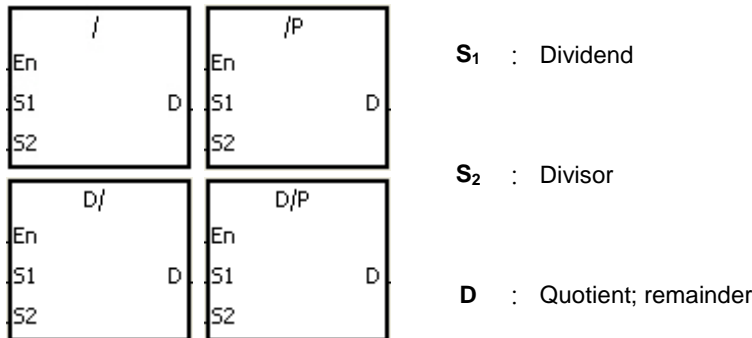
FB/FC	Instruction			Operand	Description
FC	D*	/	P	S ₁ , S ₂ , D	Division of binary values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D(array, 2)		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

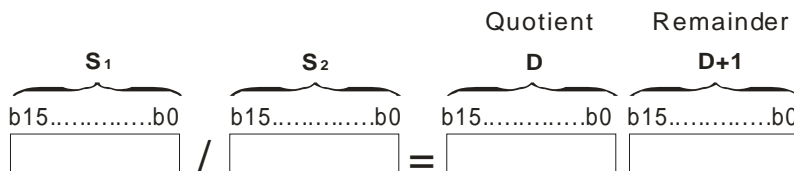
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



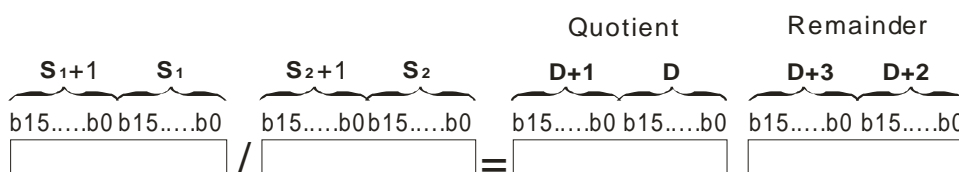
Explanation:

- The signed binary value in **S₁** is divided by the signed binary value in **S₂**. The quotient and the remainder are stored in **D**.
- Only the 32-bit instructions can use the 32-bit counter.
- When the sign bit is 0, the value is a positive one. When the sign bit is 1, the value is a negative one.
- The division of 16-bit values:



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

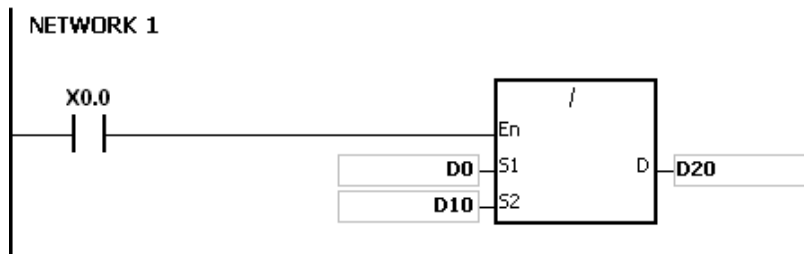
- The division of 32-bit values:



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 X0.0 is ON, the dividend in D0 is divided by the divisor in D10, the quotient is stored in D20, and the remainder is stored in D21. Whether the result is a positive value or a negative value depends on the state of the highest bit.



Additional remark:

1. If the device is out of 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 the operand **D** used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
4. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of DWORD/DINT.

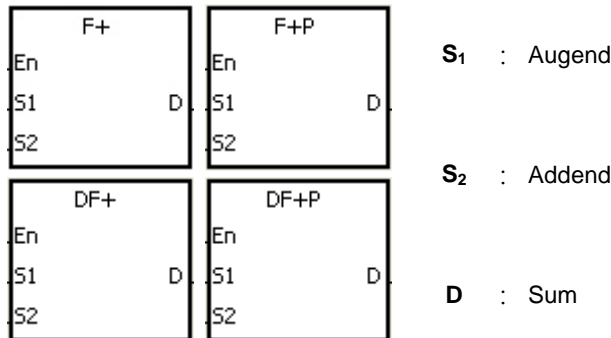
FB/FC	Instruction			Operand				Description				
FC	D*	F+	P	S ₁ , S ₂ , D				Addition of floating-point values				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			
D										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

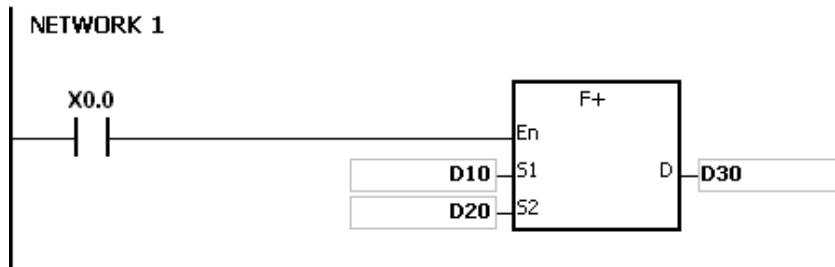


Explanation:

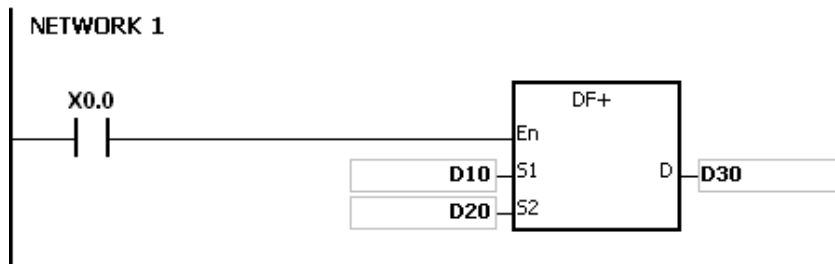
- The floating-point value in **S₂** is added to the floating-point value in **S₁**, and the sum is stored in **D**.
- The addition of 32-bit single-precision floating-point values:
 - When the operation result is zero, SM600 is ON.
 - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FF7FFFFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7F7FFFFFFF.
- The addition of 64-bit double-precision floating-point values:
 - When the operation result is zero, SM600 is ON.
 - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FFEFFFFFFFFFFFFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7FEFFFFFFFFFFFFFFF.

Example:

- The addition of single-precision floating-point values: When X0.0 is ON, the addend 16#4046B852 in (D21, D20) is added to the augend 16#3FB9999A in (D11, D10), and the sum 16#4091C28F is stored in (D31, D30). 16#4046B852, 16#3FB9999A, and 16#4091C28F represent the floating point numbers 3.105, 1.450, and 4.555 respectively.



2. The addition of double-precision floating-point values: When X0.0 is ON, the addend 16#4008D70A3D70A3D7 in (D23, D22, D21, D20) is added to the augend 16#3FF7333333333333 in (D13, D12, D11, D10), and the sum 16# 40123851EB851EB8 is stored in (D33, D32, D31, D30).



3

Additional remark:

If the value in **S₁** or the value in **S₂** is out of the range of values which can be represented by the floating-point values, the instruction is not executed, SMO is ON, and the error code in SR0 is 16#2013.

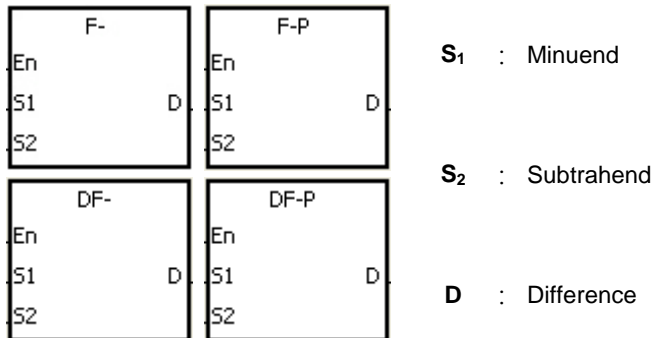
FB/FC	Instruction			Operand	Description
FC	D*	F-	P	S ₁ , S ₂ , D	Subtraction of floating-point values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			
D										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

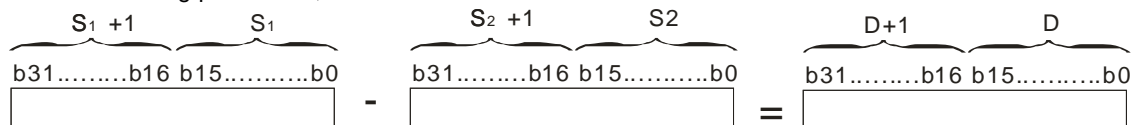
Graphic expression:



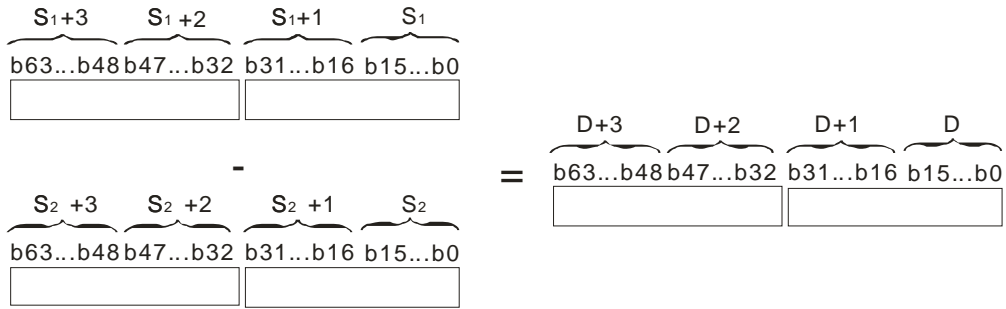
Explanation:

- The floating-point value in **S₂** is subtracted from the floating-point value in **S₁**, and the difference is store in **D**.
- When the operation result is zero, SM600 is ON.
- The subtraction of 32-bit single-precision floating-point values:

- When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FF7FFFFFFF.
- When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7F7FFFFFFF.

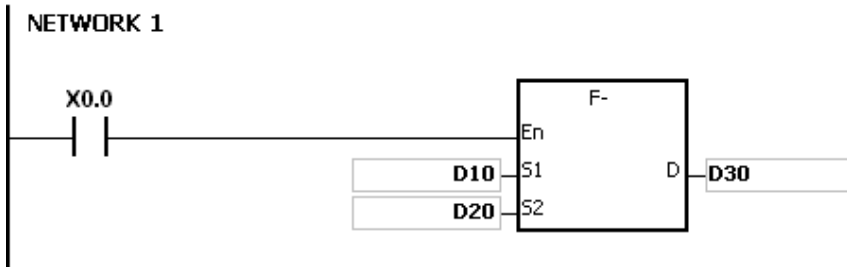


- The subtraction of 64-bit double-precision floating-point values:
- When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FFEFFFFFFFFFFFFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7FEFFFFFFFFFFFFFFF.

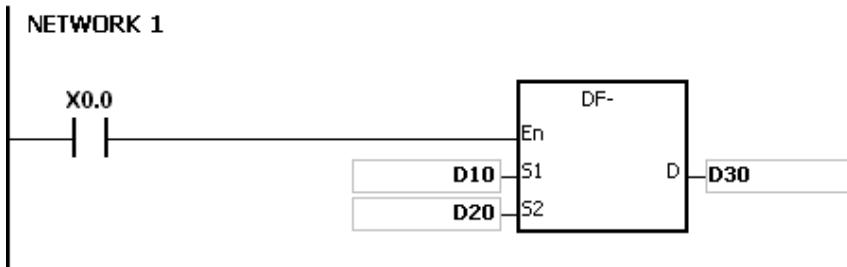


Example:

1. The subtraction of 32-bit single-precision floating-point values: When X0.0 is ON, the subtrahend in (D21, D20) is subtracted from the minuend in (D21, D20), and the difference is stored in (D31, D30).



2. The subtraction of 64-bit double-precision floating-point values: When X0.0 is ON, the subtrahend in (D23, D22, D21, D20) is subtracted from the minuend in (D13, D12, D11, D10), and the difference is stored in (D33, D32, D31, D30).



Additional remark:

If the value in **S₁** or the value in **S₂** is out of the range of values which can be represented by the floating-point values, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

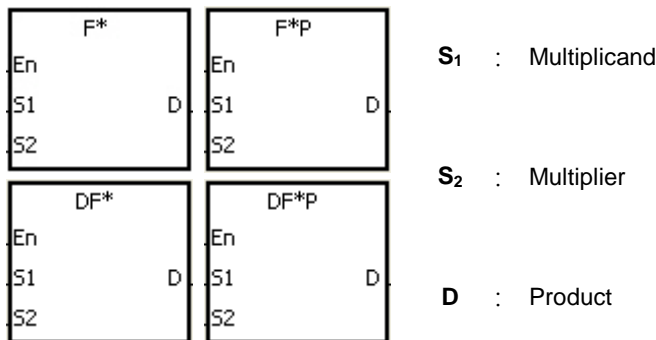
FB/FC	Instruction			Operand	Description
FC	D*	F*	P	S ₁ , S ₂ , D	Multiplication of floating-point values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			
D										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○
D	●	●			●	●	●	●	●		●	○	●				

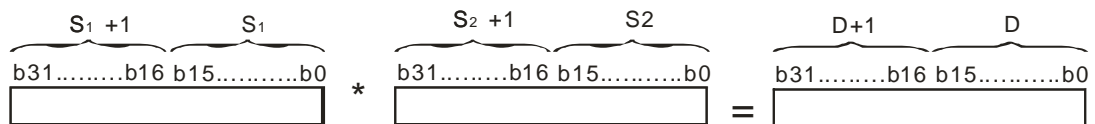
Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

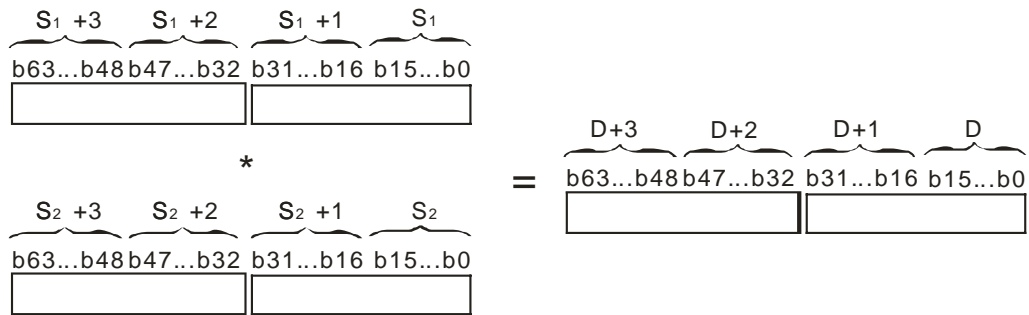


Explanation:

- The floating-point value in **S₁** is multiplied by the floating-point value in **S₂**, and the product is stored in **D**.
- When the operation result is zero, SM600 is ON.
- The multiplication of 32-bit single-precision floating-point values:
 - When the absolute value of the operation result is less than value which can be represented by the minimum floating-point value, the value in **D** is 16#FF7FFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7F7FFFFF.

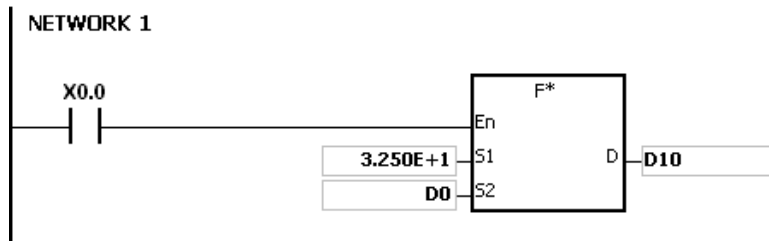


- The multiplication of 64-bit double-precision floating-point values:
 - When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FFEFFFFFFFFFFFFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7FEFFFFFFFFFFFFFFF.

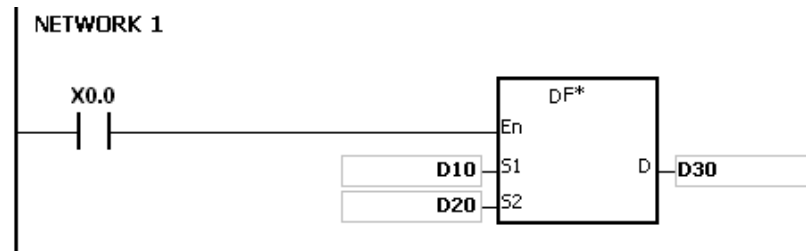


Example:

1. The multiplication of 32-bit single-precision floating-point values: When X0.0 is ON, the multiplicand 32.5 is multiplied by the multiplier in (D1, D0), and the product is stored in (D11, D10).



2. The multiplication of 64-bit double-precision floating-point values: When X0.0 is ON, the multiplicand in (D13, D12, D11, D10) is multiplied by the multiplier in (D23, D22, D21, D20), and the product is stored in (D33, D32, D31, D30).



Additional remark:

If the value in **S₁** or the value in **S₂** is out of the range of values which can be represented by the floating-point values, the instruction is not executed, **SM0** is ON, and the error code in **SR0** is 16#2013.

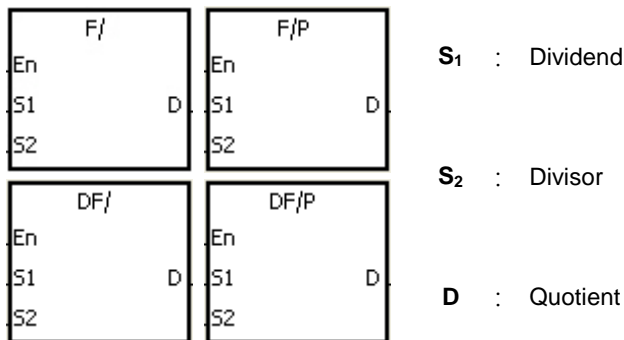
FB/FC	Instruction			Operand	Description
FC	D*	F/	P	S ₁ , S ₂ , D	Division of floating-point values

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			
D										●	●*			

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●				○
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



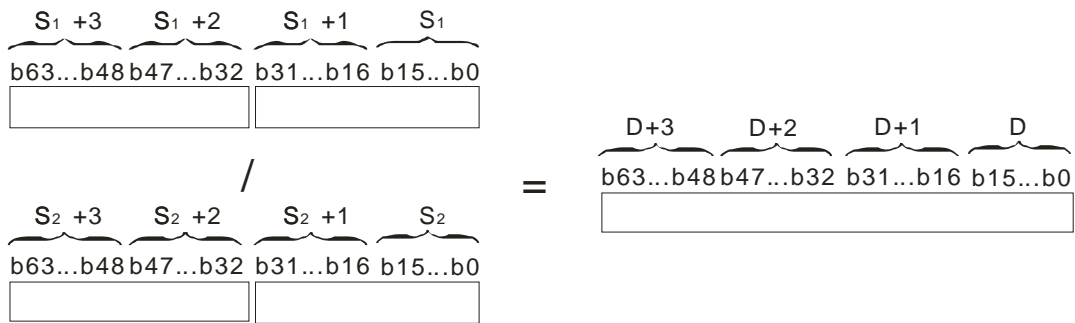
Explanation:

- The single-precision floating-point value in S1 is divided by the single-precision floating-point number in S2. The quotient is stored in D.
- When the operation result is zero, SM600 is ON.
- The division of 32-bit single-precision floating-point values:

- When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FF7FFFFF.
- When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7F7FFFFF.

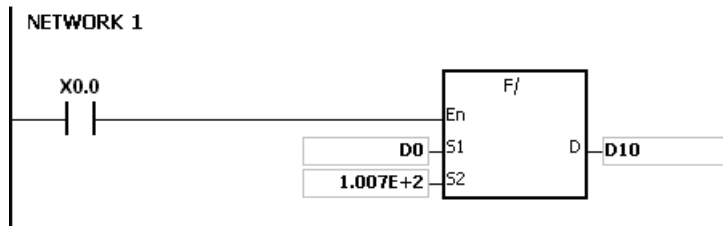
$$\begin{array}{c}
 \overbrace{b31 \dots b16}^{S_1 + 1} \quad \overbrace{b15 \dots b0}^{S_1} \\
 \hline
 \boxed{}
 \end{array}
 /
 \begin{array}{c}
 \overbrace{b31 \dots b16}^{S_2 + 1} \quad \overbrace{b15 \dots b0}^{S_2} \\
 \hline
 \boxed{}
 \end{array}
 =
 \begin{array}{c}
 \overbrace{b31 \dots b16}^{D + 1} \quad \overbrace{b15 \dots b0}^{D} \\
 \hline
 \boxed{}
 \end{array}$$

- The division of 64-bit double-precision floating-point values:
- When the absolute value of the operation result is less than the value which can be represented by the minimum floating-point value, the value in **D** is 16#FFEFFFFFFFFFFFFFFF.
 - When the absolute value of the operation result is larger than the value which can be represented by the maximum floating-point value, the value in **D** is 16#7FEFFFFFFFFFFFFFFF.

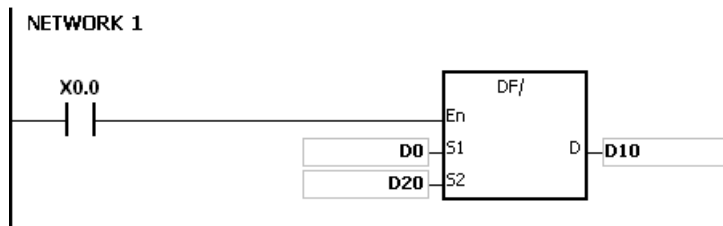


Example:

1. The division of 32-bit single-precision floating-point values: When X0.0 is ON, the dividend in (D1, D0) is divided by the divisor 100.7, and the quotient is stored in (D11, D10).



2. The division of 64-bit double-precision floating-point values: When X0.0 is ON, the dividend in (D3, D2, D1, D0) is divided by the divisor in (D23, D22, D21, D20), and the quotient is stored in (D13, D12, D11, D10).



Additional remark:

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 S1 or the value in S2 is out of the range of values which can be represented by the floating-point values, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

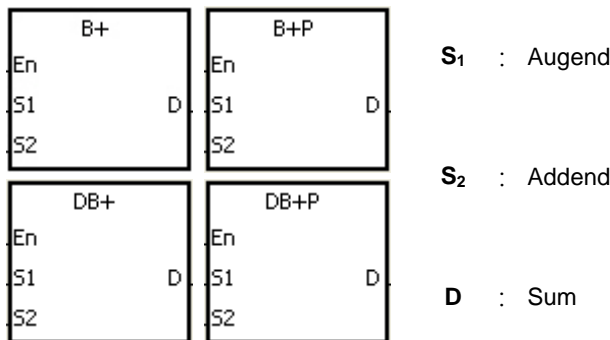
FB/FC	Instruction			Operand	Description
FC	D*	B+	P	S ₁ , S ₂ , D	Addition of binary-coded decimal numbers

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

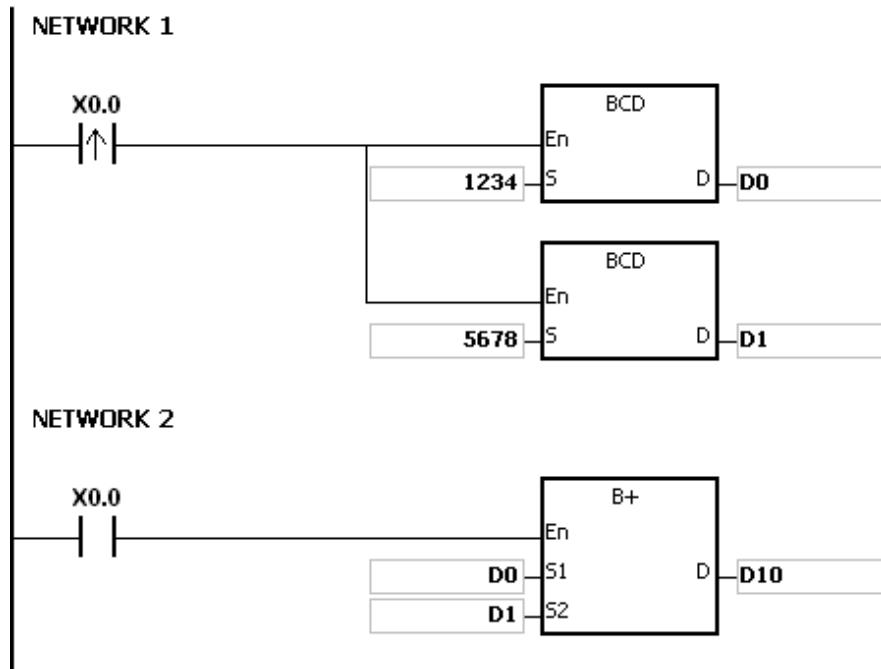


Explanation:

- The binary-coded decimal value in S₂ is added to the binary-coded decimal value in S₁, and the sum is stored in D.
- Only the instruction DB+ can use the 32-bit counter.
- The binary-coded decimal value is represented by the hexadecimal number, and every digit is within the range between 0 and 9.
- The addition of 16-bit binary-coded decimal values:
- When the binary-coded decimal values in S₁ and S₂ are 9999 and 0002 respectively, S₁ plus S₂ equals the binary-coded decimal value 10001. Since the carry is ignored, the binary coded-decimal value stored in D is 0001.
- The addition of 32-bit binary-coded decimal values:
- When the binary-coded decimal values in S₁ and S₂ are 99999999 and 00000002 respectively, S₁ plus S₂ equals the binary-coded decimal value 100000001. Since the carry is ignored, the binary coded-decimal value stored in D is 00000001.

Example:

When X0.0 is ON, the constants 1234 and 5678 are converted into the binary-coded decimal values which are stored in D0 and D1 respectively. The binary-coded decimal value in D1 is added to the binary-coded decimal value in D0, and the sum is stored in D10.



Additional remark:

1. If the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~9999, the instruction B+ 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. If the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~99999999, the instruction DB+ 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. The instruction does not support SM600, SM601 and SM602.

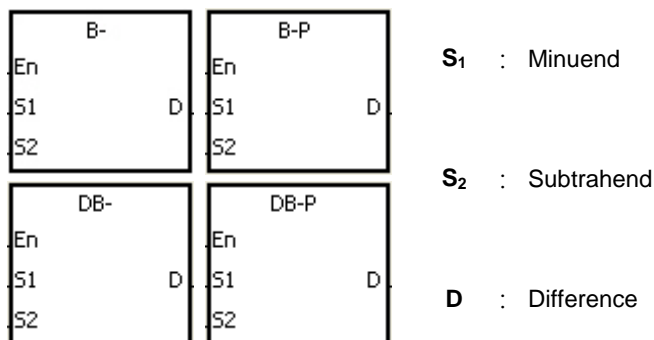
FB/FC	Instruction			Operand	Description
FC	D*	B-	P	S ₁ , S ₂ , D	Subtraction of binary-coded decimal numbers

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		○
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



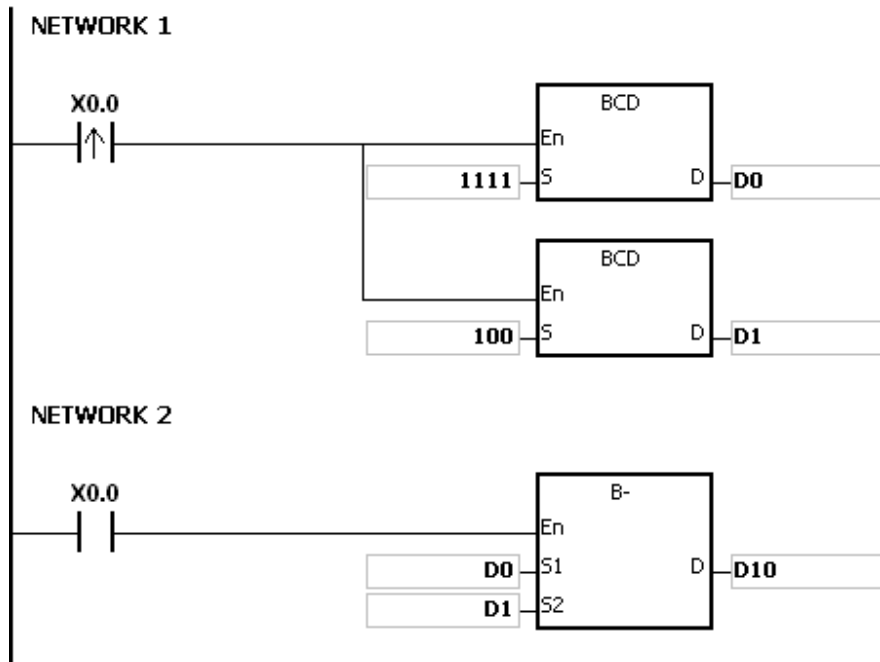
Explanation:

- The binary-coded decimal value in S₂ is subtracted from the binary-coded decimal value in S₁, and the difference is stored in D.
- Only the instruction DB- can use the 32-bit counter.
- The binary-coded decimal value is represented by the hexadecimal number, and every digit is within the range between 0 and 9.
- The subtraction of 16-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 9999 and 9998 respectively, S₁ minus S₂ leaves the binary-coded decimal value 0001, and 0001 is stored in D.
 - When the binary-coded decimal values in S₁ and S₂ are 0001 and 9999 respectively, S₁ minus S₂ leaves the binary-coded decimal value -9998, and the binary-coded decimal value 0002 is stored in D.
 - When the binary-coded decimal values in S₁ and S₂ are 0001 and 0004 respectively, S₁ minus S₂ leaves the binary-coded decimal value -0003, and the binary-coded decimal value 9997 is stored in D.
- The subtraction of 32-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 99999999 and 99999998 respectively, S₁ minus S₂ leaves the binary-coded decimal value 00000001, and 00000001 is stored in D.
 - When the binary-coded decimal values in S₁ and S₂ are 00000001 and 99999999 respectively, S₁ minus S₂ leaves the binary-coded decimal value -99999998, and the binary-coded decimal value 00000002 is stored in D.

- When the binary-coded decimal values in S1 and S2 are 00000001 and 00000004 respectively, S1 minus S2 leaves the binary-coded decimal value -00000003, and the binary-coded decimal value 99999997 is stored in D.

Example:

When X0.0 is ON, the constants 1111 and 100 are converted into the binary-coded decimal values which are stored in D0 and D1 respectively. The binary-coded decimal value in D1 is subtracted from the binary-coded decimal value in D0, and the difference is stored in D10.



Additional remark:

- If the value in S₁ or the value in S₂ is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~9999, the instruction B- 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.).
- If the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~99999999, the instruction DB- 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.).
- The instruction does not support SM600, SM601 and SM602.

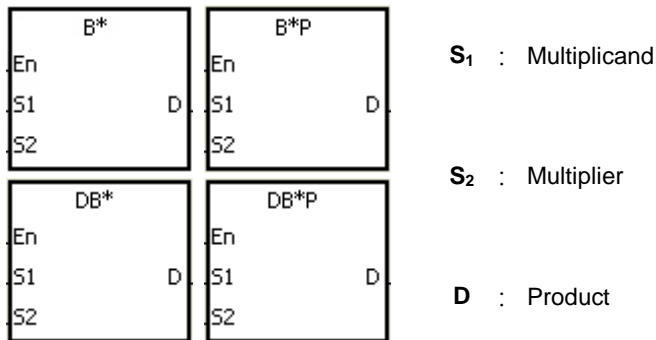
FB/FC	Instruction			Operand	Description
FC	D*	B*	P	S ₁ , S ₂ , D	Multiplication of binary-coded decimal numbers

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D			●	●*				●	●*					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

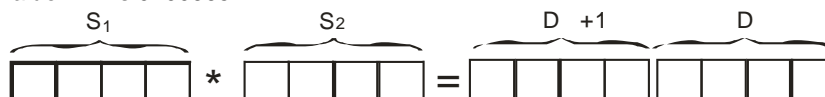
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



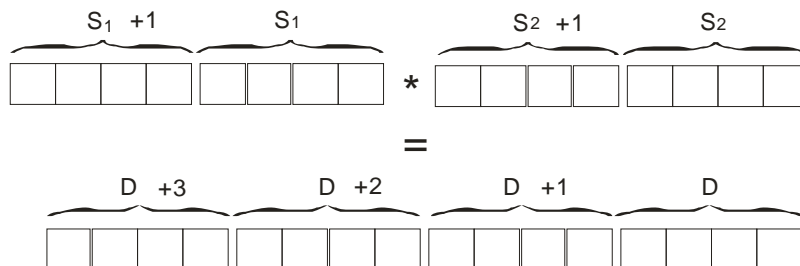
Explanation:

- The binary-coded decimal value in S₁ is multiplied by the binary-coded decimal value in S₂, and the product is stored in D.
- Only the instruction DB* can use the 32-bit counter.
- The binary-coded decimal value is represented by the hexadecimal number, and every digit is within the range between 0 and 9.
- The multiplication of 16-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 1234 and 5678 respectively, the binary-coded decimal value in D is 07006652.



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

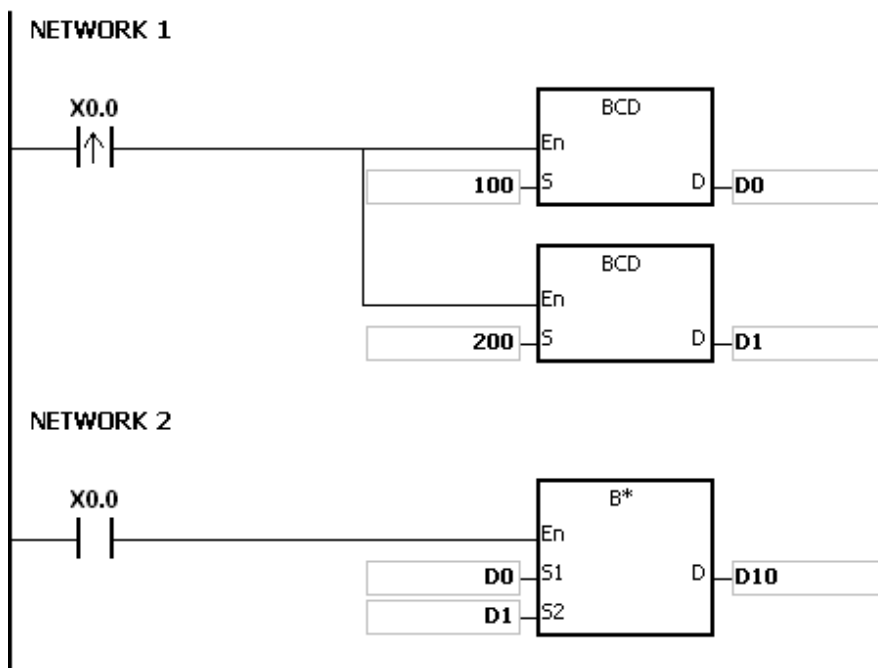
- The multiplication of 32-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 99999999 and 99999998 respectively, the binary-coded decimal value in D is 9999999700000002.



- The product is a 64-bit value, and is stored in the register (D+3, D+2, D+1, D), which is composed of 64 bits.

Example:

1. When X0.0 is ON, the constants 100 and 200 are converted into the binary-coded decimal values which are stored in D0 and D1 respectively. The binary-coded decimal value in D0 is multiplied by the binary-coded decimal value in D1, and the product is stored in D10.



Additional remark:

1. When the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~9999, the instruction B* 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. When the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~99999999, the instruction DB* 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. The instruction does not support SM600, SM601 and SM602.

FB/FC	Instruction			Operand			Description		
FC	D*	B/	P	S ₁ , S ₂ , D			Division of binary-coded decimal numbers		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D			●	●*				●	●*					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

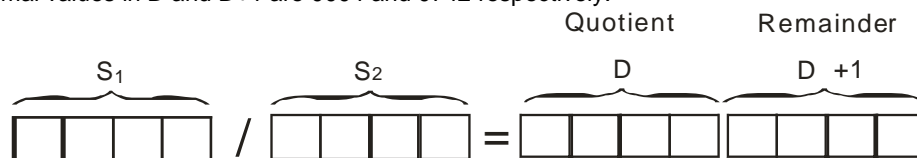
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

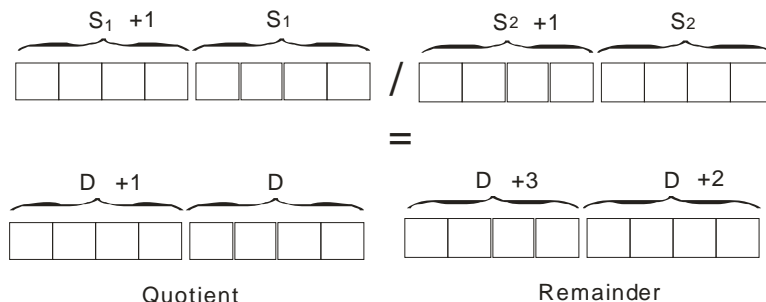


Explanation:

- The binary-coded decimal value in S₁ is divided by the binary-coded decimal value in S₂, and the quotient is stored in D.
- Only the instruction DB/ can use the 32-bit counter.
- The binary-coded decimal value is represented by the hexadecimal number, and every digit is within the range between 0 and 9.
- The division of 16-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 1234 and 5678 respectively, the binary-coded decimal values in D and D+1 are 0004 and 0742 respectively.



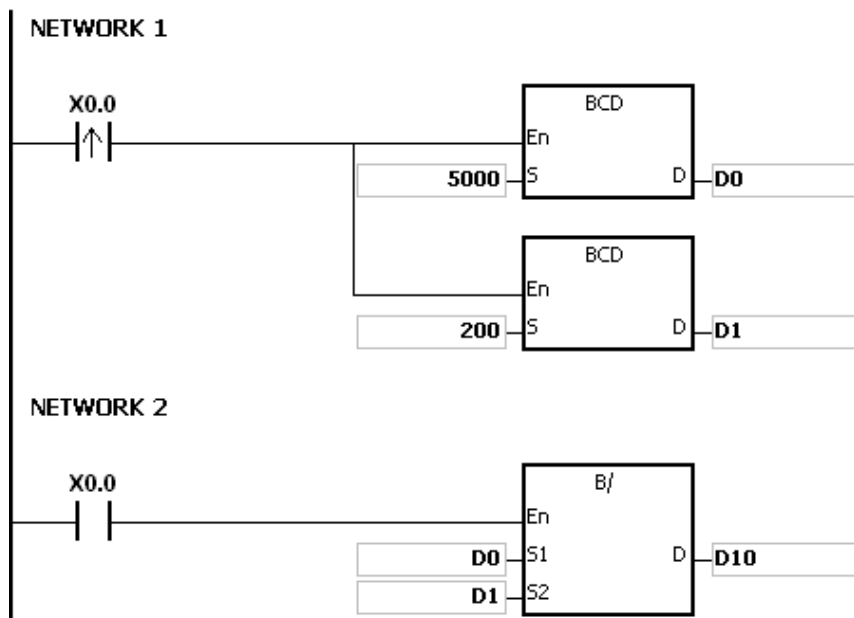
- The operand D occupies two consecutive devices. The quotient is stored in D, and the remainder is stored in D+1.
- The division of 32-bit binary-coded decimal values:
 - When the binary-coded decimal values in S₁ and S₂ are 87654321 and 12345678 respectively, the binary-coded decimal values in (D+1, D) and (D+3, D+2) are 00000007 and 01234575 respectively.



- 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 X0.0 is ON, the constants 5000 and 200 are converted into the binary-coded decimal values which are stored in D0 and D1 respectively. The binary-coded decimal value in D0 is divided by the binary-coded decimal value in D1. The quotient and the remainder are stored in D10 and D11 respectively.



Additional remark:

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 S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~9999, the instruction B/ 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. If the value in S1 or the value in S2 is out of the range of values which can be represented by the binary-coded decimal values, i.e. 0~99999999, the instruction DB/ 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.).
4. The instruction does not support SM600, SM601 and SM602.
5. If the operand D used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
6. If the operand D used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.

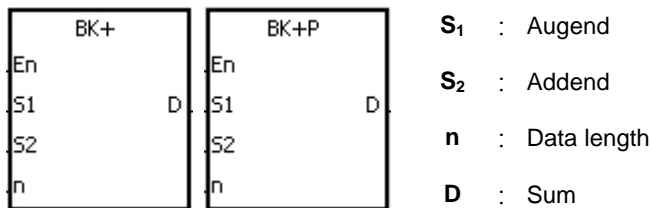
FB/FC	Instruction			Operand				Description				
FC		BK+	P	S₁, S₂, n, D				Addition of binary values in blocks				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							
n		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●	●	●	●		●	○	●				○
S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
n	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

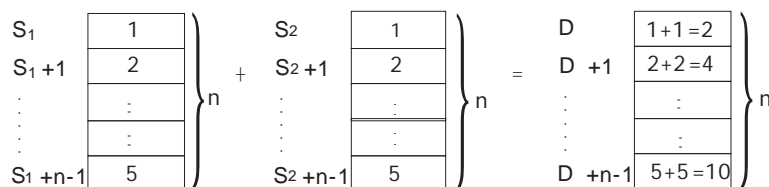
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

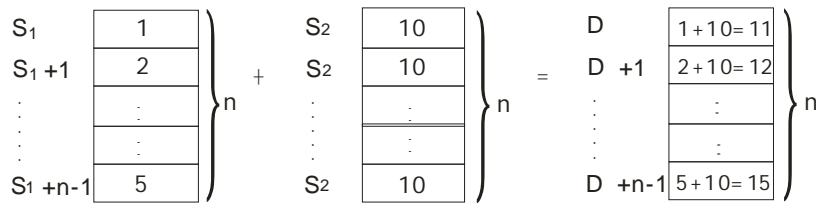


Explanation:

1. **n** pieces of data in devices starting from **S₂** are added to those in devices starting from **S₁**. The augends and the addends are binary values, and the sums are stored in **D**.
2. The operand **n** should be within the range between 1 and 256.
3. When the operation result is zero, SM600 is ON.
4. When the operation result is less than -32,768, SM601 is ON.
5. When the operation result is larger than 32,767, SM602 is ON.
6. When the operand **S₂** is a device (not a constant or a hexadecimal value):

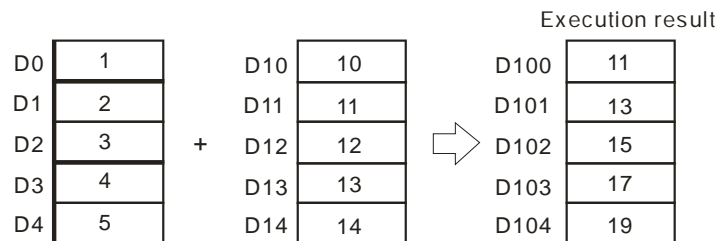
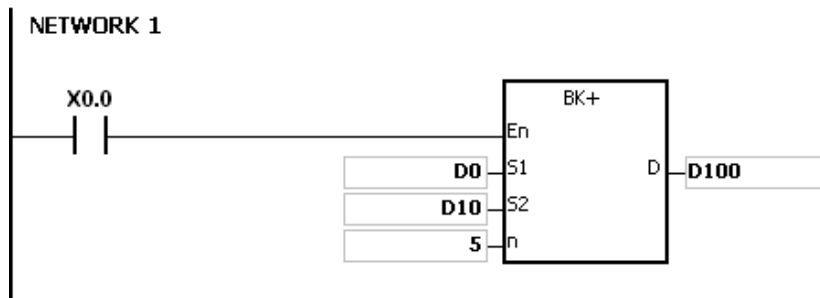


7. When the operand **S₂** is a constant or a hexadecimal value:



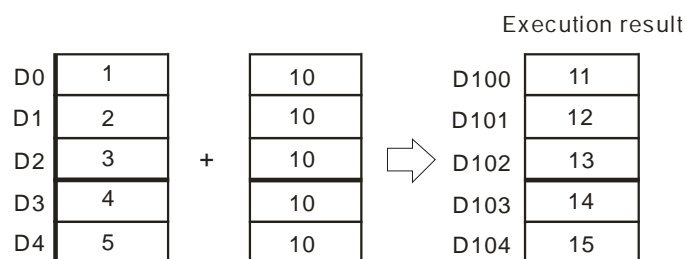
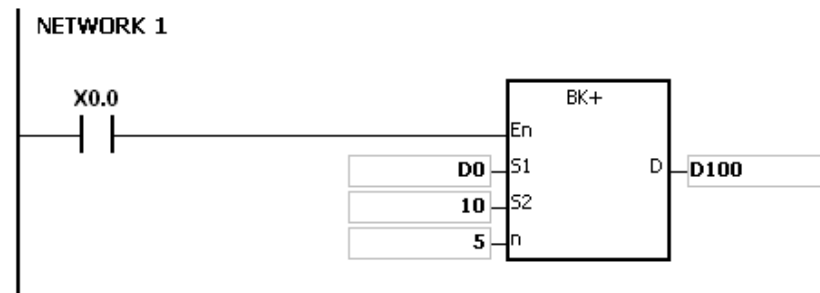
Example 1:

When X0.0 is ON, the binary values in D10~D14 are added to the binary values in D0~D4, and the sums are stored in D100~D104.



Example 2:

When X0.0 is ON, the addend 10 is added to the binary values in D0~D4, and the sums are stored in D100~D104.



Additional remark:

1. If the devices $S_1\sim S_1+n-1$, $S_2\sim S_2+n-1$, or $D\sim D+n-1$ exceed the device range, the instruction is not executed, SM is ON, and the error code in SR0 is 16#2003.
2. If $n < 1$ or $n > 256$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If $S_1\sim S_1+n-1$ overlap $D\sim D+n-1$, the instruction is not executed, SM0 is ON, and the error code in SR0 is

16#200C.

4. If $S2 \sim S2+n-1$ overlap $D \sim D+n-1$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.
5. If $S1 \sim S1+n-1$ overlap $S2 \sim S2+n-1$, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.

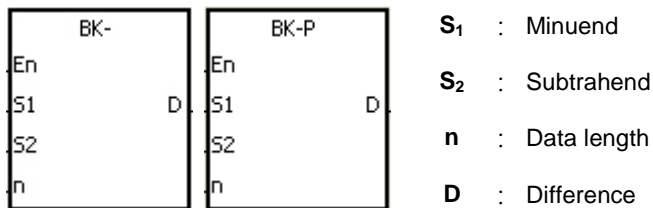
FB/FC	Instruction			Operand				Description						
FC		BK-	P	S₁, S₂, n, D				Subtraction of binary values in blocks						

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁			●					●						
S ₂			●					●						
n			●					●						
D			●					●						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●	●	●	●		●	○	●				○
S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
n	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

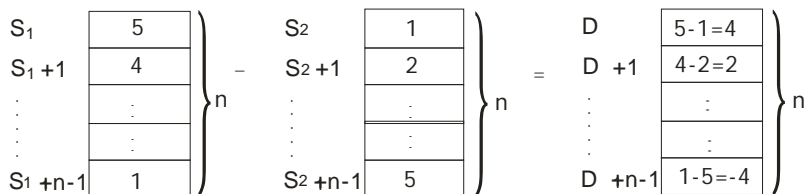
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

- n pieces of data in devices starting from S2 are subtracted from those in devices starting from S1. The minuends and the subtrahends are binary values, and the differences are stored in D.
- The operand n should be within the range between 1 and 256.
- When the operation result is zero, SM600 is ON.
- When the operation result is less than -32,768, SM601 is ON.
- When the operation result is larger than 32,767, SM602 is ON.
- When the operand S2 is a device (not a constant or a hexadecimal value):

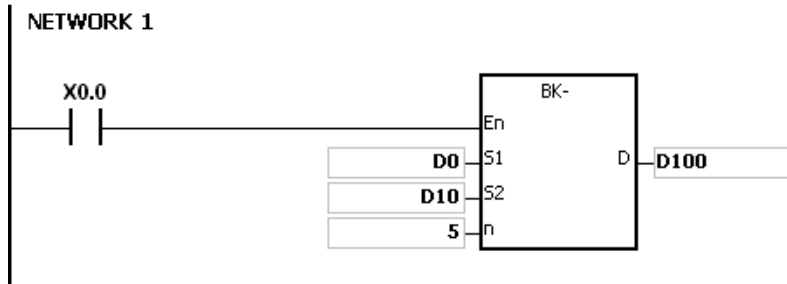


- When the operand S2 is a constant or a hexadecimal value:

$$\begin{array}{c}
 S_1 \quad \begin{array}{|c|} \hline 5 \\ \hline \end{array} \\
 S_{1+1} \quad \begin{array}{|c|} \hline 4 \\ \hline \end{array} \\
 \vdots \\
 S_{1+n-1} \quad \begin{array}{|c|} \hline 1 \\ \hline \end{array}
 \end{array}
 \left. \vphantom{\begin{array}{c} S_1 \\ S_{1+1} \\ \vdots \\ S_{1+n-1} \end{array}} \right\} n
 \quad - \quad
 \begin{array}{c}
 S_2 \quad \begin{array}{|c|} \hline 1 \\ \hline \end{array} \\
 S_2 \\
 \vdots \\
 S_2 \\
 S_2 \quad \begin{array}{|c|} \hline 1 \\ \hline \end{array}
 \end{array}
 \left. \vphantom{\begin{array}{c} S_2 \\ S_2 \\ \vdots \\ S_2 \end{array}} \right\} n
 \quad = \quad
 \begin{array}{c}
 D \quad \begin{array}{|c|} \hline 5-1=4 \\ \hline \end{array} \\
 D+1 \quad \begin{array}{|c|} \hline 4-1=3 \\ \hline \end{array} \\
 \vdots \\
 \vdots \\
 D+n-1 \quad \begin{array}{|c|} \hline 1-1=0 \\ \hline \end{array}
 \end{array}
 \left. \vphantom{\begin{array}{c} D \\ D+1 \\ \vdots \\ D+n-1 \end{array}} \right\} n$$

Example 1:

When X0.0 is ON, the binary values in D10~D14 are subtracted from the binary values in D0~D4, and the differences are stored in D100~D104.

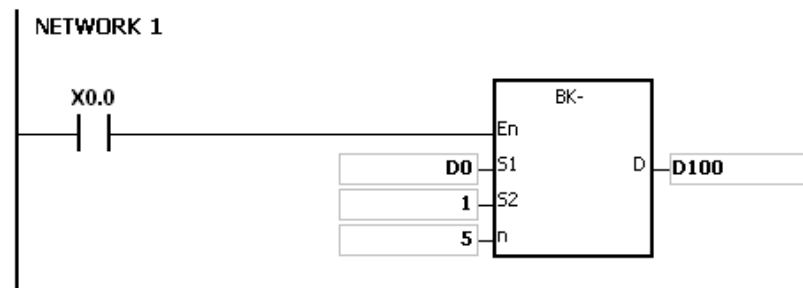


Execution result

D0	5	D10	1	D100	4	1 SM600
D1	4	D11	2	D101	2	
D2	3	D12	3	D102	0	
D3	2	D13	4	D103	-2	
D4	1	D14	5	D104	-4	

Example 2:

When X0.0 is ON, the subtrahend 1 is subtracted from the binary values in D0~D4, and the differences are stored in D100~D104.



Execution result

D0	10	1	D100	9
D1	9	1	D101	8
D2	8	1	D102	7
D3	7	1	D103	6
D4	6	1	D104	5

Additional remark:

1. If the devices S1~S1+n-1, S2~S2+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. If n<1 or n>256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If S1~S1+n-1 overlap D~D+n-1, the instruction is not executed, SM0 is ON, and the error code in SR0 is

16#200C.

4. If $S2 \sim S2+n-1$ overlap $D \sim D+n-1$, the instruction is not executed, $SM0$ is ON, and the error code in $SR0$ is 16#200C.
5. If $S1 \sim S1+n-1$ overlap $S2 \sim S2+n-1$, the instruction is not executed, $SM0$ is ON, and the error code in $SR0$ is 16#200C.

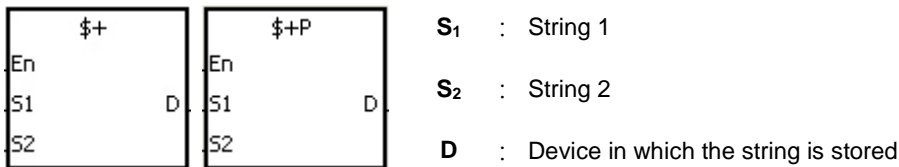
FB/FC	Instruction			Operand				Description				
FC		\$+	P	S ₁ , S ₂ , D				Linking the strings				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂														●
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●		●	●		●	○	●			○	
D	●	●			●	●		●	●		●	○	●				

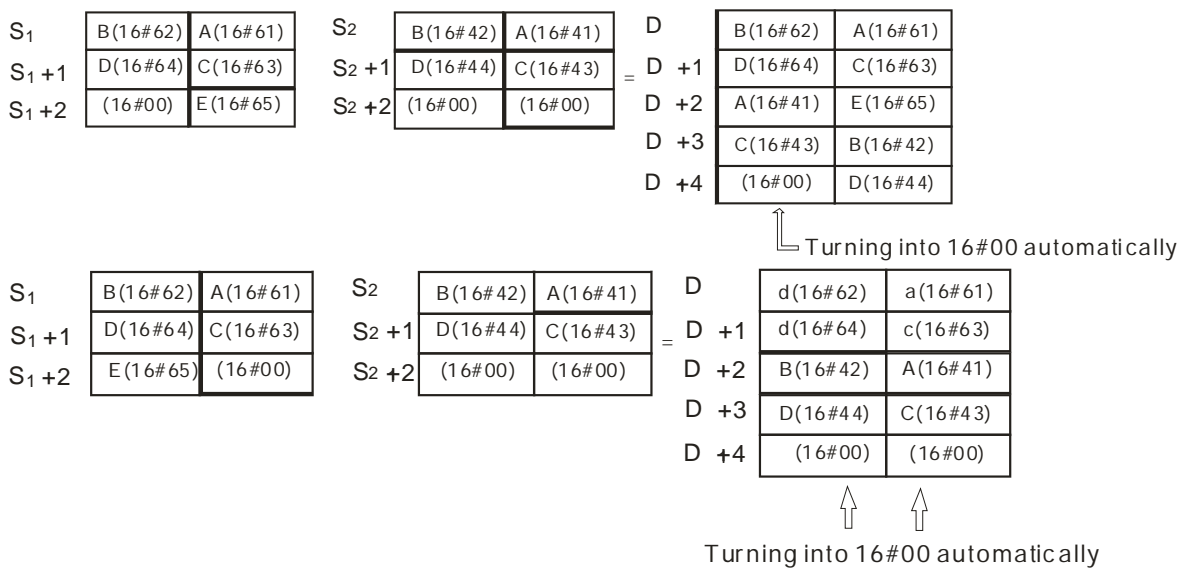
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

- When the instruction is executed, the string starting with the data in the device specified by S1 (exclusive of 16#00), and the string starting with the data in the device specified by S2 (exclusive of 16#00) are linked and moved to the operand D. Besides, the code 16#00 is added to the end of the linked string in the operand D. When the instruction is not executed, the data in D is unchanged.
- The string in the operand S1 and the string in the operand S2 are linked and moved to the operand D, as illustrated below.

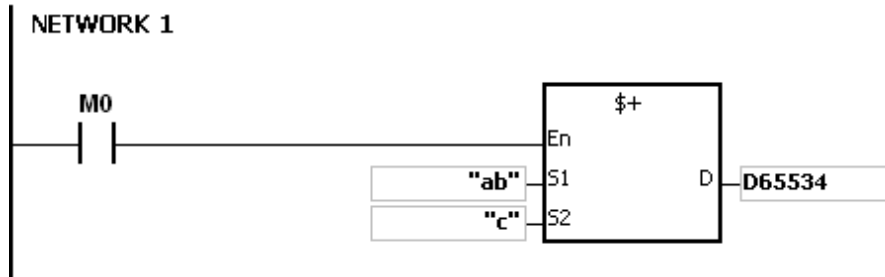


- When S1 or S2 is not a string, the code 16#00 should be added to the end of the data which is moved.
- Suppose S1 or S2 is not a string. When the instruction is executed and the first character is the code 16#00,

16#00 is still linked and moved.

Example:

Suppose **S₁** is the string “ab” and **S₂** is the string “c”. After the conditional contact M0 is enabled, the data in D65534 is 16#4241, and the data in D65535 is 16#0043.



Additional remark:

1. If S1 or S2 is a string, at most 31 characters can be moved. For a string, the number of steps=1+(the number of characters +1)/4 (The value will be rounded up to the nearest whole digit if (the number of characters +1) is not divisible by 4.).

Number of characters	1~3	4~7	8~11	12~15	16~19	20~23	24~27	28~31
Number of steps	2	3	4	5	6	7	8	9

Example: For \$+“ABCDE” D0 D100, the number of steps= 1 (instruction)+3 (string)+2 (D0)+2 (D100)=8.

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 S1 or S2 overlaps D, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200C.
4. 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.

3

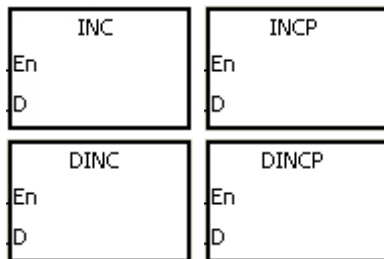
FB/FC	Instruction			Operand	Description
FC	D*	INC	P	D	Adding one to the binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



D : Destination device

Explanation:

1. One is added to the value in D.
2. Only the instruction DINC can use the 32-bit counter.
3. When the 16-bit operation is performed, 32,767 plus 1 equals -32,768. When the 32-bit operation is performed, 2,147,483,647 plus 1 equals -2,147,483,648.

Example:

When X0.0 is switched from OFF to ON, the value in D0 increases by one.



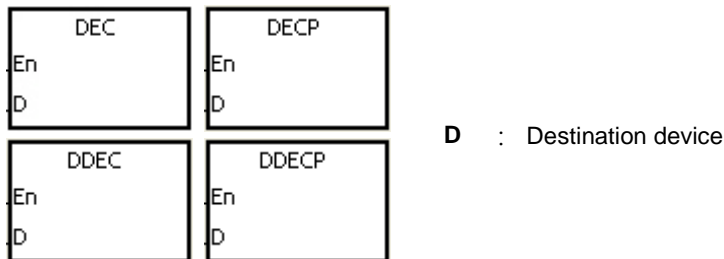
FB/FC	Instruction			Operand	Description
FC	D*	DEC	P	D	Subtracting one from the binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



Explanation:

1. One is subtracted from the value in D.
2. Only the instruction DDEC can use the 32-bit counter.
3. When the 16-bit operation is performed, -32,768 minus 1 leaves 32,767. When the 32-bit operation is performed, -2,147,483,648 minus 1 leaves 2,147,483,647.

Example:

When X0.0 is switched from OFF to ON, the value in D0 decreases by one.



FB/FC	Instruction			Operand	Description
FC	MUL16	MUL32*	P	S ₁ , S ₂ , D	16-bit binary multiplication 32-bit binary multiplication

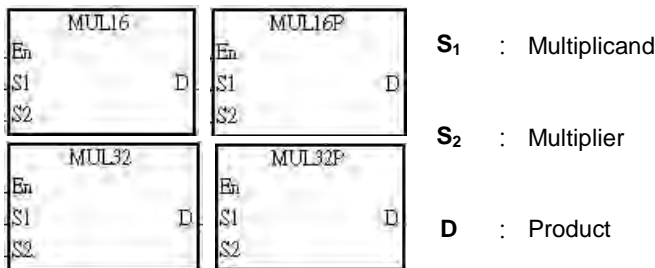
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

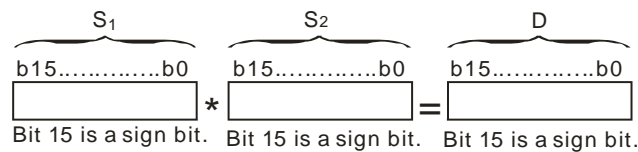
3

Graphic expression:



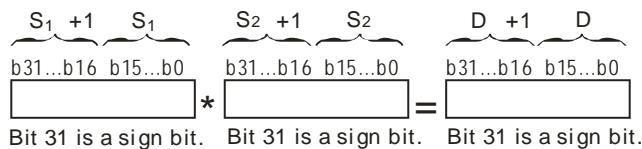
Explanation:

- The signed binary value in **S₁** is multiplied by the signed binary value in **S₂**, and the product is stored in **D**.
- Only MUL32 can use an HC device.
- 16-bit binary multiplication:



The product gotten is a 16-bit value. It is 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.

- 32-bit binary multiplication:

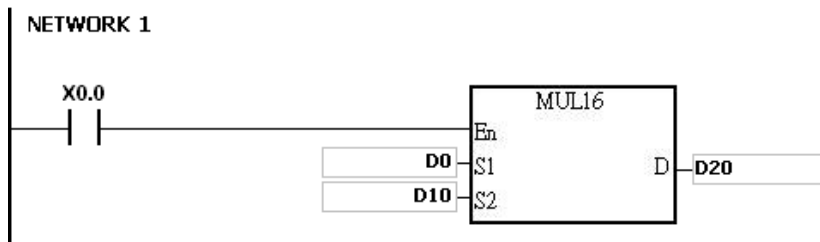


The product gotten is a 32-bit value. It is stored in (**D**, **D+1**) which is a 32-bit register. If b31 in **D** is 0, the product stored in (**D**, **D+1**) is a positive value. If b31 in **D** is 1, the product stored in (**D**, **D+1**) is a negative value.

Example:

The 16-bit value in D0 is multiplied by the 16-bit value in D10, and the product is stored in D20. Whether the

product is a positive value or a negative value 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.



$D0 \times D10 = D20$

16-bit value \times 16-bit value = 16-bit value

Additional remark:

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 will be ON, and only the low 16 bits will be written.
2. If users need the complete result of a 16-bit multiplication (a 32-bit value), they have to use API 0102 */*P. Please refer to the explanation of API 0102 */*P 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 will be ON, and only the low 32 bits will be written.
4. If users need the complete result of a 32-bit multiplication (a 64-bit value), they have to use API 0102 D*/D*P. Please refer to the explanation of API 0102 D*/D*P for more information

3

FB/FC	Instruction			Operand	Description
FC	DIV16	DIV32*	P	S₁, S₂, D	16-bit binary division 32-bit binary division

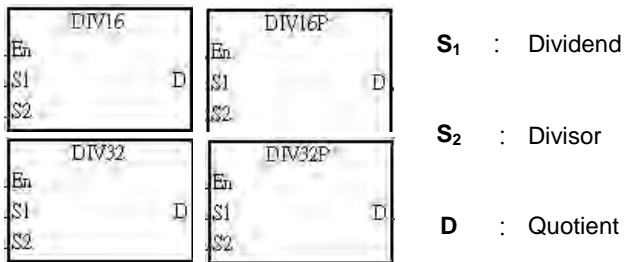
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●	●	●	●		●	○	●	●	●		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

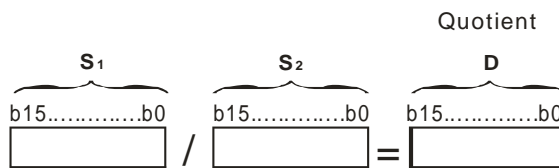
3

Graphic expression:



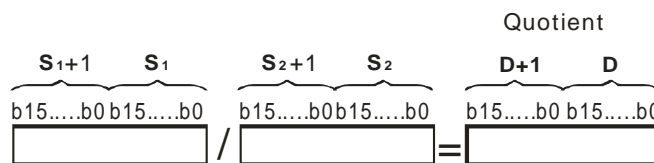
Explanation:

- The signed binary value in **S₁** is divided by the signed binary value in **S₂**. The quotient is stored in **D**.
- Only the 32-bit instruction can use an HC device.
- Sign bit=0 (Positive number); sign bit =1 (Negative number)
- 16-bit binary division:



The quotient is stored in **D**.

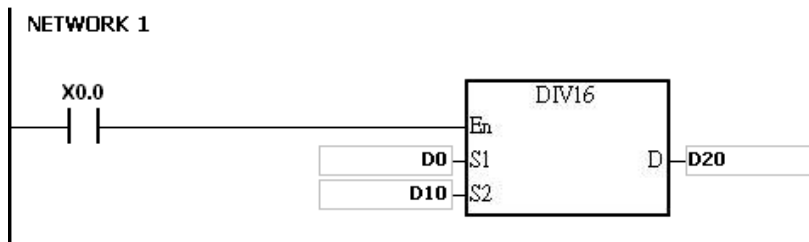
- 32-bit binary division:



D occupies two consecutive devices. The quotient is stored in (**D+1, D**).

Example:

When X0.0 is ON, the dividend in D0 is divided by the divisor in D10, and the quotient is stored in D20. Whether the quotient is a positive value or a negative value depends on the leftmost bit in D20.



Additional remark:

1. If the device used is not available, the instruction will not be executed, SM0 will be ON, and the error code stored in SR0 will be 16#2003.
2. If the divisor used is 0, the instruction will not be executed, SM0 will be ON, and the error code stored in SR0 will be 16#2012.
3. If you want to store the remainder gotten, you have to use “/” instruction (Division of binary values). Please refer to the explanation of “/” instruction for more information.

3.5 Data Conversion Instructions

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>BCD</u>	<u>DBCD</u>	–	✓	Converting the binary number into the binary-coded decimal number	5
FC	<u>BIN</u>	<u>DBIN</u>	–	✓	Converting the binary-coded decimal number into the binary number	5
FC	<u>FLT</u>	<u>DFLT</u>	–	✓	Converting the binary integer into the binary floating-point value	5
FC	<u>FLTD</u>	<u>DFLTD</u>	–	✓	Converting the binary integer into the 64-bit floating-point value	5
FC	<u>INT</u>	<u>DINT</u>	–	✓	Converting the 32-bit floating-point value into the binary integer	5
FC	–	<u>FINT</u>	<u>DFINT</u>	✓	Converting the 64-bit floating-point value into the binary integer	5
FC	<u>MMOV</u>	–	–	✓	Converting the 16-bit value into the 32-bit value	5
FC	<u>RMOV</u>	–	–	✓	Converting the 32-bit value into the 16-bit value	5
FC	<u>GRY</u>	<u>DGRY</u>	–	✓	Converting the binary number into the Gray code	5
FC	<u>GBIN</u>	<u>DGBIN</u>	–	✓	Converting the Gray code into the binary number	5
FC	<u>NEG</u>	<u>DNEG</u>	–	✓	Two's complement	3
FC	–	<u>FNEG</u>	–	✓	Reversing the sign of the 32-bit floating-point value	3
FC	–	<u>FBCD</u>	–	✓	Converting the binary floating-point value into the decimal floating-point value	5
FC	–	<u>FBIN</u>	–	✓	Converting the decimal floating-point value into the binary floating-point value	5
FC	<u>BKBCD</u>	–	–	✓	Converting the binary values in blocks into the binary-coded decimal numbers in blocks	7
FC	<u>BKBIN</u>	–	–	✓	Converting the binary values in blocks into the binary-coded decimal numbers in blocks	7
FC	<u>SCAL</u>	–	–	✓	Scale value operation	9
FC	<u>SCLP</u>	<u>DSCLP</u>	–	✓	Parameter type of scale value operation	9
FC	<u>LINE</u>	<u>DLINE</u>	–	✓	Converting a column of data into a line of data	7
FC	<u>COLM</u>	<u>DCOLM</u>	–	✓	Converting a line of data into a column of data	7

FB/FC	Instruction			Operand	Description
FC	D*	BCD	P	S, D	Converting the binary number into the binary-coded decimal number

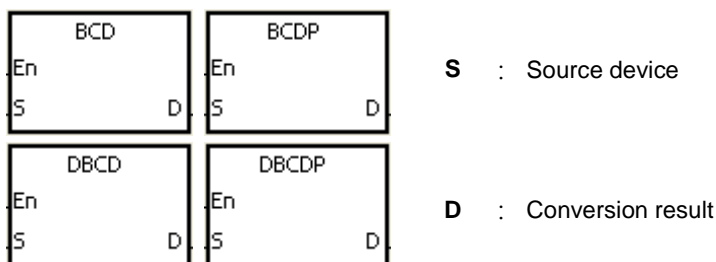
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:

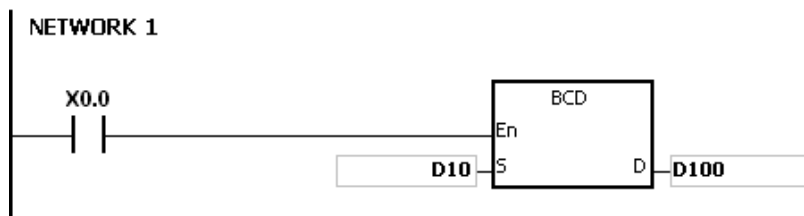


Explanation:

1. The binary value in S is converted into the binary-coded decimal value, and the conversion result is stored in D.
2. Only the instruction DBCD can use the 32-bit counter.
3. The four fundamental operations of arithmetic in the PLC, the instruction INC, and the instruction DEC all involve binary values. To show the decimal value on the display, you can use the instruction BCD to convert the binary value into the binary-coded decimal value

Example:

When X0.0 is ON, the binary value in D10 is converted into the binary-code decimal value, and the conversion result is stored in D100.



If D10=16#04D2=1234, the conversion result will be that D100=16#1234.

Additional remark:

1. If the conversion result is out of the range between 0 and 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 within the range between 0 and 9.).

2. If the conversion result is out of the range between 0 and 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 within the range between 0 and 9.).

FB/FC	Instruction			Operand	Description
FC	D*	BIN	P	S, D	Converting the binary-coded decimal number into the binary number

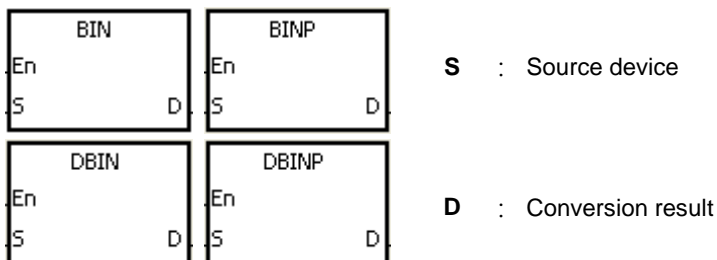
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:

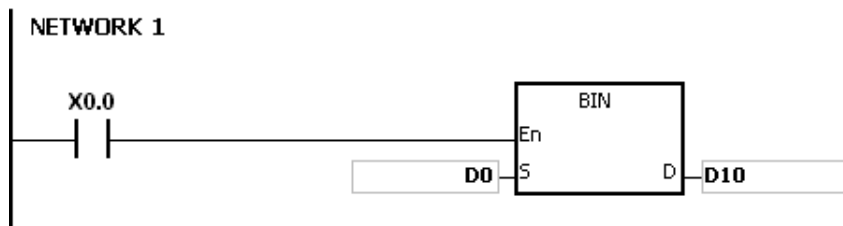


Explanation:

1. The binary-coded decimal value in S is converted into the binary value, and the conversion result is stored in D.
2. The 16-bit binary-coded decimal value in S should be within the range between 0 and 9,999, and the 32-bit binary-coded decimal value in S should be within the range between 0 and 99,999,999.
3. Only the 32-bit instructions can use the 32-bit counter.
4. Constants and hexadecimal values are converted into binary values automatically. Therefore, you do not need to use the instruction.

Example:

When X0.0 is ON, the binary-coded decimal value in D0 is converted into the binary value, and the conversion result is stored in D10.

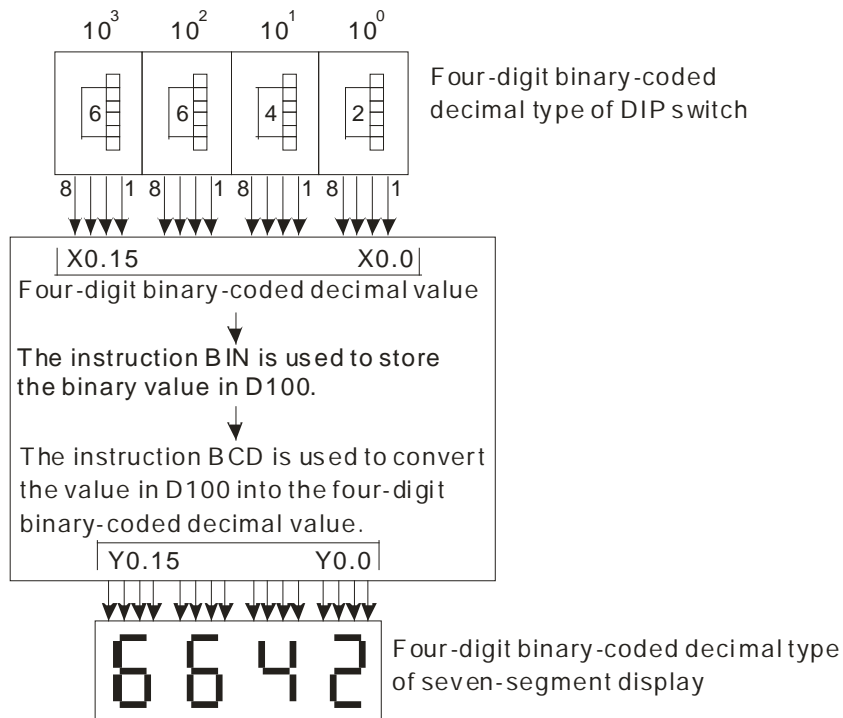
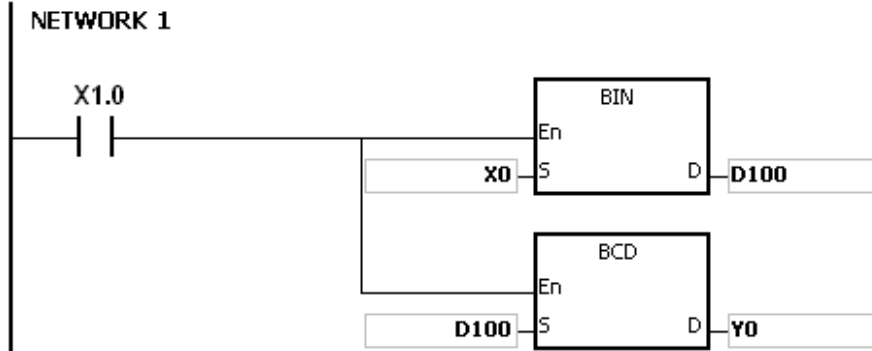


Additional remark:

1. If the value in S is not the binary-coded decimal value, the 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 within the range between 0 and 9.).

2. The application of the instructions BCD and BIN:

- Before the value of the binary-coded decimal type of DIP switch is read into the PLC, you have to use the instruction BIN 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, you have to use the instruction BCD to convert the data into the binary-coded decimal value before the data is sent to the seven-segment display.
- When X1.0 is ON, the binary-coded decimal value in X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D100. Subsequently, the binary value in D100 is converted into the binary-coded decimal value, and the conversion result is stored in Y0.0~Y0.15.



FB/FC	Instruction			Operand	Description
FC	D*	FLT	P	S, D	Converting the binary integer into the binary floating-point value

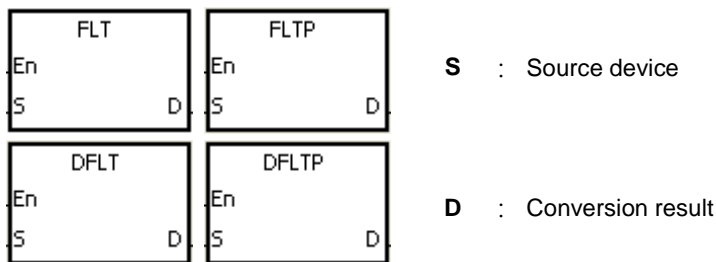
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D			●/●*					●/●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:



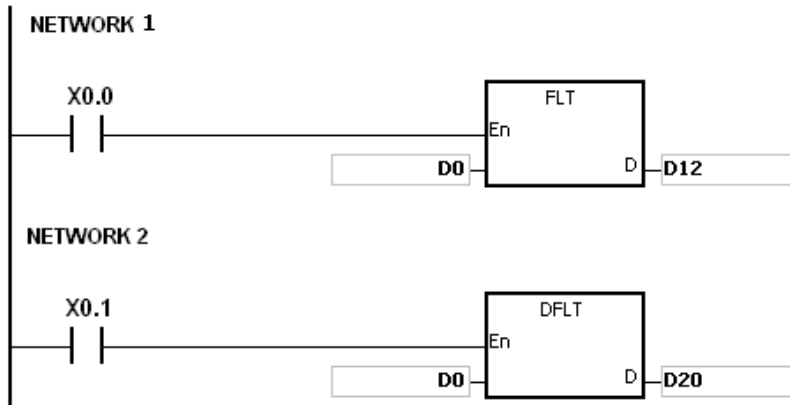
Explanation:

- The instruction is used to convert the binary integer into the single-precision floating-point value.
- The operand S used in the instruction FLT can not be the 32-bit counter.
- The source device S used in the instruction FLT occupies one register, and D used in FLT occupies two registers.
- 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 which can be represented by the maximum floating-point value, SM602 is ON, and the maximum floating-point value is stored in D.
 - When the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point value, SM601 is ON, and the minimum floating-point value is stored in D.
 - When the conversion result is zero, SM600 is ON.

Example 1:

- When X0.0 is ON, the binary integer in D0 is converted into the single-precision floating-point value, and the conversion result is stored in (D13, D12).
- When X0.1 is ON, the binary integer in (D1, D0) is converted into the single-precision floating-point value, and the conversion result is stored in (D21, D20).
- Suppose the value in D0 is 10. When X0.0 is ON, 10 is converted into the single-precision floating-point value 16#41200000, and 16#41200000 is stored in the 32-bit register (D13, D12).

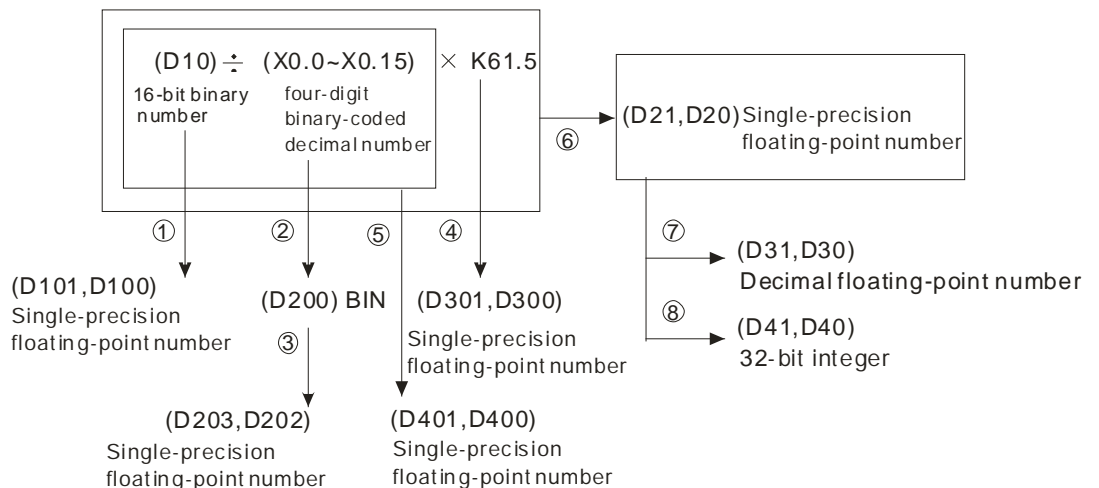
- Suppose the value in the 32-bit register (D1, D0) is 100,000. When X0.1 is ON, 100,000 is converted into the single-precision floating-point value 16#47C35000, 16#47C35000 is stored in the 32-bit register (D21, D20).



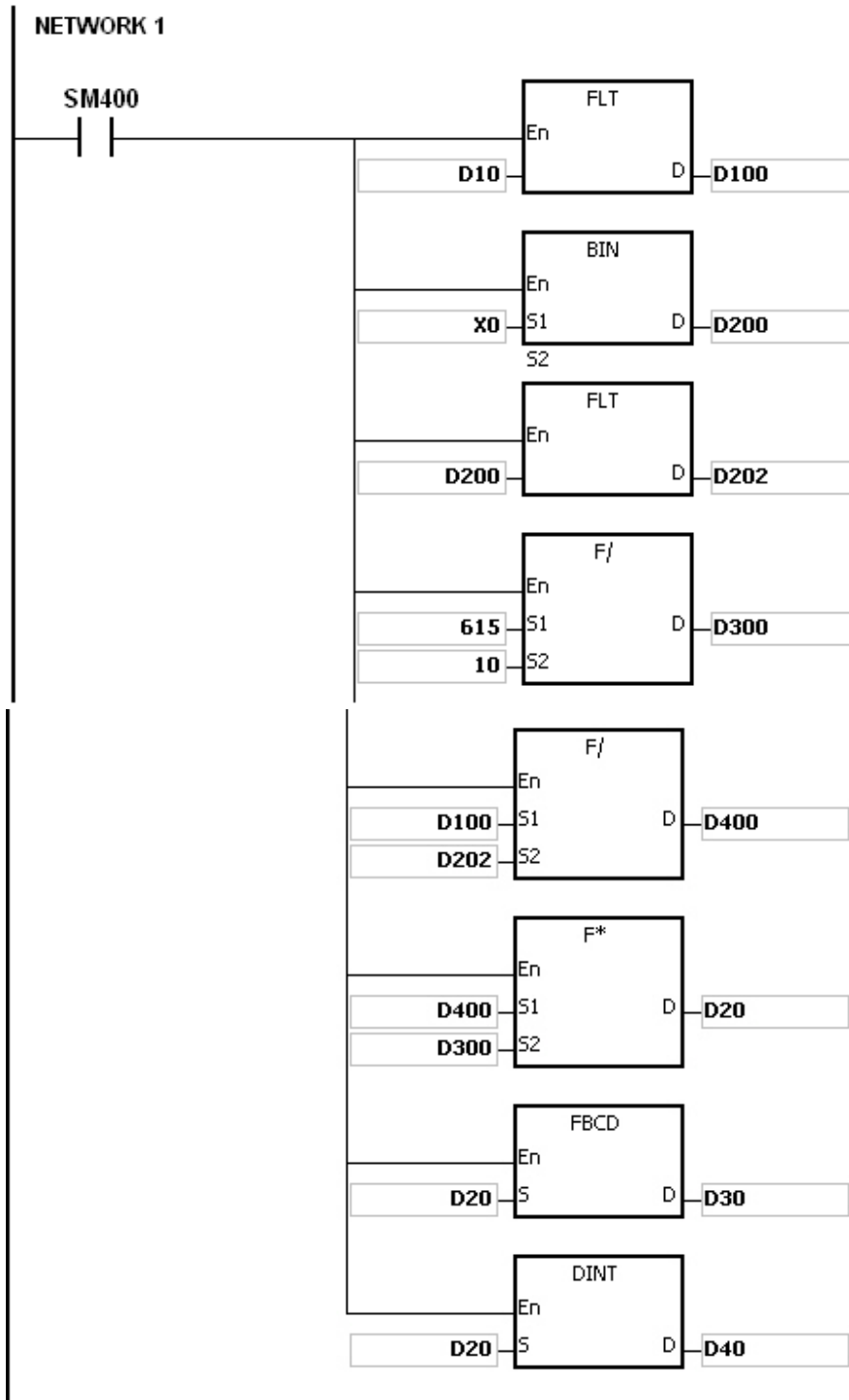
Example 2:

You can use the applied instructions to perform the following calculation.

- The binary integer in D10 is converted into the single-precision floating-point value, and the conversion result is stored in (D101, D100).
- The binary-coded decimal value in X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D200.
- The binary integer in D200 is converted into the single-precision floating-point value, and the conversion result is stored in (D203, D202).
- The constant 615 is divided by the constant 10, and the quotient which is the single-precision floating-point value is stored in (D301, D300).
- The single-precision floating-point value in (D101, D100) is divided by the single-precision floating-point value in (D203, D202), and the quotient which is the single-precision floating-point value is stored in (D401, D400).
- The single-precision floating-point value in (D401, D400) is multiplied by the single-precision floating-point value in (D301, D300), and the product which is the single-precision floating-point value is stored in (D21, D20).
- The single-precision floating-point value in (D21, D20) is converted into the decimal floating-point value, and the conversion result is stored in (D31, D30).
- The single-precision floating-point value in (D21, D20) is converted into the binary integer, and the conversion result is stored in (D41, D40).



3

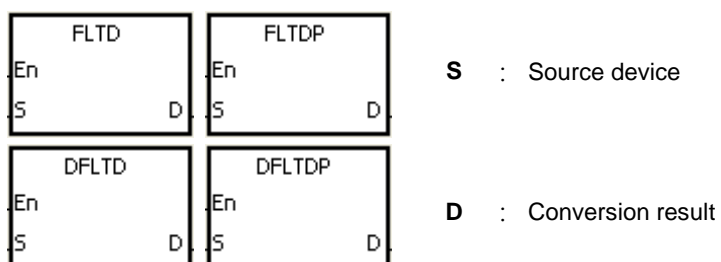


FB/FC	Instruction			Operand	Description
FC	D*	FLTD	P	S, D	Converting the binary integer into the 64-bit floating-point value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

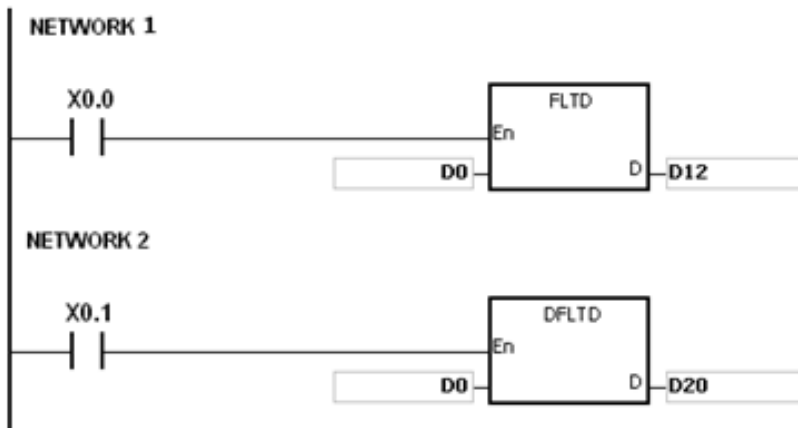
Graphic expression:**Explanation:**

- When the instruction is executed, the binary integer is converted into the double-precision floating-point value.
- The operand S used in the instruction FLTD can not be the 32-bit counter.
- The source device S used in the instruction FLTD occupies one register, and D used in FLTD occupies four registers.
- The source device S used in the instruction DFLTDP occupies two registers, and D used in DFLTDP occupies four registers.
- When the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point value, SM602 is ON, and the maximum floating-point value is stored in D.
- When the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point value, SM601 is ON, and the minimum floating-point value is stored in D.
- When the conversion result is zero, SM600 is ON.

Example:

- When X0.0 is ON, the 16-bit binary integer in D0 is converted into the double-precision floating-point value, and the conversion result is stored in (D15, D14, D13, D12).
- When X0.1 is ON, the 32-bit binary integer in (D1, D0) is converted into the double-precision floating-point value, and the conversion result is stored in (D23, D22, D21, D20).
- Suppose the 16-bit binary integer in D0 is 10. When X0.0 is ON, 10 is converted into 16#4024000000000000, and 16#4024000000000000 is stored in the 64-bit register (D15, D14, D13, D12).

4. Suppose the 32-bit binary integer in (D1, D0) is 100,000. When X0.1 is ON, 100,000 is converted into 16#40F86A0000000000, and 16#40F86A0000000000 is stored in the 64-bit register (D23, D22, D21, D20).

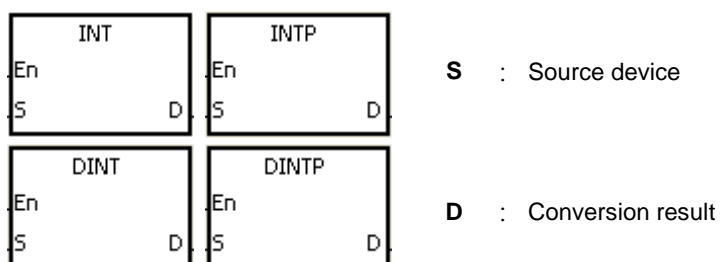


FB/FC	Instruction			Operand	Description
FC	D*	INT	P	S, D	Converting the 32-bit floating-point value into the binary integer

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●			●	○	●			○	
D	●	●			●	●	●	●			●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

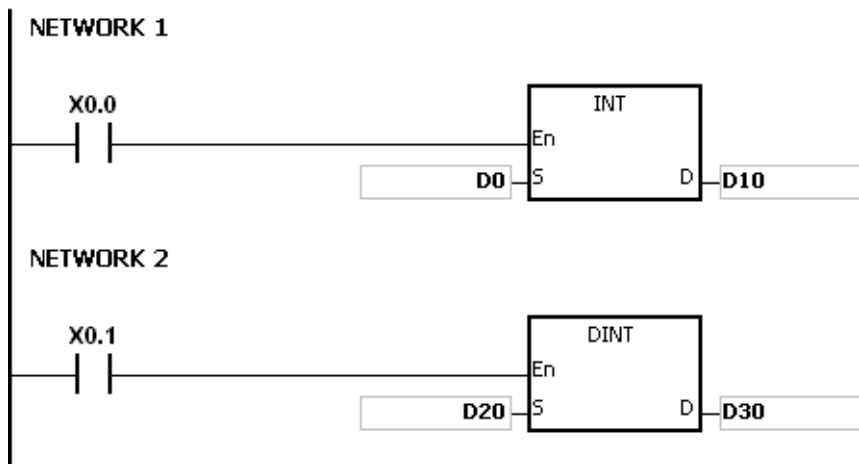
Graphic expression:**Explanation:**

- The single-precision floating-point value in the register specified by S is converted into the binary integer. The binary floating-point value is rounded down to the nearest whole digit, and becomes the binary integer. The binary integer is stored in the register specified by D.
- The source device S used in the instruction INT occupies two registers, and D used in INT occupies one register.
- The source device S used in the instruction DINT occupies two registers, and D used in DINT also occupies two registers.
- The operand D used in the instruction INT can not be the 32-bit counter.
- The instruction INT is the opposite of the instruction FLT.
- When the conversion result is zero, SM600 is ON.
- During the conversion, if the floating-point value is rounded down to the nearest whole digit, SM601 will be ON.
- When the conversion result is out of the range, SM602 is ON.
- For the 16-bit instruction, the range of conversion results is between -32,768 and 32,767.
- For the 32-bit instruction, the range of conversion results is between -2,147,483,648 and 2,147,483,647.

Example:

- When X0.0 is ON, the single-precision floating-point value in (D1, D0) is converted into the binary integer, and the conversion result is stored in D10. The binary floating-point value is rounded down to the nearest whole digit.

2. When X0.1 is ON, the single-precision floating-point value in (D21, D20) is converted into the binary integer, and the conversion result is stored in (D31, D30). The binary floating-point value is rounded down to the nearest whole digit.



3 Additional remark:

If the value in S is out of the range of values which can be represented by the floating-point values, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

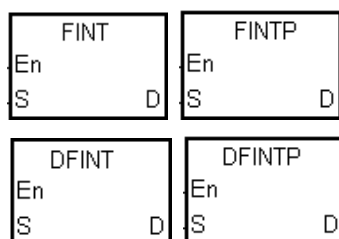
FB/FC	Instruction			Operand	Description
FC	D*	FINT	P	S, D	Converting the 64-bit floating-point value into the binary integer

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S : Source device

D : Conversion result

Explanation:

1. The double-precision floating-point value in the register specified by S is converted into the binary integer. The binary floating-point value is rounded down to the nearest whole digit, and becomes the binary integer. The binary integer is stored in the register specified by D.
2. The source device S used in the instruction FINT occupies four registers, and D used in FINT occupies one register.
3. The source device S used in the instruction DFINT occupies four registers, and D used in DFINT occupies two registers.
4. The operand D used in the instructions FINT and FLTP can not be the 32-bit counter.
5. The instruction FINT is the opposite of the instruction FLTD.
6. When the conversion result is zero, SM600 is ON.
7. During the conversion, if the floating-point value is rounded down to the nearest whole digit, SM601 will be ON.
8. When the conversion result is out of the range, SM602 is ON.

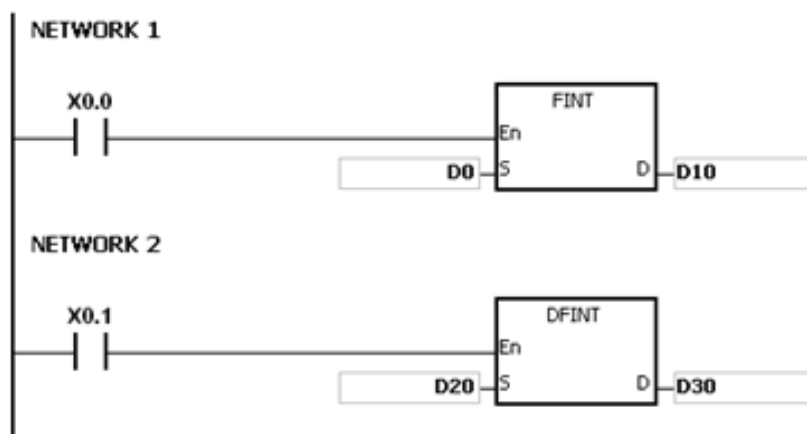
For the 32-bit instruction, the range of conversion results is between -32,768 and 32,767.

For the 64-bit instruction, the range of conversion results is between -2,147,483,648 and 2,147,483,647.

Example:

1. When X0.0 is ON, the double-precision floating-point value in (D3, D2, D1, D0) is converted into the binary integer, and the conversion result is stored in D10. The binary floating-point value is rounded down to the nearest whole digit.

- When X0.1 is ON, the double-precision floating-point value in (D23, D22, D21, D20) is converted into the binary integer, and the conversion result is stored in (D31, D30). The binary floating-point value is rounded down to the nearest whole digit.



3

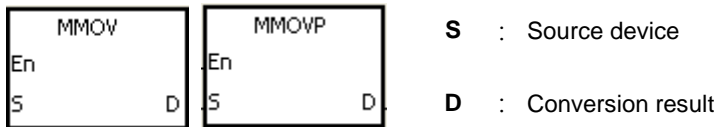
FB/FC	Instruction		Operand				Description					
FC		MMOV	P	S, D				Converting the 16-bit value into the 32-bit value				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

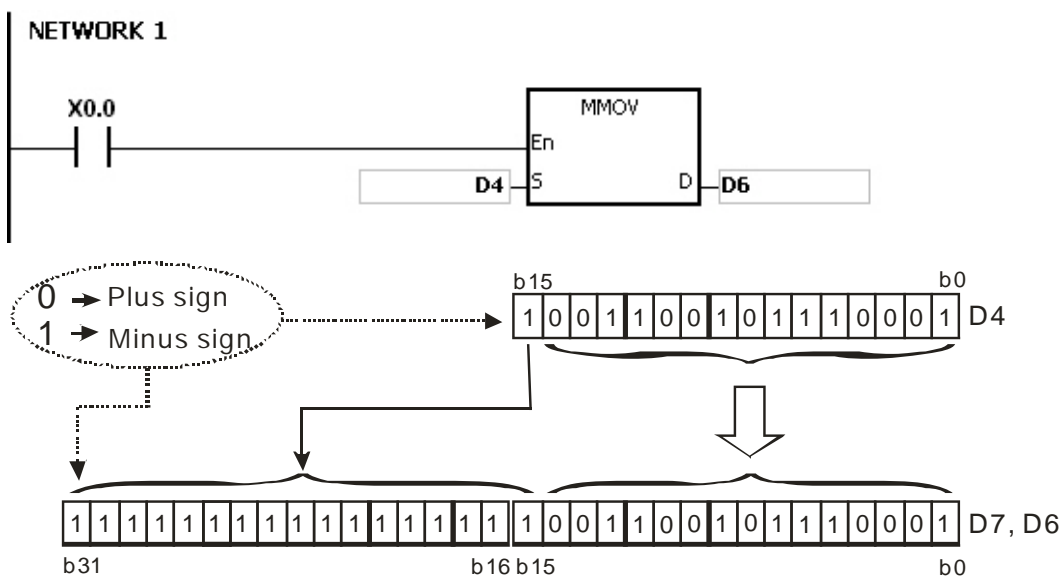


Explanation:

The data in the 16-bit device **S** is transmitted to the 32-bit device **D**. The sign bit which is specified is copied repeatedly to the destination.

Example:

When X0.0 is ON, the value of b15 in D4 is transmitted to b15~b31 in (D7, D6). The data in (D7, D6) becomes a negative value.



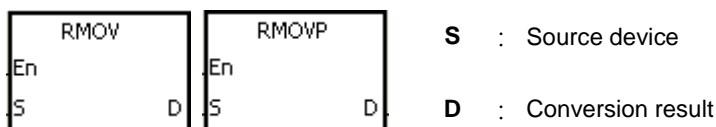
FB/FC	Instruction		Operand		Description
FC	RMOV	P	S, D		Converting the 32-bit value into the 16-bit value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

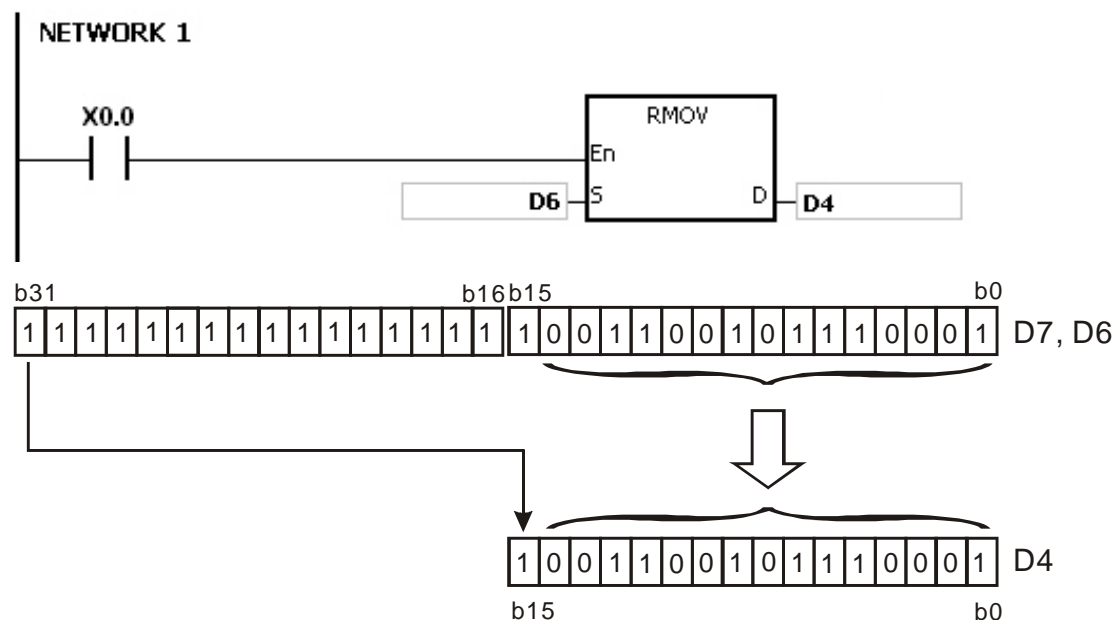


Explanation:

The data in the 32-bit device S is transmitted to the 16-bit device D. The sing bit which is specified is retained.

Example:

When X0.0 is ON, the value of b31 in D7 is transmitted to b15 in D4, the values of b0~b14 are transmitted to the corresponding bits, and the values of b15~b30 are ignored.



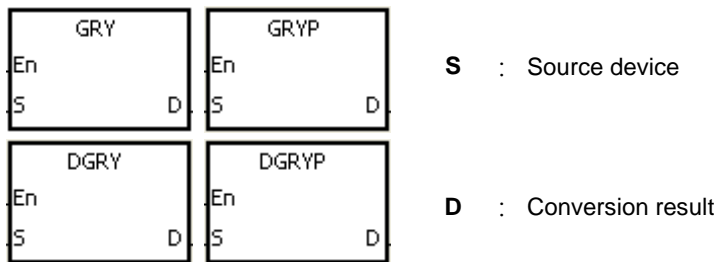
FB/FC	Instruction			Operand	Description
FC	D*	GRY	P	S, D	Converting the binary number into the Gray code

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●			●	○	●	○	○		
D	●	●			●	●	●	●			●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



Explanation:

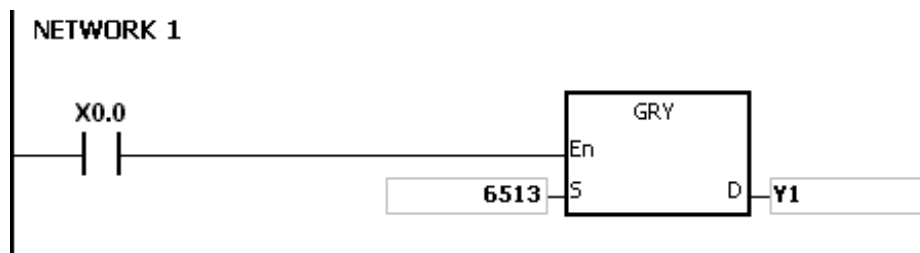
1. The binary value in the device specified by S is converted into the Gray code, and the conversion result is stored in the device specified by D.
2. Only the instruction DGRY can use the 32-counter.
3. The value in the operand S should be within the available range.

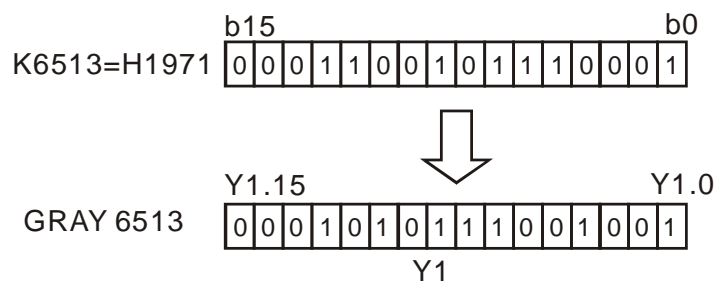
The value in the operand S used in the 16-bit instruction should be within the range between 0 and 32,767.

The value in the operand S used in the 32-bit instruction should be within the range between 0 and 2,147,483,647.

Example:

When X0.0 is ON, the constant 6513 is converted into the Gray code, and the conversion result is stored in Y1.0~Y1.15.





Additional remark:

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.

FB/FC	Instruction			Operand	Description
FC	D	GBIN	P	S, D	Converting the Gray code into the binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



Explanation:

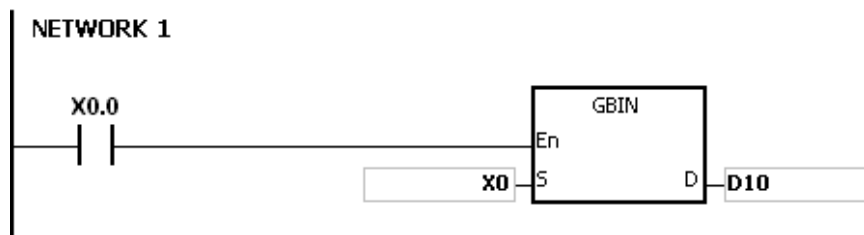
1. The Gray code in the device specified by S is converted into the binary value, and the conversion result is stored in the device specified by D.
2. The instruction is used to convert the Gary code in the absolute position encoder which is connected to the input terminal of the PLC to the binary value, and the conversion result is stored in the register which is specified.
3. The value in the operand S should be within the available range.

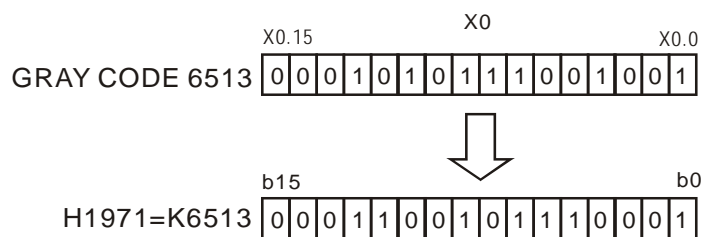
The value in the operand S used in the 16-bit instruction should be within the range between 0 and 32,767.

The value in the operand S used in the 32-bit instruction should be within the range between 0 and 2,147,483,647.

Example:

When X0.0 is ON, the Gary code in the absolute position encoder which is connected to the inputs X0.0~X0.15 is converted into the binary value, and the conversion result is stored in D10.



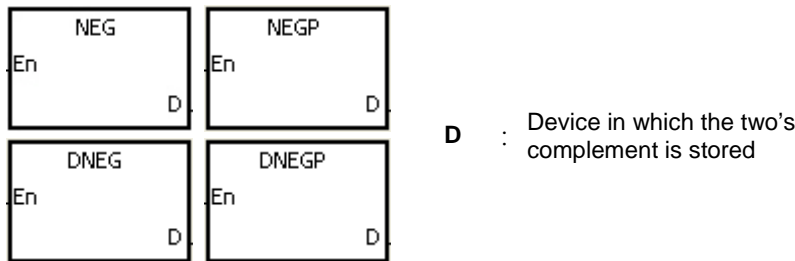


Additional remark:

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.

FB/FC	Instruction			Operand							Description						
FC	D*	NEG	P	D							Two's complement						
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING			
D		●	●*				●	●*									
Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				
Pulse instruction							16-bit instruction					32-bit instruction					
AH Motion CPU							AH Motion CPU					AH Motion CPU					

Graphic expression:

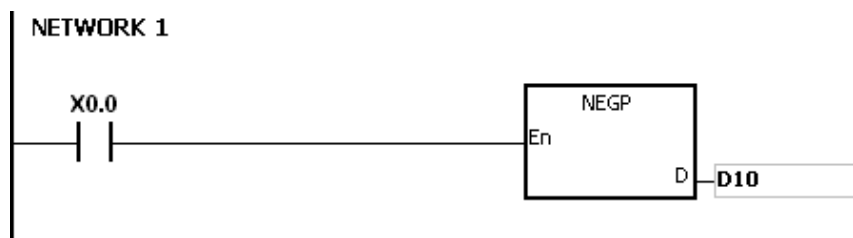


Explanation:

1. The instruction is used to convert the negative binary value into the absolute value.
2. Only the instruction DNEG can use the 32-bit counter.
3. Generally, the pulse instructions NEGP and DNEGP are used.

Example 1:

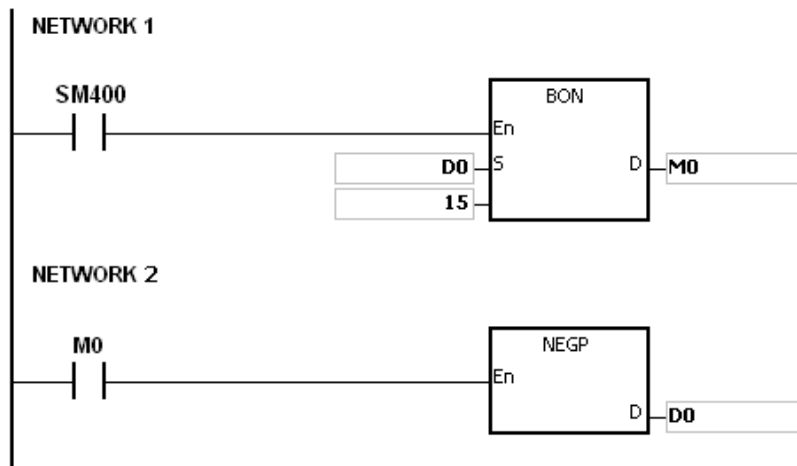
When X0.0 is switched from OFF to ON, all bits in D0 are inverted (0 becomes 1, and 1 becomes 0), and 1 is added to the result. The final value is stored in the original register D10.



Example 2:

The absolute value of the negative value:

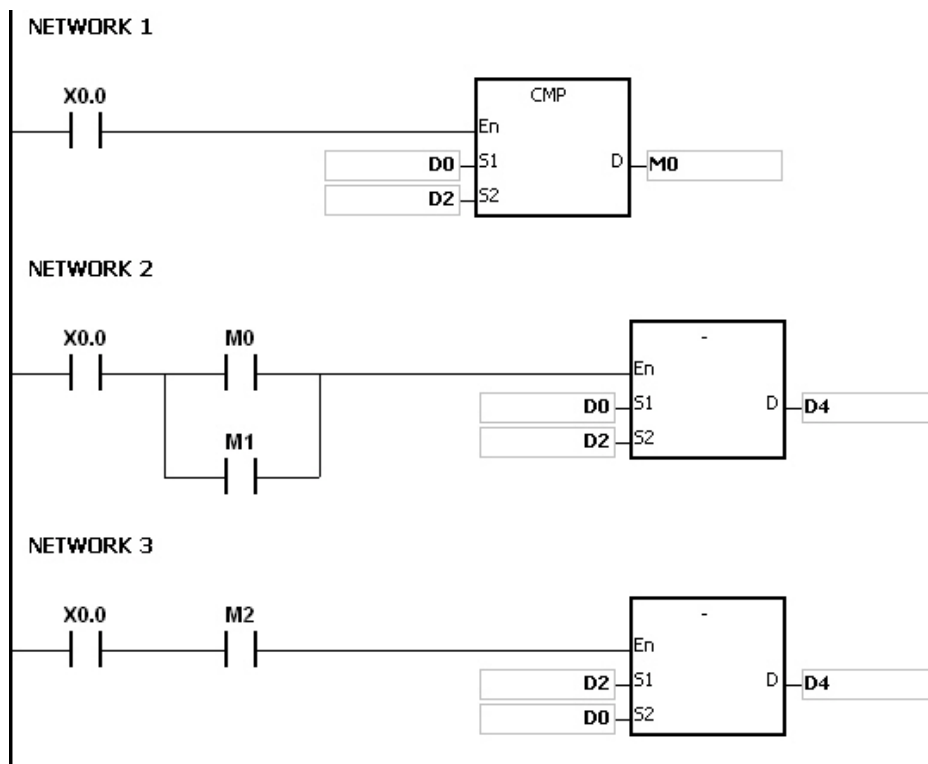
1. When the value of the 15th bit in D0 is 1, M0 is ON. (The value in D0 is a negative value.)
2. When M0 is ON, the instruction NEG is used to obtain the two's complement of the negative value in D0. (The corresponding positive value is obtained.)



Example 3:

The absolute value of the difference between two values:

1. Suppose X0.0 is ON.
2. When the value in D0 is greater than that in D2, M0 is ON.
3. When the value in D0 is equal to that in D2, M1 is ON.
4. When the value in D0 is less than that in D2, M2 is ON.
5. The value in D4 is a positive value.



Additional remark:

The 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 value of the highest in the register is 0, the data is a positive value. If it is 1, the data is a negative value.
2. The negative value can be converted into its absolute value by means of the instruction NEG.

(D0)=2

0	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	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0)=0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

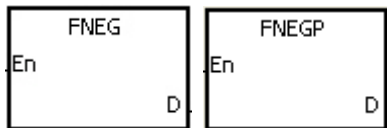
FB/FC	Instruction		Operand		Description
FC	FNEG	P	D		Reversing the sign of the 32-bit floating-point value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



D : Device in which the sign of the value is reversed

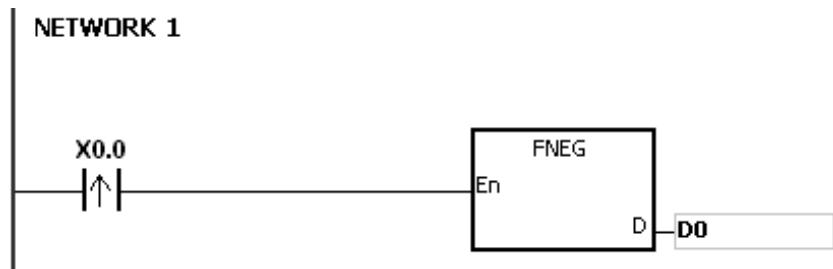
Explanation:

The sign of the single-precision floating-point value in the device **D** is reversed.

Example:

Before the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000. When X0.0 is switched from OFF to ON, the sign of the single-precision floating-point value in (D1, D0) is reversed. 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 X0.0 is switched from OFF to ON, the sign of the single-precision floating-point value in (D1, D0) is reversed. In other words, after the instruction is executed, the value in (D1, D0) is the negative value 16#AE0F9000.



FB/FC	Instruction		Operand		Description
FC		FBCD	P	S, D	Converting the binary floating-point value into the decimal floating-point value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

- The single-precision floating-point value in the register specified by S is converted into the decimal floating-point value, and the conversion result is stored in the register specified by D.
- The floating-point operation in the PLC is based on the single-precision floating-point values, and the instruction FBCD is used to convert the single-precision floating-point value into the decimal floating-point value.
- The Flags: SM600 (zero flag), SM601 (borrow flag), and SM602 (carry flag)

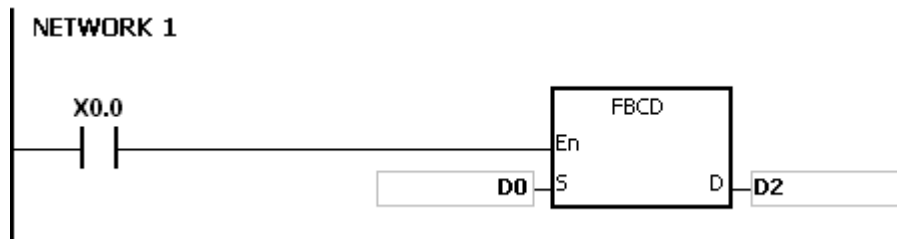
When the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point value, SM602 is ON.

When the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point value, SM601 is ON.

When the conversion result is zero, SM600 is ON.

Example:

When X0.0 is ON, the single-precision floating-point value in (D1, D0) is converted into the decimal floating-point value, and the conversion result is stored in (D3, D2).



Binary floating-point number

D 1	D 0
-----	-----

 Real number: 23 bits; Exponent: 8 bits; sign: 1 bit



Decimal floating-point number

D 3	D 2
-----	-----

 Mathematical form $\Rightarrow [D2] \times 10^{[D3]}$

Additional remark:

If the value in **S** is out of the range of values which can be represented by the floating-point values, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

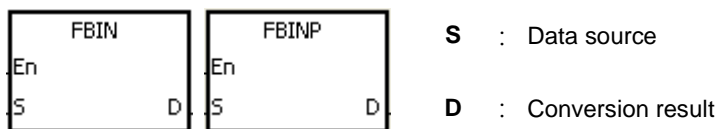
FB/FC	Instruction		Operand		Description
FC		FBIN	P	S, D	Converting the decimal floating-point value into the binary floating-point value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●			○	
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

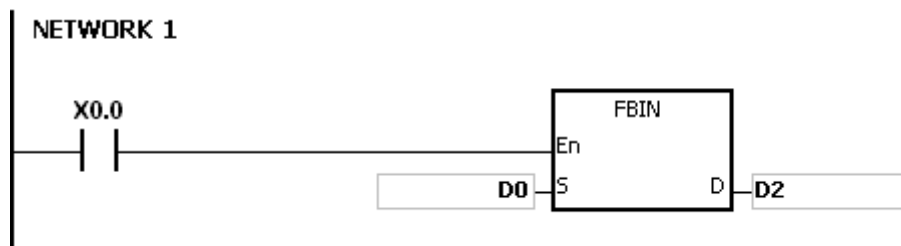


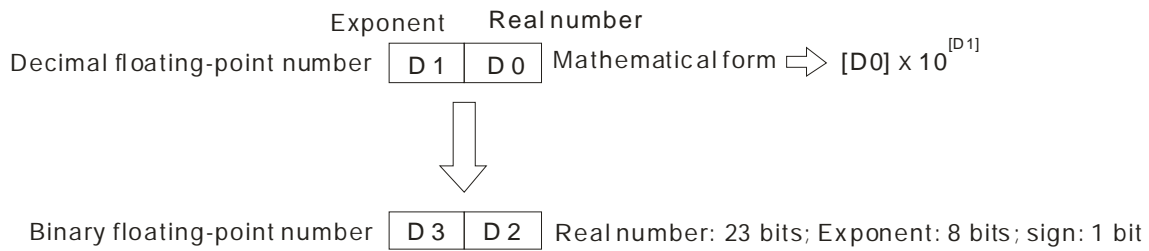
Explanation:

1. The decimal floating-point value in the register specified by S is converted into the single-precision floating-point value, and the conversion result is stored in the register specified by D.
2. Suppose the value in S is 1234, and the value in S+1 is 3. The value in S is converted into 1.234x10⁶.
3. The value in D should be a single-precision floating-point value, and the values in S and S+1 represent the decimal real number and the decimal exponent respectively.
4. The instruction FBIN is used to convert the decimal floating-point value into the single-precision floating-point value.
5. The real number of decimal floating-point values range from -9,999 to +9,999, the exponents of decimal floating-point values range from -41 to +35, and the practical range of decimal floating-point values in PLC is between ±1175x10⁻⁴¹ and ±3402x10⁺³⁵. When the operation result is zero, SM600 is ON.

Example 1:

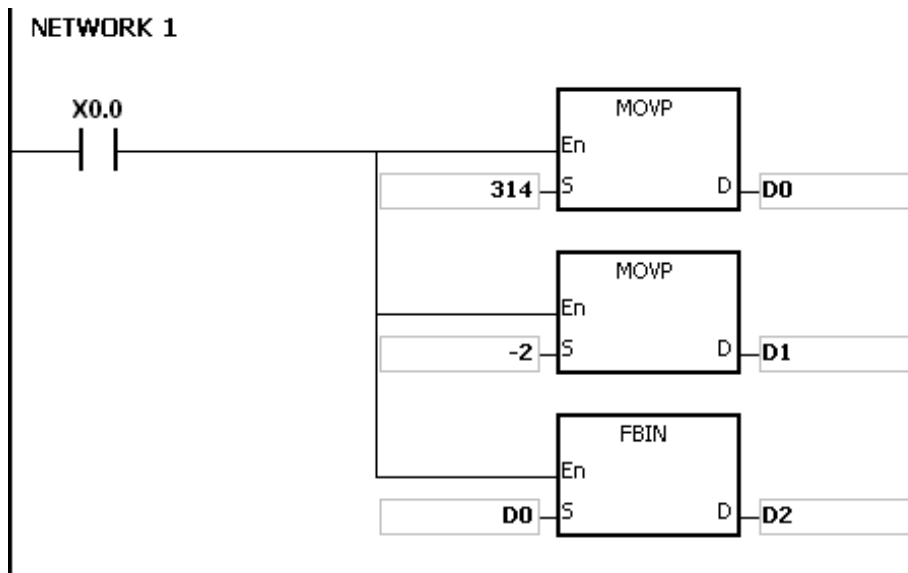
When X0.0 is ON, the decimal floating-point value in the register in (D1, D0) is converted into the single-precision floating-point value, and the conversion result is stored in (D3, D2).





Example 2:

1. Before the floating-point operation is performed, you have to use the instruction FLT to convert the binary integer into the single-precision floating-point value. The premise of the conversion is that the value converted in the binary integer. However, the instruction FBIN can be used to convert the floating-point value into the single-precision floating-point value.
2. When X0.0 is ON, K314 and K-2 are moved to D0 and D1 respectively, and combine into the decimal floating-point value ($3.14=314 \times 10^{-2}$).



Additional remark:

If the real number of the decimal floating-point value in the operand **S** is not within the range between -9,999 and +9,999, or if the exponent of the decimal floating-point value in the operand **S** is not within the range between -41 and +35, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2013.

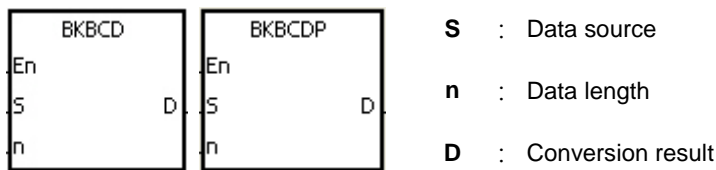
FB/FC	Instruction		Operand				Description
FC	BKBCD	P	S, n, D				Converting the binary values in blocks into the binary-coded decimal numbers in blocks

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

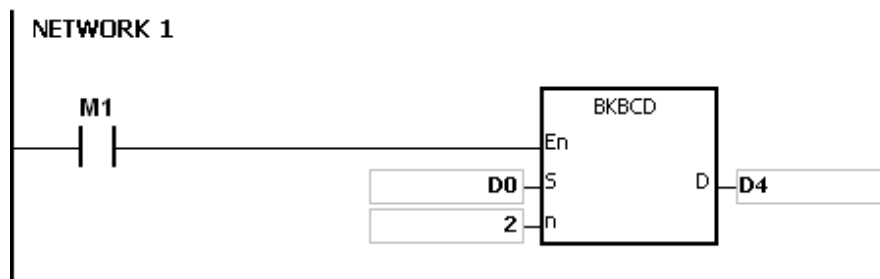


Explanation:

1. n pieces of data (the binary values) in devices starting from S are converted into the binary-coded decimal values, and the conversion results are stored in D.
2. The operand n should be within the range between 1 and 256.

Example:

When M1 is ON, the binary values in D0 and D1 are converted into the binary-coded decimal values, and the conversion results are stored in D4 and D5.



Additional remark:

1. If n is less than 1, or when n is larger than 256, the instruction is not execute, 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, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the conversion result is not within the range between 0 and 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 within the range between 0 and 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.

FB/FC	Instruction		Operand				Description	
FC		BKBIN	P	S, n, D				Converting the binary values in blocks into the binary-coded decimal numbers in blocks

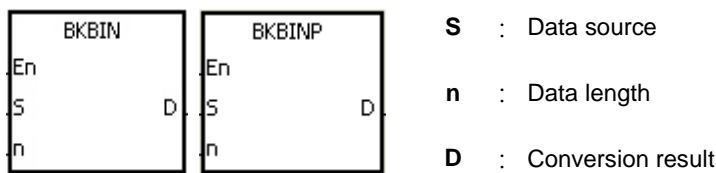
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●					●							
n		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:

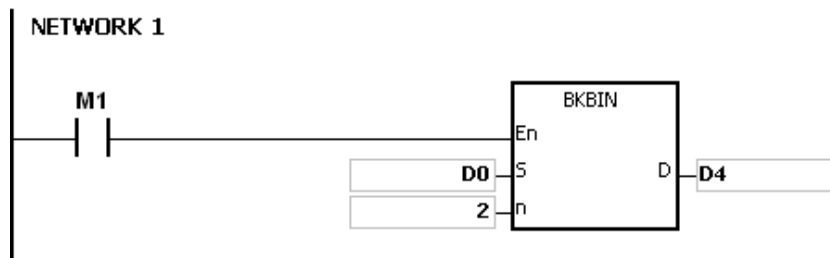


Explanation:

1. n pieces of data (the binary-coded decimal values) in devices starting from S are converted into the binary values, and the conversion results are stored in D.
2. The binary-coded decimal value in S should be within the range between 0 and 9,999.
3. The operand n should be within the rang between 1 and 256.

Example:

When M1 is ON, the binary-code decimal values in D0 and D1 are converted into the binary values, and the conversion results are stored in D4 and D5.



Additional remark:

1. If n is less than 1, or when n is larger than 256, the instruction is not execute, 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, 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 the 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 within the range between 0 and 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.

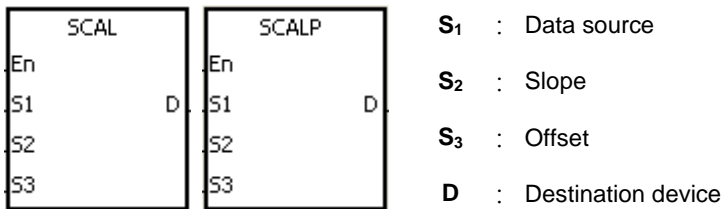
FB/FC	Instruction		Operand				Description					
FC		SCAL	P	S₁, S₂, S₃, D				Scale value operation				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂ , S ₃		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂ , S ₃	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



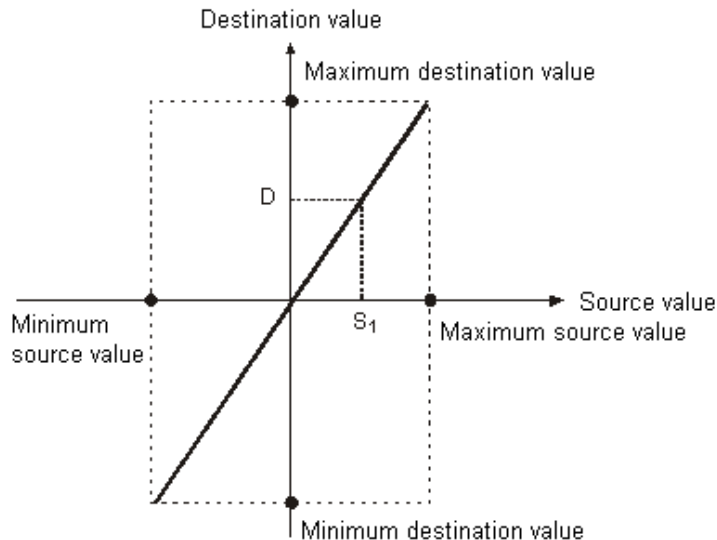
Explanation:

- The operation equation in the instruction: $D=(S1 \times S2) \div 1,000 + S3$
- To obtain the values in S2 and S3, you have to use the slope equation and the offset equation below first, and then round off the results to the nearest whole digit. The final 16-bit values are entered into S2 and S3.

The slope equation: $S2 = [(Maximum\ destination\ value - Minimum\ destination\ value) \div (Maximum\ source\ value - Minimum\ source\ value)] \times 1,000$

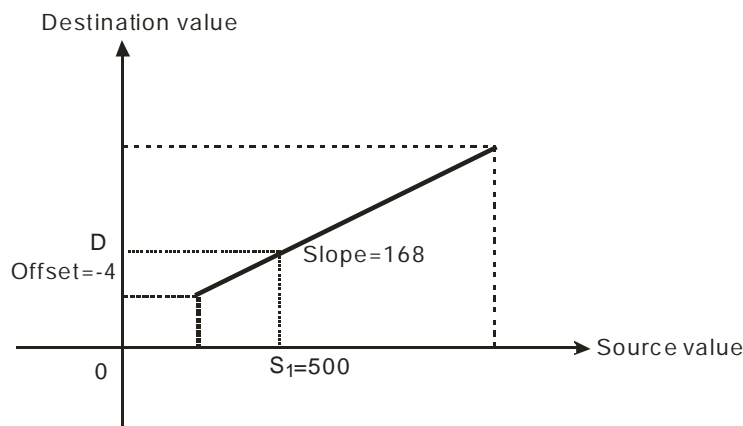
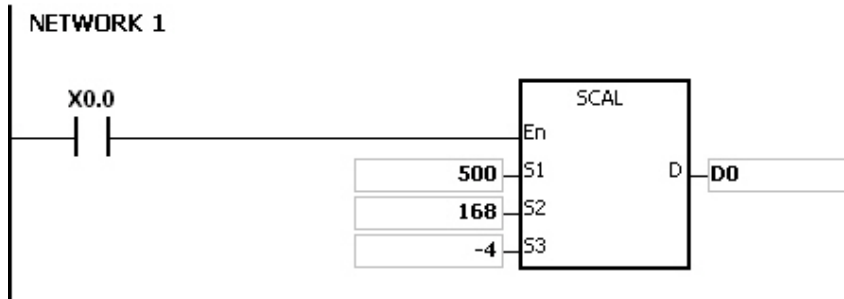
The offset equation: $S3 = Minimum\ destination\ value - Minimum\ source\ value \times S2 \div 1,000$

The output curve is as shown below:



Example 1:

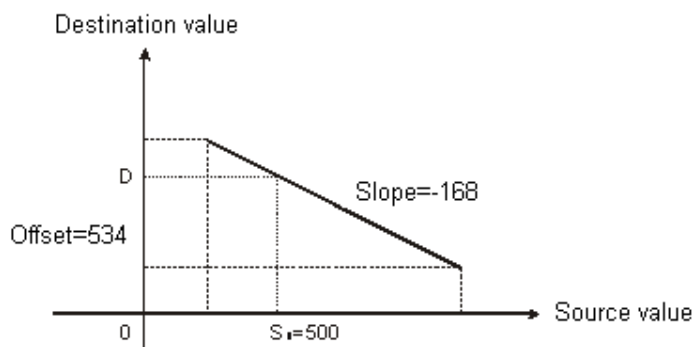
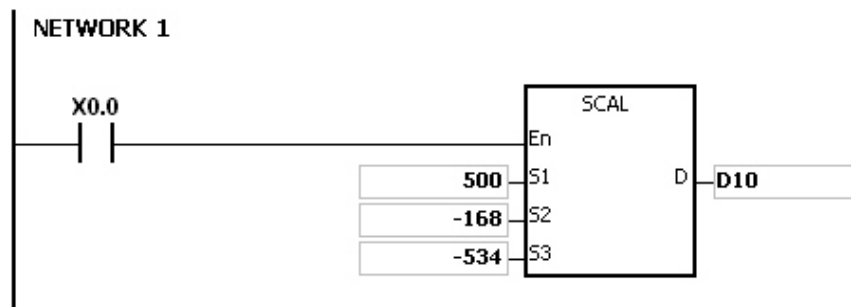
1. Suppose the values in S1, S2, and S3 are 500, 168, and -4 respectively. When X0.0 is ON, the instruction SCAL is executed, and the scale value is stored in D0.
2. The operation equation: $D0=(500 \times 168) \div 1,000 + (-4) = 80$



3

Example 2:

1. Suppose the values in S1, S2, and S3 are 500, -168, and 534 respectively. When X0.0 is ON, the instruction SCAL is executed, and the scale value is stored in D10.
2. The operation equation: $D10=(500 \times -168) \div 1,000 + 534 = 450$



Additional remark:

1. Only when the slope and the offset are known can the instruction SCAL be used. If the slope and the offset are unknown, you are suggested to use the instruction SCLP to perform the operation.
2. The value entered into S2 should be within the range between $-32,768$ and $32,767$. (The practical value is within the range between $-32,768$ and $32,767$.)
3. When you use the slope equation, they have to notice 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.
4. If the value in D is larger than $32,767$, the value stored in D will be $32,767$. If the value in D is less than $-32,768$, the value stored in D will be $-32,768$.

FB/FC	Instruction			Operand	Description
FC	D*	SCLP	P	S ₁ , S ₂ , D	Parameter type of scale value operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3 Graphic expression:



Explanation:

- Only the 32-bit instructions can use the 32-bit counter.
- The operand S₂ used in the 16-bit instruction is set as follows.

Device number	Parameter	Setting range
S ₂	Maximum source value	-32,768~32,767
S ₂ +1	Minimum source value	-32,768~32,767
S ₂ +2	Maximum destination value	-32,768~32,767
S ₂ +3	Minimum destination value	-32,768~32,767

- The operand S₂ used in the 16-bit instruction occupies four devices.
- The operand S₂ used in the 32-bit instruction is set as follows.

Device number	Parameter	Setting range	
		Integer	Floating-point value
S ₂ , S ₂ +1	Maximum source value	-2,147,483,648~ 2,147,483,647	The range of 32-bit floating-point values
S ₂ +2, S ₂ +3	Minimum source value		
S ₂ +4, S ₂ +5	Maximum destination value		
S ₂ +6, S ₂ +7	Minimum destination value		

- The operand S₂ used in the 32-bit instruction occupies eight devices.

6. If the values used in the 32-bit instruction are floating-point values, SM658 can be set to ON. If the values are decimal integers, SM685 can be set to OFF.
7. The operation equation in the instruction: $D = [(S_1 - \text{Minimum source value}) \times (\text{Maximum destination value} - \text{Minimum destination value})] \div (\text{Maximum source value} - \text{Minimum source value}) + \text{Minimum destination value}$
8. The operational relation between the source value and the destination value:

$$y = kx + b$$

$$y = \text{Destination value (D)}$$

$$k = \text{Slope} = (\text{Maximum destination value} - \text{Minimum destination value}) \div (\text{Maximum source value} - \text{Minimum source value})$$

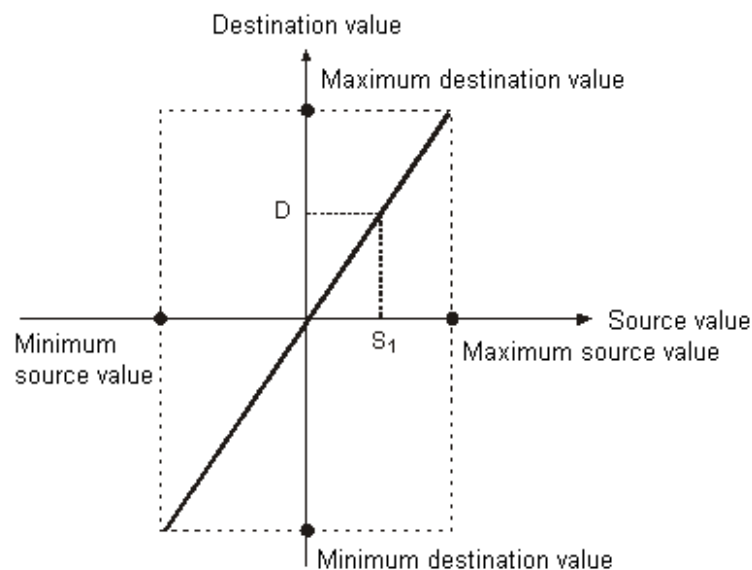
$$x = \text{Source value (S}_1\text{)}$$

$$b = \text{Offset} = \text{Minimum destination value} - \text{Minimum source value} \times \text{Slope}$$

The parameters above are being substituted for y, k, x, and b in the equation $y = kx + b$, and the operation equation in the instruction is obtained.

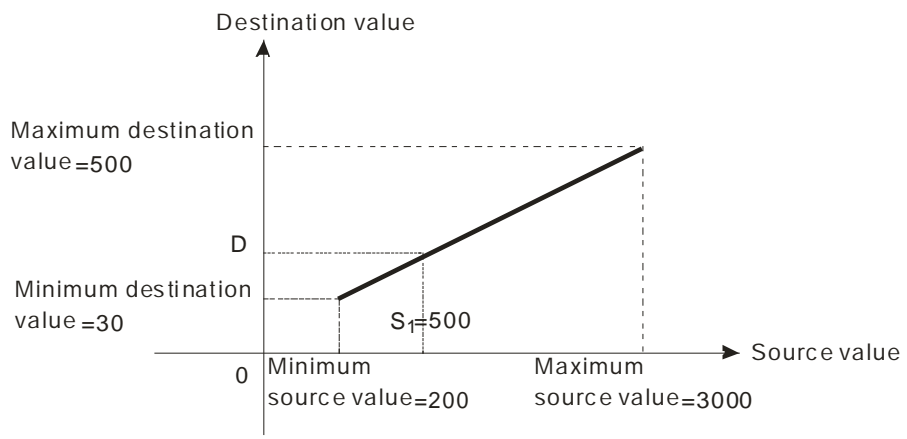
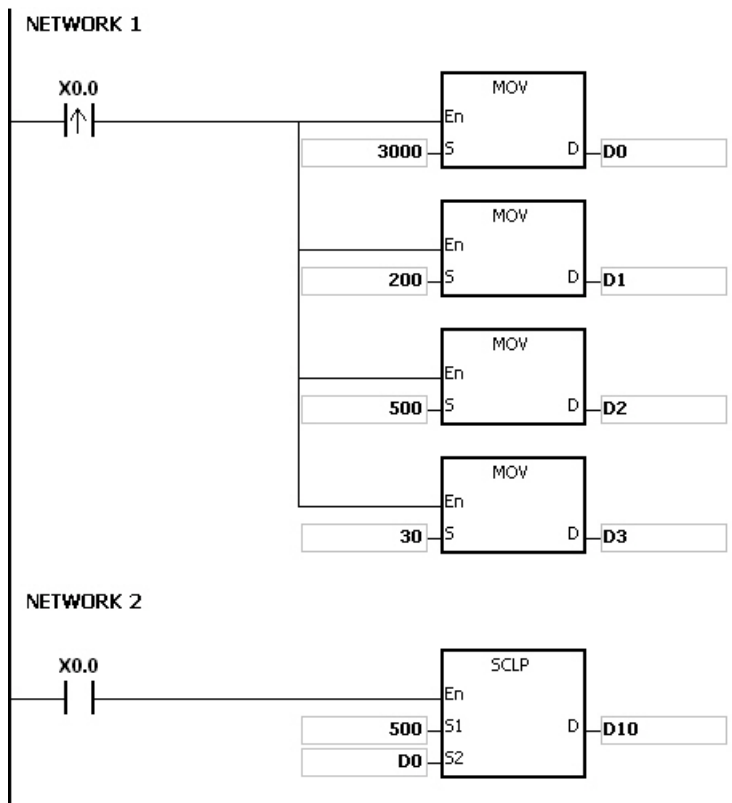
$$\begin{aligned} y = kx + b = D &= kS_1 + b = \text{Slope} \times S_1 + \text{Offset} = \text{Slope} \times S_1 + \text{Minimum destination value} - \text{Minimum source} \\ &\text{value} \times \text{Slope} = \text{Slope} \times (S_1 - \text{Minimum source value}) + \text{Minimum destination value} = (S_1 - \text{Minimum source} \\ &\text{value}) \times (\text{Maximum destination value} - \text{Minimum destination value}) \div (\text{Maximum source value} - \text{Minimum source} \\ &\text{value}) + \text{Minimum destination value} \end{aligned}$$

9. If S_1 is larger than the maximum source value, the maximum source value will be the value in S_1 . If S_1 is less than the minimum source value, the minimum source value will be the value in S_1 . After the input values and the parameters are set, the output curve is as shown below.



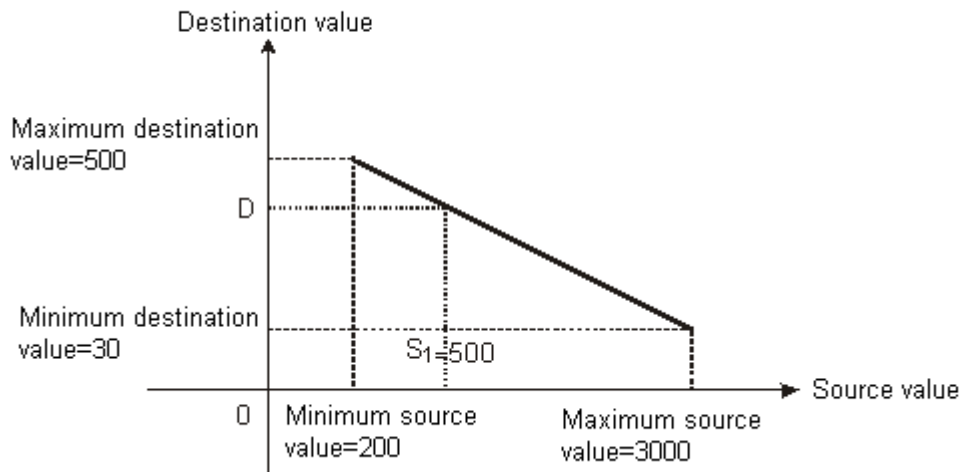
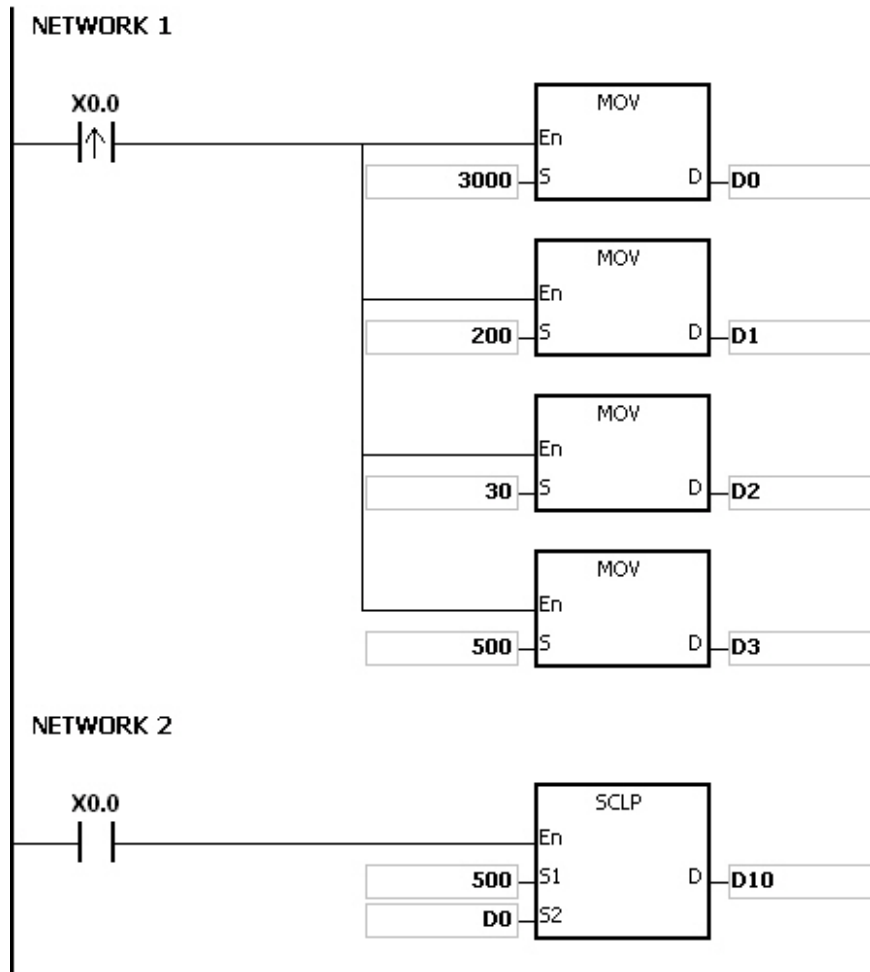
Example 1:

1. Suppose the value in S_1 is 500, the maximum source value in D_0 is 3,000, the minimum source value in D_1 is 200, the maximum destination value in D_2 is 500, and the minimum destination value in D_3 is 30. When X0.0 is ON, the instruction SCLP is executed, and the scale value is stored in D_{10} .
2. The operation equation: $D_{10} = [(500 - 200) \times (500 - 30)] \div (3,000 - 200) + 30 = 80.35$
3. 80.35 is rounded off to the nearest whole digit, and becomes 80. 80 is stored in D_{10} .



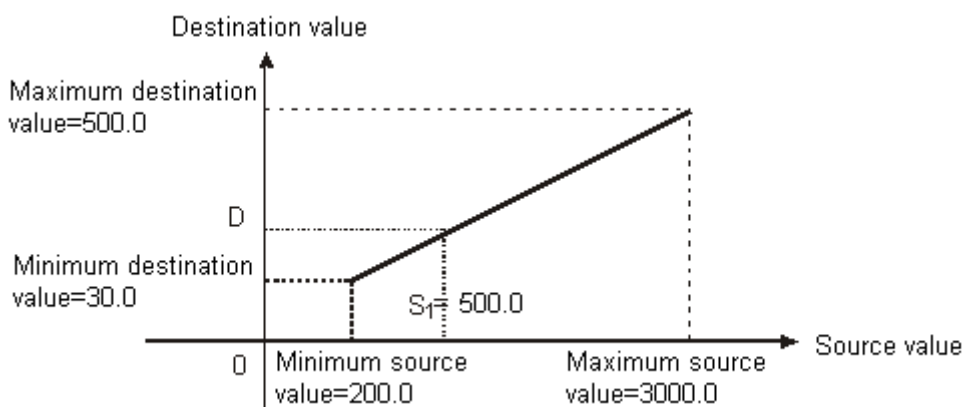
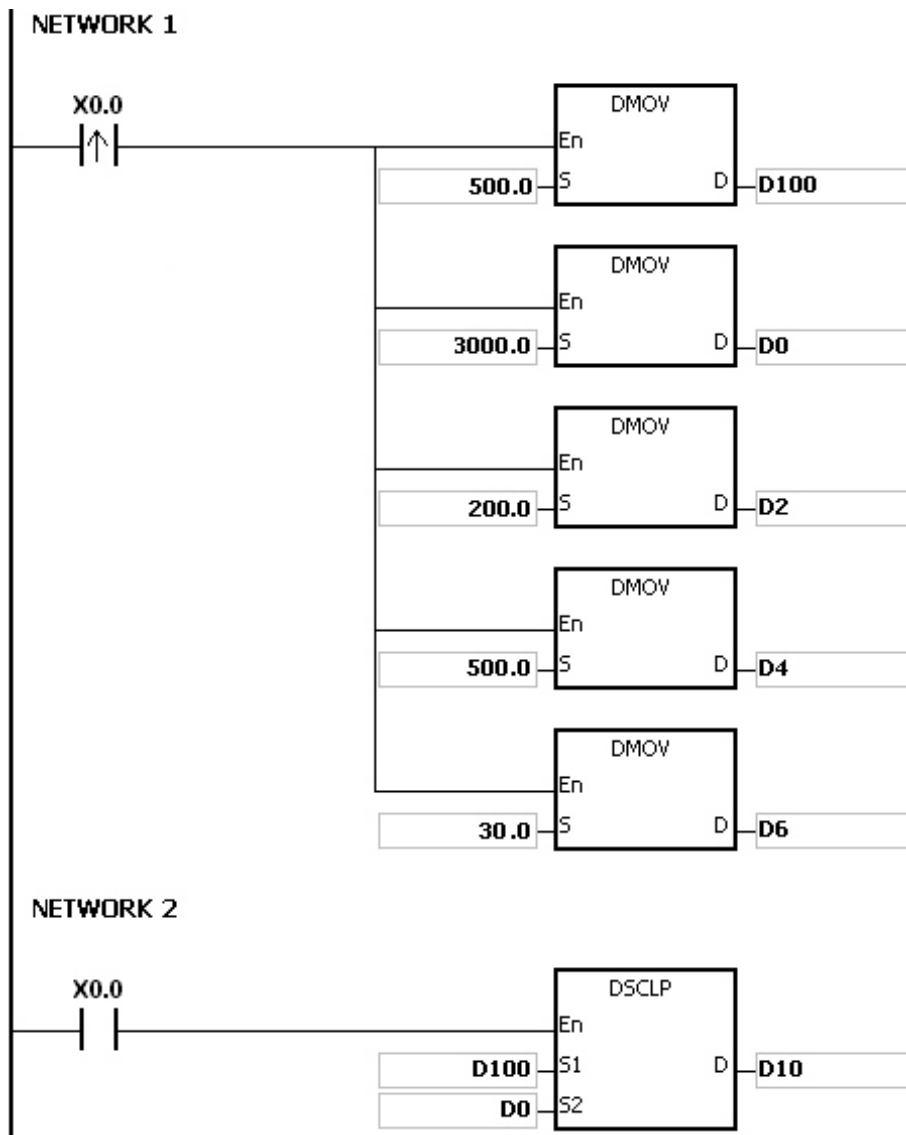
Example 2:

1. Suppose the value in S1 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 X0.0 is ON, the instruction SCLP is executed, and the scale value is stored in D10.
2. The operation equation: $D10 = [(500 - 200) \times (30 - 500)] \div (3,000 - 200) + 500 = 449.64$
3. 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 S1 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 X0.0 is ON, SM685 is set to ON, the instruction DSCLP is executed, and the scale value is stored in D10.
2. The operation equation: $D10 = [(500.0 - 200.0) \times (500.0 - 30.0)] \div (3000.0 - 200.0) + 30.0 = 80.35$
3. 80.35 is rounded off to the nearest whole digit, and becomes 80.0. 80.0 is stored in D10.

3



Additional remark:

1. The value in S2 which is used in the 16-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. between -32,768 and 32,767. If the value exceeds the boundary value, the boundary value is used in the operation.
2. The integer in S1 which is used in the 32-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. between -2,147,483,648 and 2,147,483,647. If the integer exceeds

the boundary value, the boundary value is used in the operation.

3. The floating-point value in S1 which is used in the 32-bit instruction should be within the range between the minimum source value and the maximum source value, i.e. within the range of floating-point values. If the floating-point value exceeds the boundary value, the boundary value is used in the operation.
4. When you use the instruction, you have to notice 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. If the maximum source value is set to the same value of the minimum source value, an error will occur and the instruction will not be executed. SM0 will be ON, and error code 16#2012 will be recorded in SR0.
6. If the operand S2 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [4] of WORD.
7. If the operand S2 used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [4] of DWORD.

FB/FC	Instruction			Operand	Description
FC	D*	LINE	P	S, n, D	Converting a column of data into a line of data

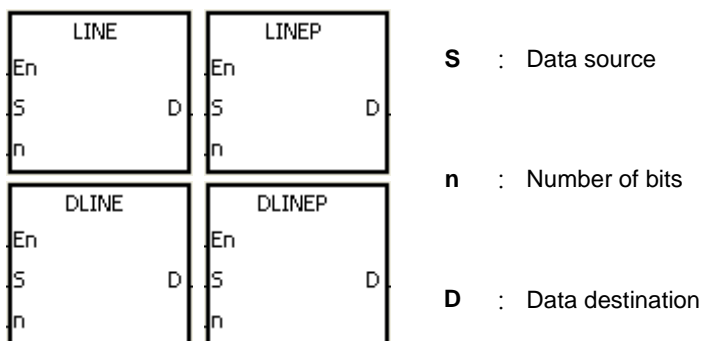
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
n		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●		●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

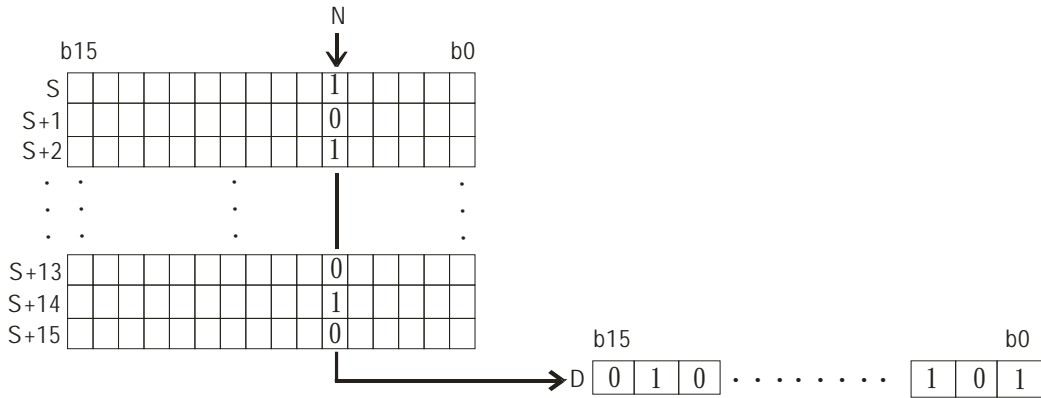
3

Graphic expression:

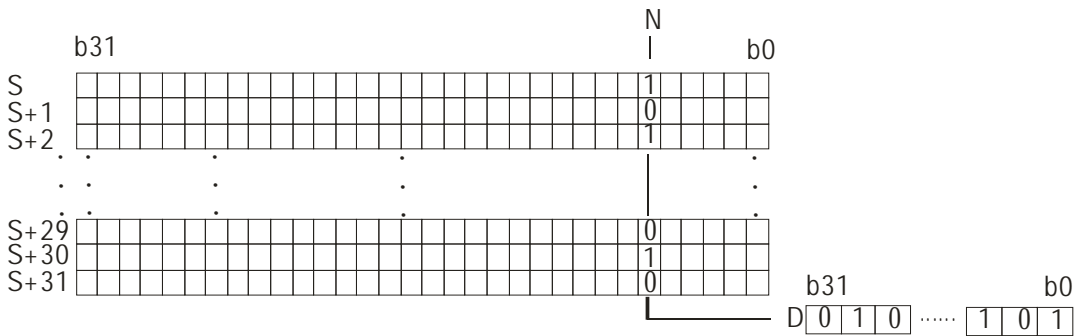


Explanation:

1. The operand S used in the 16-bit instruction occupies sixteen 16-bit registers, i.e. S~S+15.
2. The operand S used in the 32-bit instruction occupies thirty-two 32-bit registers, i.e. S~S+31.
3. The operand n indicates that the value of the nth bit in every piece of data in the operand S is retrieved. Besides, the operand n used in the 16-bit instruction should be within the range between 0 and 15, and the operand n used in the 32-bit instruction should be within the range between 0 and 31.
4. The operand n used in the 16-bit instruction indicates that the values of the nth bits in S~S+15 are retrieved, and the values of the nth bits are stored in the operand D in order.
5. The operand n used in the 32-bit instruction indicates that the values of the nth bits in S~S+31 are retrieved, and the values of the nth bits are stored in the operand D in order.
6. Only the 32-bit instructions can use the 32-bit counter.
7. Take the 16-bit instruction for example.

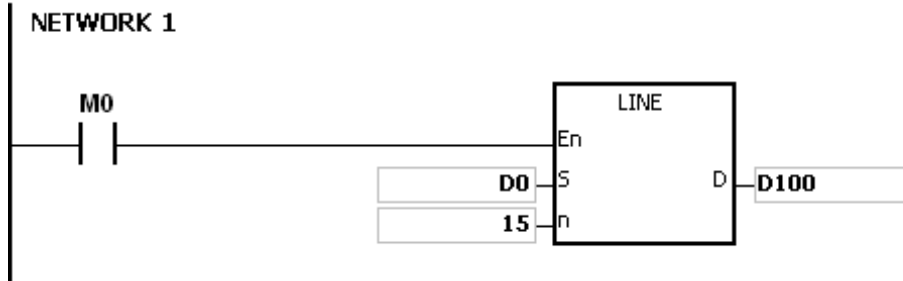


8. Take the 32-bit instruction for example.



Example:

When M0 is ON, the values of the 15th bits in D0~D14 are stored in b0~b15 in D100.



Additional remark:

1. If the device S+15 used in the 16-bit instruction is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the device S+31 used in the 32-bit instruction is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If n is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

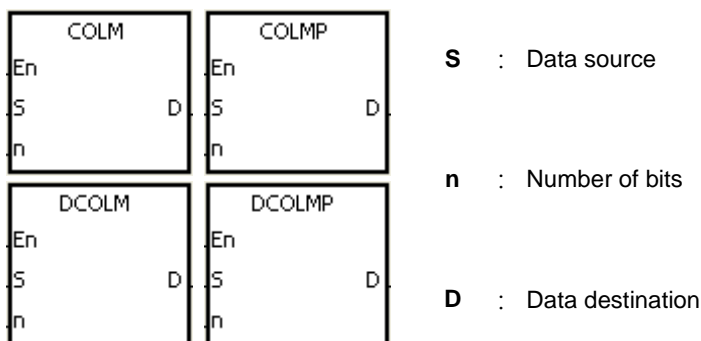
FB/FC	Instruction			Operand	Description
FC	D*	COLM	P	S, n, D	Converting a line of data into a column of data

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●		●				

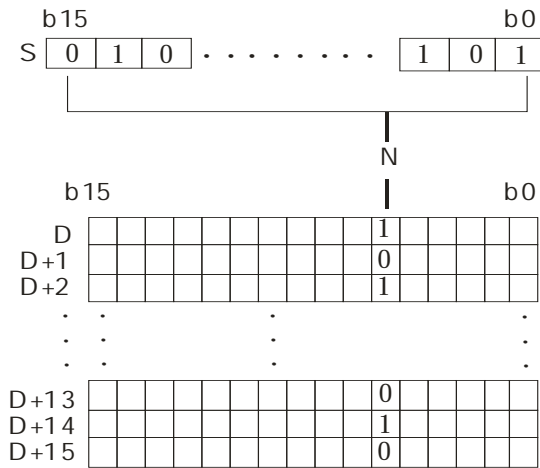
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

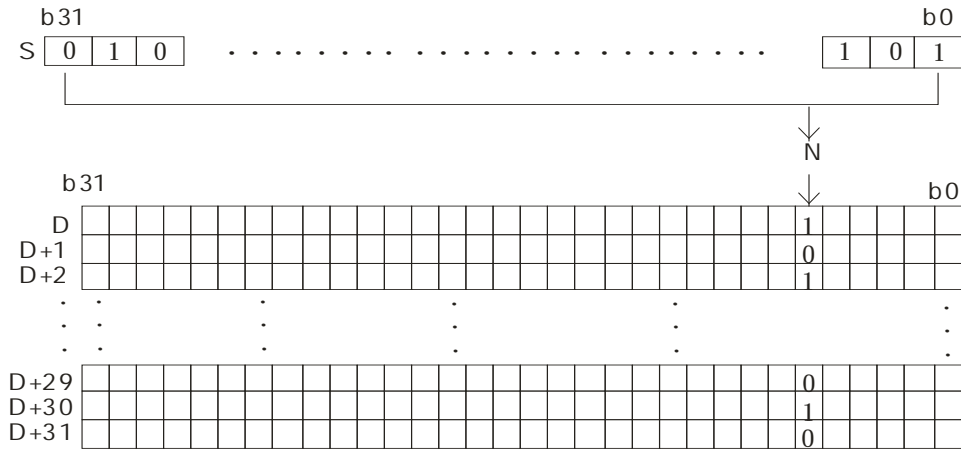


Explanation:

1. The operand D used in the 16-bit instruction occupies sixteen 16-bit registers, i.e. D~D+15.
2. The operand D used in the 32-bit instruction occupies thirty-two 32-bit registers, i.e. D~D+31.
3. The operand n indicates that the values of the bits in the operand S are stored in the nth bits in the operand D. Besides, the operand n used in the 16-bit instruction should be within the range between 0 and 15, and the operand n used in the 32-bit instruction should be within the range between 0 and 31.
4. The operand n used in the 16-bit instruction indicates that the values of the bits in S are stored in the nth bits in D~D+15 in order.
5. The operand n used in the 32-bit instruction indicates that the values of the bits in S are stored in the nth bits in D~D+31 in order.
6. Only the 32-bit instructions can use the 32-bit counter.
7. Take the 16-bit instruction for example.

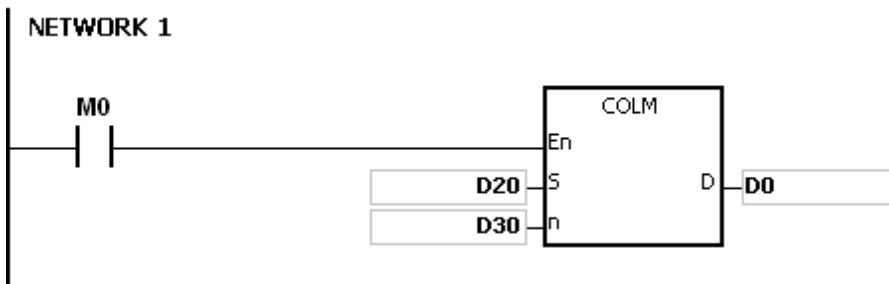


8. Take the 32-bit instruction for example.



Example:

Suppose the value in D30 is 3. When M0 is ON, the values of the bits in D20 are stored in the third bits in D0~D15 in order.



Additional remark:

1. If the device D+15 used in the 16-bit instruction is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the device D+31 used in the 32-bit instruction is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If n is out of the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3.6 Data Transfer Instructions

FB/FC	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>MOV</u>	<u>DMOV</u>	–	✓	Transferring the data	5
FC	–	–	<u>DFMOV</u>	✓	Transferring the 64-bit floating-point value	5-6
FC	<u>\$MOV</u>	–	–	✓	Transferring the string	5-11
FC	<u>CML</u>	<u>DCML</u>	–	✓	Inverting the data	5
FC	<u>BMOV</u>	–	–	✓	Transferring all data	7
FC	<u>NMOV</u>	<u>DNMOV</u>	–	✓	Transferring the data to several devices	7
FC	<u>XCH</u>	<u>DXCH</u>	–	✓	Exchanging the data	5
FC	<u>BXCH</u>	–	–	✓	Exchanging all data	7
FC	<u>SWAP</u>	<u>DSWAP</u>	–	✓	Exchange the high byte with the low byte	3
FC	<u>SMOV</u>	–	–	✓	Transferring the digits	11
FC	<u>MOVB</u>	–	–	✓	Transferring several bits	7

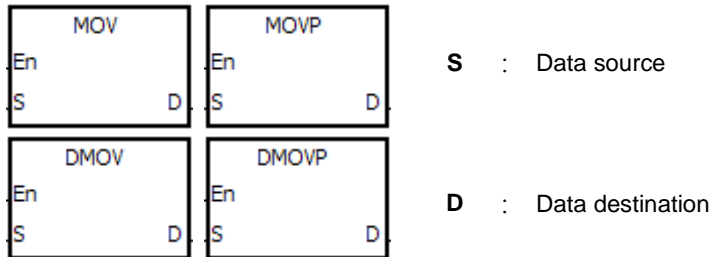
FB/FC	Instruction			Operand	Description
FC	D*	MOV	P	S, D	Transferring the data

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○		○	○		○
D	●	●			●	●	●	●	●		●	○					

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



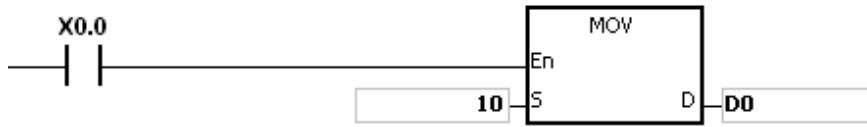
Explanation:

- When the instruction is executed, the data in S is transferred to D. When the instruction is not executed, the data in D is unchanged.
- Only the data in S which is used in the 32-bit instruction can be the floating-point value.
- Only the 32-bit instructions can use the 32-bit counter.

Example:

- To transfer the 16-bit data, you should use the instruction MOV.
 - When X0.0 is OFF, the data in D0 is unchanged. When X0.0 is ON, the value 10 is transferred to the data register D0.
 - When X0.1 is OFF, the data in D10 is unchanged. When X0.1 is ON, the current value of T0 is transferred to the data register D10.
- To transfer the 32-bit data, you should use the instruction DMOV.
 - When X0.0 is OFF, the data in (D31, D30) and (D41, D40) is unchanged. When X0.2 is ON, the current value in (D21, D20) is transferred to (D31, D30), and the current value of HC0 is transferred to (D41, D40).
- To transfer the floating-point value, you should use the instruction DMOV.
 - When X0.3 is OFF, the data in (D51, D50) is unchanged. When X0.3 is ON, the floating-point value 3.450 is converted into the binary floating-point value, and the conversion result is transferred to (D51, D50).

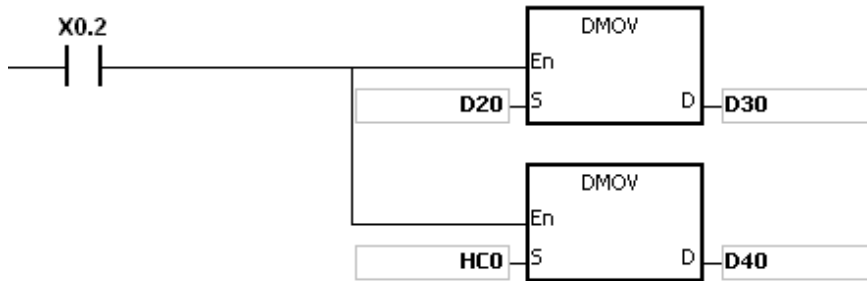
NETWORK 1



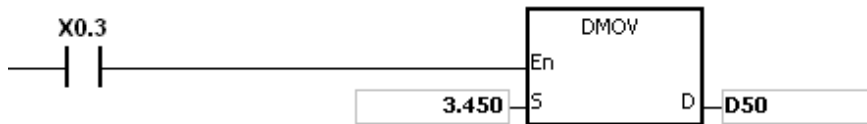
NETWORK 2



NETWORK 3



NETWORK 4



3

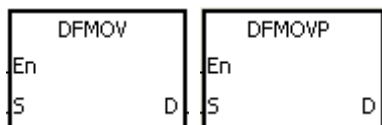
FB/FC	Instruction			Operand	Description
FC	D*	FMOV	P	S, D	Transferring the 64-bit floating-point value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●			●	○					○
D	●	●			●	●	●	●			●	○					

Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	-	AH Motion CPU

Graphic expression:



S : Data source

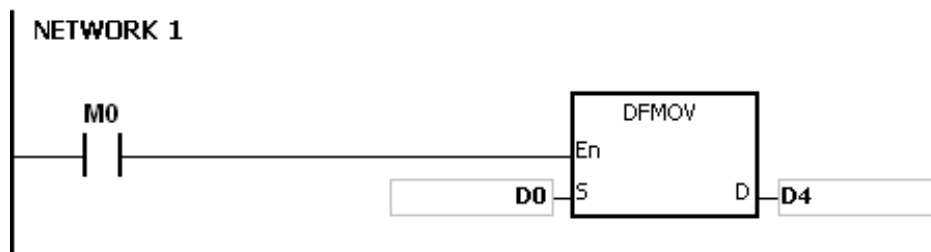
D : Data destination

Explanation:

1. When the instruction is executed, the data in S is transferred to D. When the instruction is not executed, the data in D is unchanged.
2. Only the 64-bit instructions are supported.
3. The instructions DFMOV and DFMOV P are double-precision data transfer instructions.

Example:

When M0 is ON, the values in D0~D3 are transferred to D4~D7.



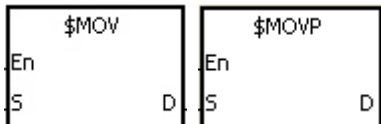
FB/FC	Instruction			Operand	Description
FC	\$MOV	P		S, D	Transferring the string

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●	○	●			○	
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S : Data source
D : Data destination

Explanation:

1. If the operand S is a string, at most 31 characters can be moved. For a string, the number of steps=1+(the number of characters +1)/4 (The value will be rounded up to the nearest whole digit if (the number of characters +1) is not divisible by 4.).

Number of characters	1~3	4~7	8~11	12~15	16~19	20~23	24~27	28~31
Number of steps	2	3	4	5	6	7	8	9

2. When the operand S is a string and the instruction is executed, the string is transferred to D, and the code 16#00 is added to the end of the data
3. When the operand S is not a string, the code 16#00 should be added to the end of the data transferred.
4. When the 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. Suppose the operand S is not a string. When the instruction is executed and the first character is the code 16#00, 16#00 is still transferred to D.
6. 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			B15~b8 b7~b0		
S	16#31	16#30	D	16#38	16#39
S+1	16#33	16#32	D+1	16#36	16#37
S+2	16#35	16#34	D+2	16#34	16#35
S+3	16#30	16#00	D+3	16#32	16#33

After the instruction is executed:

b15~b8 b7~b0			b15~b8 b7~b0		
S	16#31	16#30	D	16#31	16#30
S+1	16#33	16#32	D+1	16#33	16#32
S+2	16#35	16#34	D+2	16#35	16#34
S+3	16#30	16#00	D+3	16#00	16#00

↑ 16#30 in the high byte is not transferred.
 ↑ 16#32 in the high byte turns into 16#00.

7. When 16#00 appears in the high byte, the execution of the instruction is as follows.

Before the instruction is executed:

b15~b8 b7~b0			b15~b8 b7~b0		
S	16#31	16#30	D	16#38	16#39
S+1	16#33	16#32	D+1	16#36	16#37
S+2	16#00	16#34	D+2	16#34	16#35
S+3	16#37	16#36	D+3	16#32	16#33

After the instruction is executed:

b15~b8 b7~b0			b15~b8 b7~b0		
S	16#31	16#30	D	16#31	16#30
S+1	16#33	16#32	D+1	16#33	16#32
S+2	16#00	16#34	D+2	16#00	16#34
S+3	16#37	16#36	D+3	16#32	16#33

8. 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.

Before the instruction is executed:

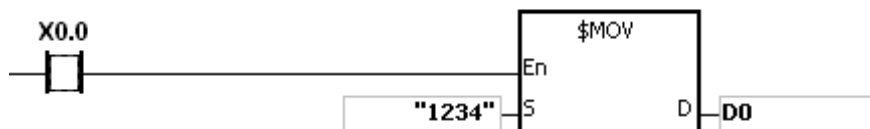
	b15~b8	b7~b0		b15~b8	b7~b0
D0	16#31	16#30	→	D1	16#33
D1	16#33	16#32		D2	16#35
D2	16#35	16#34		D3	16#30
D3	16#30	16#00		D4	16#38

After the instruction is executed:

	b15~b8	b7~b0		b15~b8	b7~b0
D0	16#31	16#30	→	D1	16#31
D1	16#33	16#32		D2	16#33
D2	16#35	16#34		D3	16#35
D3	16#30	16#00		D4	16#00

Example 1:

Suppose the operand S is the even string "1234". When the conditional contact X0.0 is enabled, the data in D0~D3 is as follows.



The operand S:

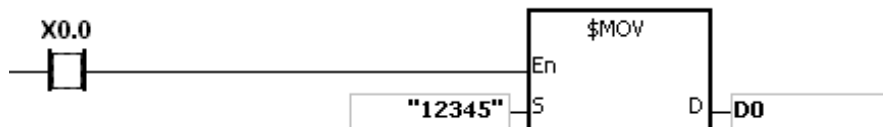
String	'1'	'2'	'3'	'4'
Hexadecimal value	16#31	16#32	16#33	16#34

After the instruction is executed, the data in 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#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 operand S is the odd string "12345". When the conditional contact X0.0 is enabled, the data in D0~D3 is 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 operand 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
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 operand 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

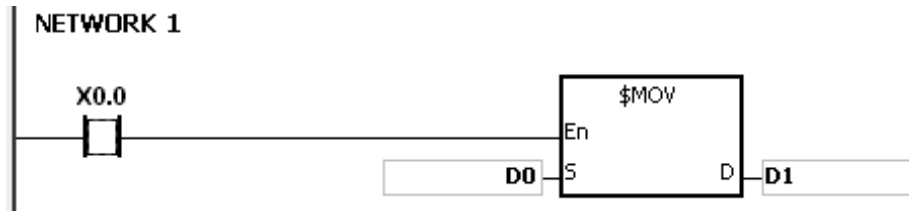
After the instruction is executed, the data in the operand D is as follows.

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 the operand 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	

Additional remark:

1. If the string in S does not end with 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the capacity of the device D is not sufficient to contain the string in S, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

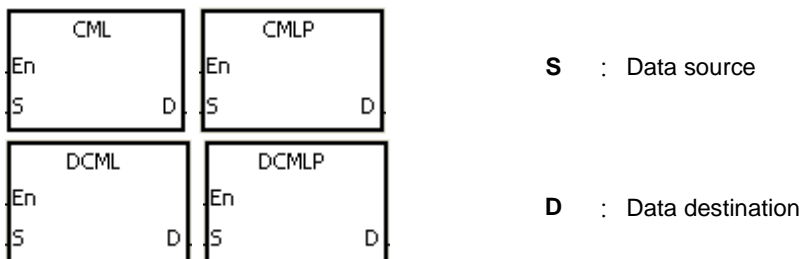
FB/FC	Instruction			Operand	Description
FC	D*	CML	P	S, D	Inverting the data

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

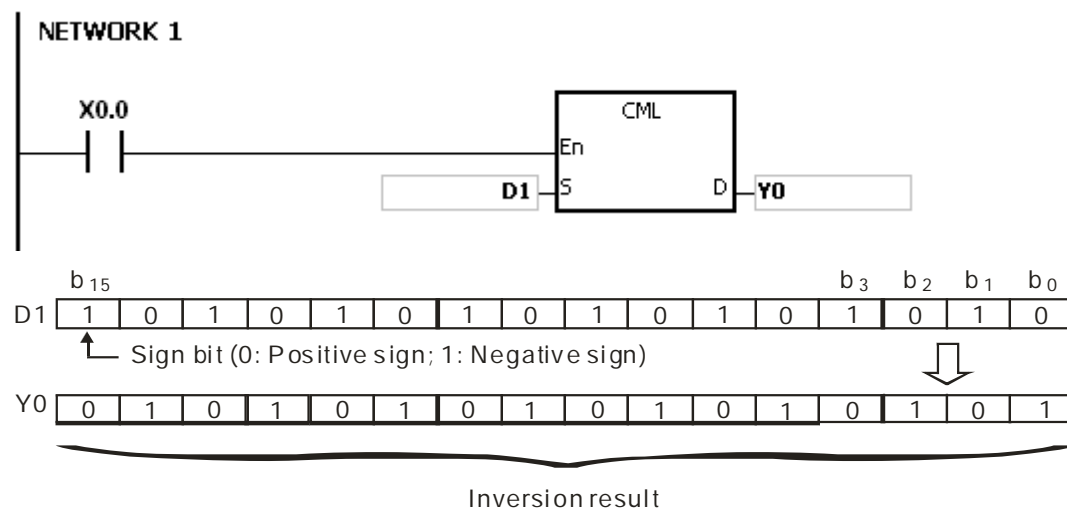


Explanation:

1. The instruction is used to invert all bits in S, i.e. 0 becomes 1, and 1 becomes 0. The inversion result is stored in D. If the data in S is the constant, the constant will be converted into the binary value.
2. Only the 32-bit instructions can use the 32-bit counter.

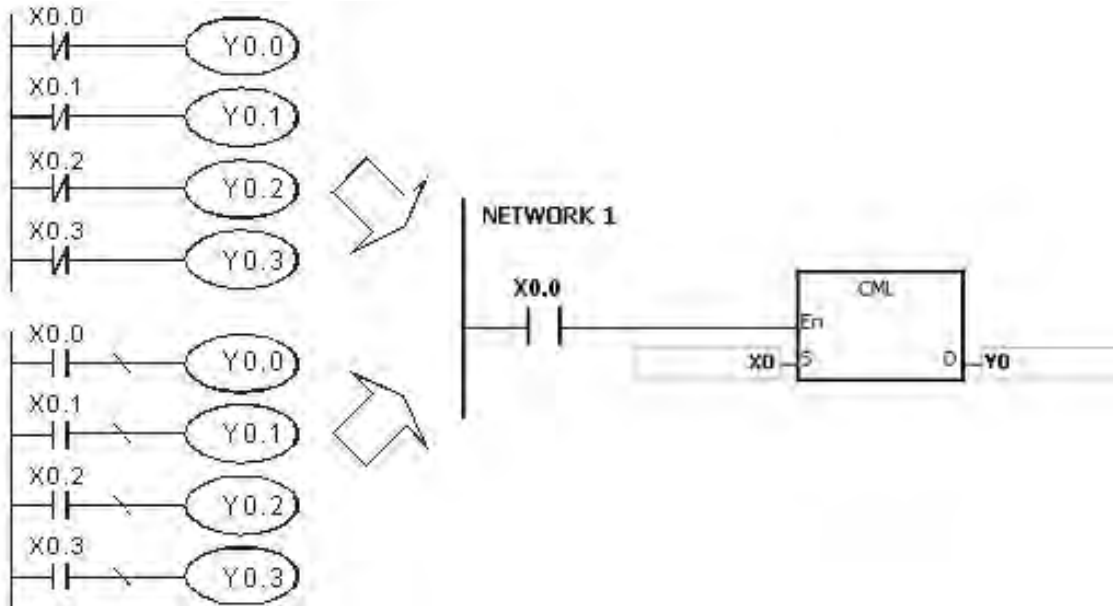
Example 1:

When X0.0 is ON, all bit in D1 are inverted, and the conversion result is stored in Y0.0~Y0.15.



Example 2:

The circuits below can be represented by means of the instruction CML.



3

FB/FC	Instruction		Operand	Description
FC	BMOV	P	S, D, n	Transferring all data

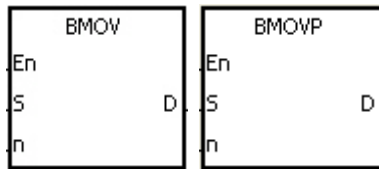
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●					●							
D		●					●							
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	

3

Graphic expression:

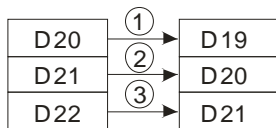


- S** : Data source
- D** : Data destination
- n** : Data length

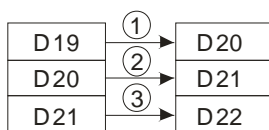
Explanation:

- n pieces of data in devices starting from the device specified by S are transferred to the devices starting from the device specified by D.
- The operand n should be within the range between 1 and 256.
- In order to prevent the error which results from the overlap between the source devices and the destination devices, the data is transferred in the following way.

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.

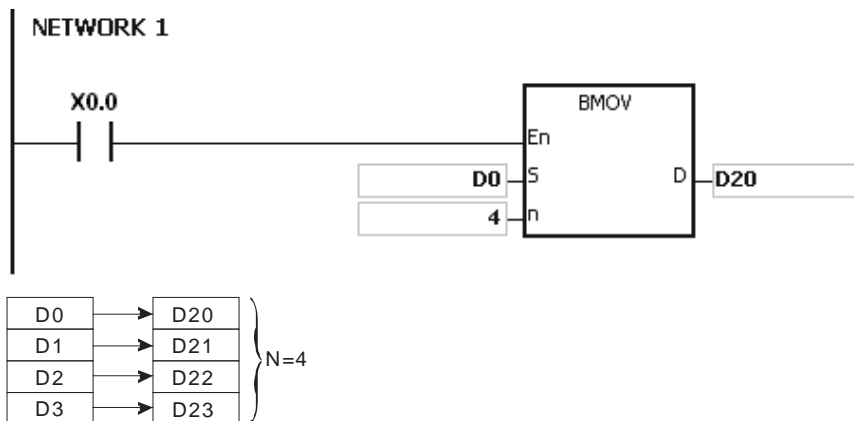


When the device number of S is less than the device number of D, the data is transferred in the order from ③ to ①.



Example 1:

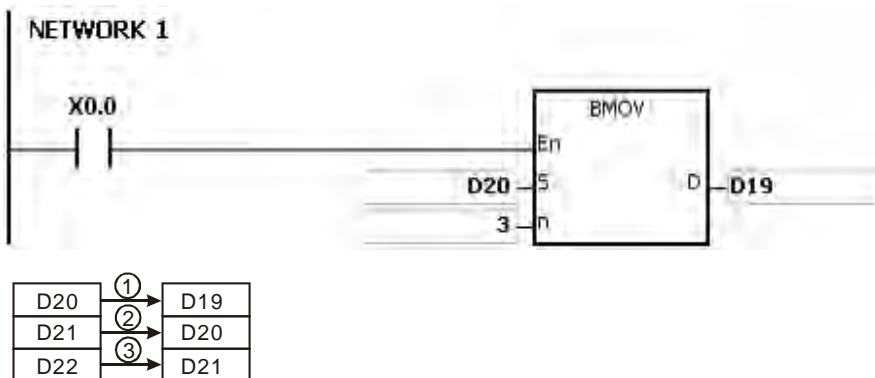
When X0.0 is ON, the data in D0~D3 is transferred to D20~D23.



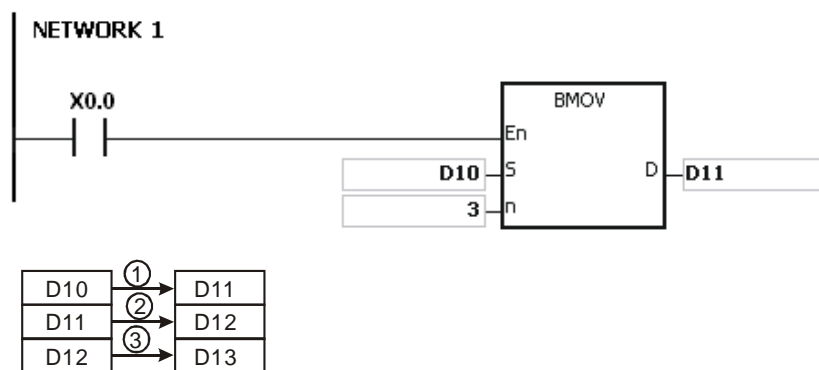
Example 2:

In order to prevent the error which results from the overlap between the source devices and the destination devices, the data is transferred in the following way.

- 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.



Additional remark:

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 larger than 256, or if n is less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	D*	NMOV	P	S, D, n	Transferring the data to several devices

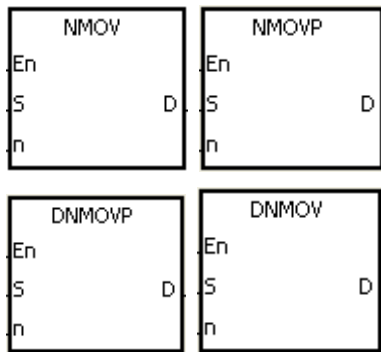
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D		●	●*				●	●*						
n		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:



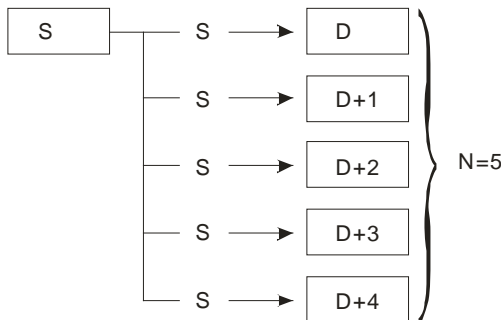
S : Data source

D : Data destination

n : Data length

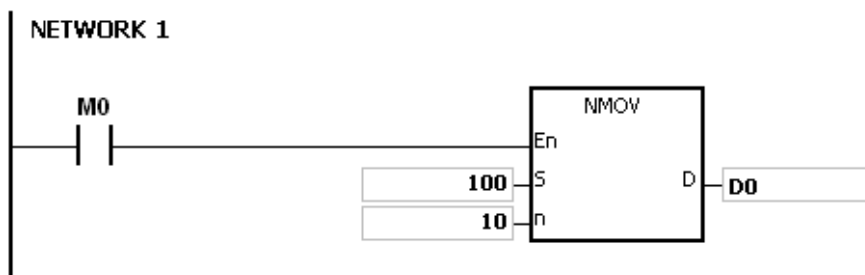
Explanation:

1. When the instruction is executed, the data in S is transferred 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 operand n used in the instruction NMOV should be within the range between 1 and 256, and the operand n used in the instruction DNMOV should be within the range between 1 and 128.



Example:

When M0 is ON, 100 is transferred to D0~D9.



Additional remark:

1. If $D \sim D+n-1$ exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the operand n used in the 16-bit instrucion is larger than 256 or less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If the operand n used in the 32-bit instrucion is larger than 128 or less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

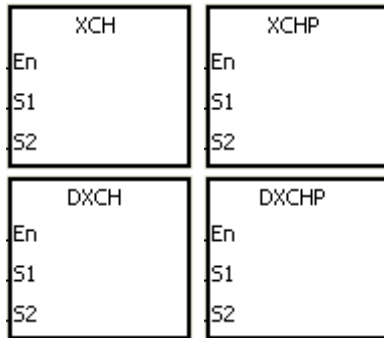
FB/FC	Instruction			Operand	Description
FC	D*	XCH	P	S ₁ , S ₂	Exchanging the data

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●	●*				●	●*						
S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●	●	●	●		●	○	●				
S ₂	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data which will be exchanged

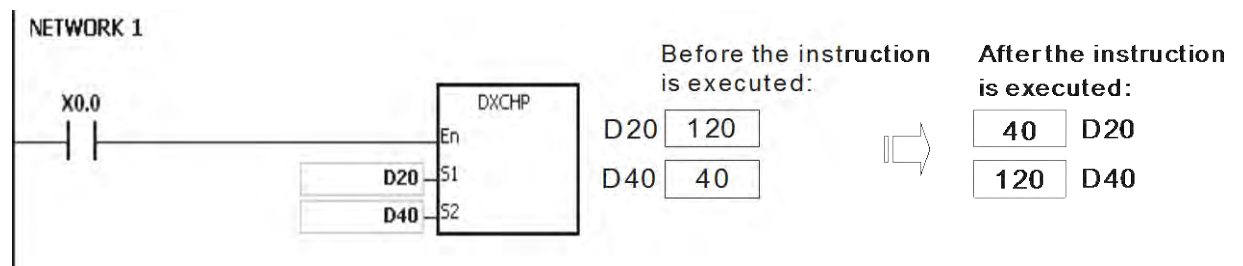
S₂ : Data which will be exchanged

Explanation:

1. The data in the device specified by S1 is exchanged with the data in the device specified by S2.
2. Only the 32-bit instructions can use the 32-bit counter.

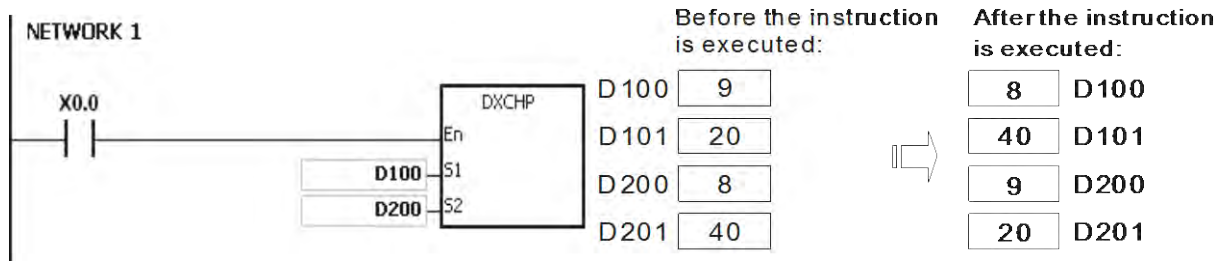
Example 1:

When X0.0 is switched from OFF to ON, the data in D20 is exchanged with the data in D40.



Example 2:

When X0.0 is switched from OFF to ON, the data in D100 is exchanged with the data in D200.



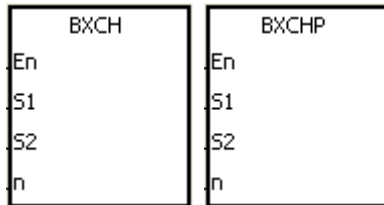
FB/FC	Instruction		Operand		Description
FC	BXCH	P	S₁, S₂, n		Exchanging all data

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



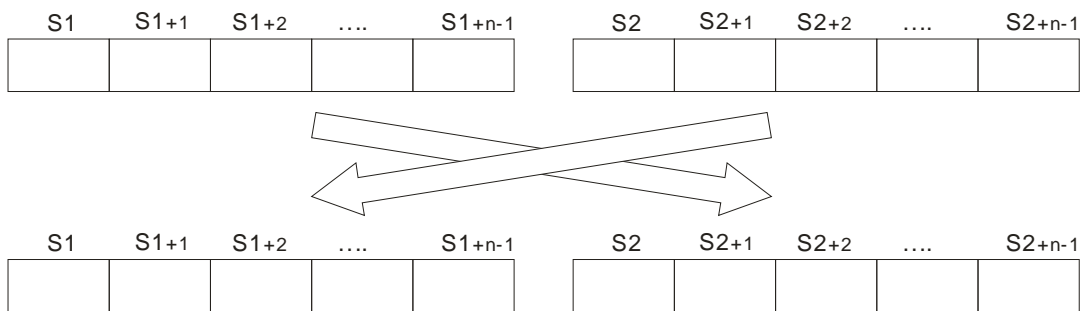
S₁ : Data which will be exchanged

S₂ : Data which will be exchanged

n : Data length

Explanation:

- The data in the devices specified by S₁~S₁+n-1 is exchanged with the data in the devices specified by S₂~S₂+n-1.
- The operand n used in the instruction should be within the range between 1 and 256.



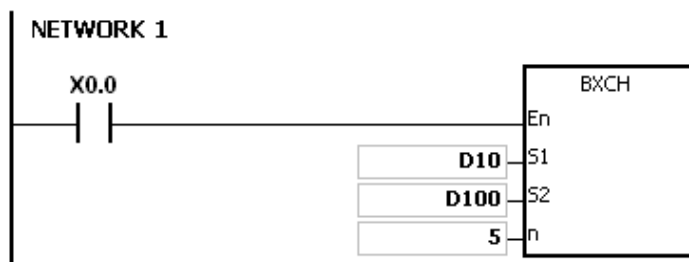
Example:

When X0.0 is ON, the data in D10~D14 is exchanged with the data in D100~D104.

D10	D11	D12	D13	D14	D100	D101	D102	D103	D104
1	2	3	4	5	16	17	18	19	20

After the instruction is executed

D10	D11	D12	D13	D14	D100	D101	D102	D103	D104
16	17	18	19	20	1	2	3	4	5



Additional remark:

1. If $S1+n-1$ exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If $S2+n-1$ exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the operand n used in the instruction is larger than 256 or less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	D*	SWAP	P	S	Exchange the high byte with the low byte

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●			○	

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



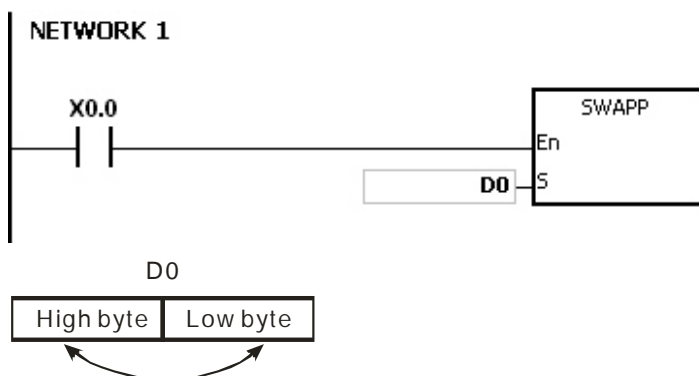
S : Data source

Explanation:

1. When the 16-bit instruction is executed, the data in the low byte in S is exchanged with the data in the high byte in S.
2. When the 32-bit instruction is executed, the data in the low byte of the high word in S is exchanged with the data in the high byte of the high word in S, and the data in the low byte of the low word in S is exchanged 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.

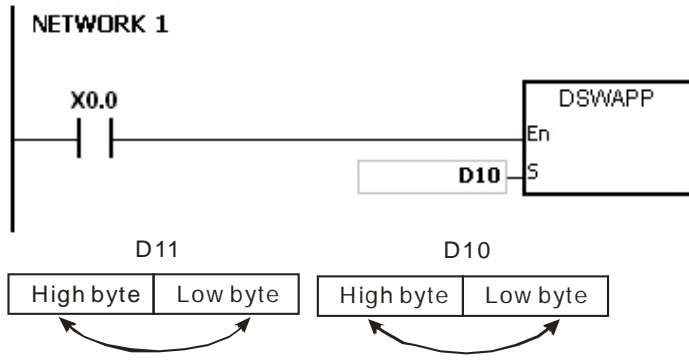
Example 1:

When X0.0 is ON, the data in the low byte in D0 is exchanged with the data in the high byte in D0.



Example 2:

When X0.0 is ON, the data in the low byte in D11 is exchanged with the data in the high byte in D11, and the data in the low byte in D10 is exchanged with the data in the high byte in D10.



3

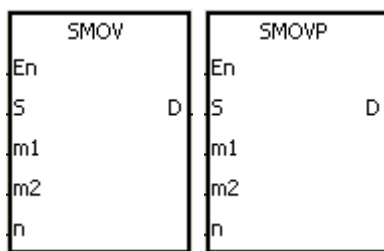
FB/FC	Instruction		Operand	Description
FC	SMOV	P	S, m₁, m₂, D, n	Transferring the digits

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●					●							
m₁, m₂		●					●							
D		●					●							
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
m₁, m₂	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

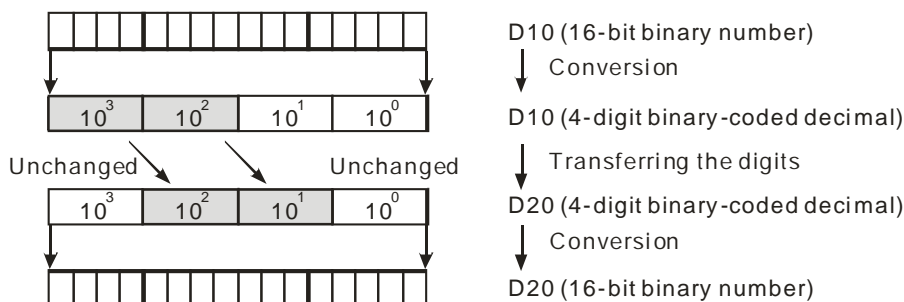
Graphic expression:



- S** : Data source
- m₁** : Start digit which will be transferred from the source device
- m₂** : Number of digits which will be transferred
- D** : Data destination
- n** : Start digit where the source data is stored in the destination device

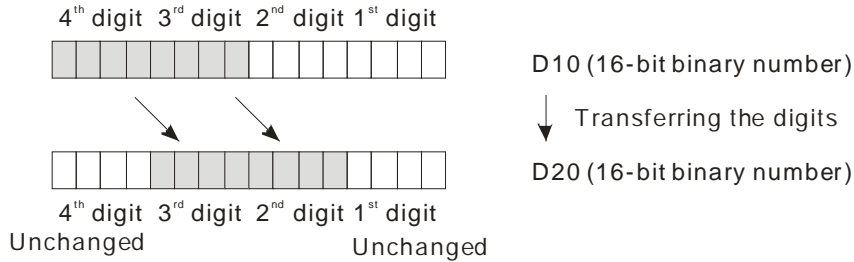
Explanation:

- The instruction can be used to allocate and combine the data. When the instruction is executed, the m₂ digits of the number which start from the m₁th digit of the number in S are transferred to the m₂ digits of the number which starts from the n_{th} digit of the number in D.
- The operand m₁ should be within the range between 1 and 4. The operand m₂ should be within the range between 1 and m₁. The operand n should be within the range between m₂ and 4. (Four bits are regarded as a unit.)
- When SM605 is OFF, the data involved in the instruction is 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.

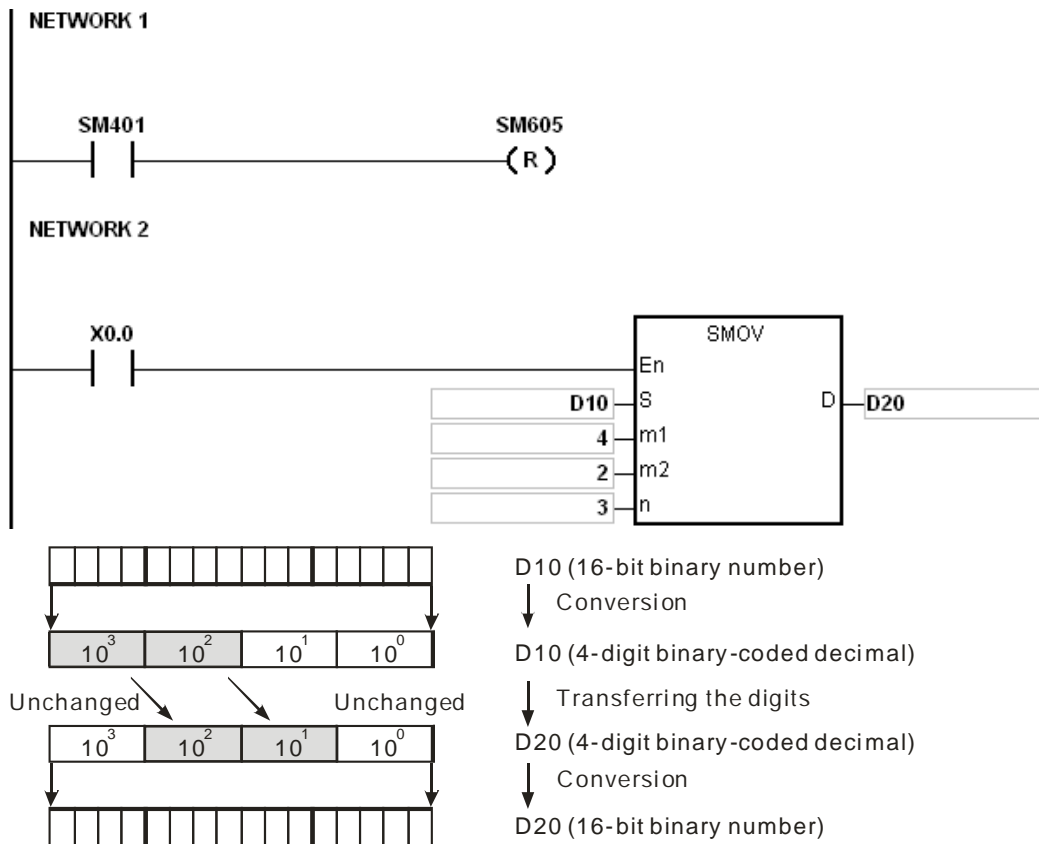
- When SM605 is ON, the data involved in the instruction is binary values.



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:

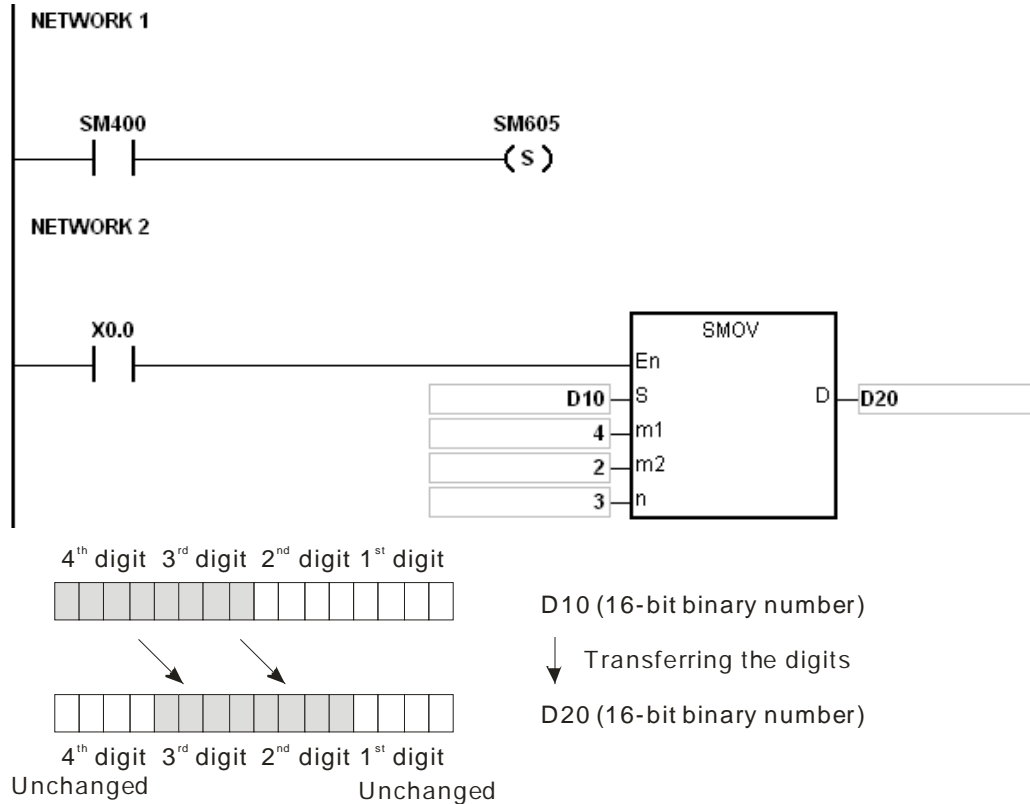
- When SM605 is OFF, the data involved in the instruction is binary-coded decimal numbers. When X0.0 is ON, the two digits of the decimal number which start from the fourth digit of the decimal number (the digit in the thousands place of the decimal number) in D10 are transferred to the two digits of the decimal number which start 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 (103) and the ones place of the decimal number (100) in D20 are unchanged.
- When the binary-code decimal number is out of the range between 0 and 9,999, the operation error occurs, 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.

Example 2:

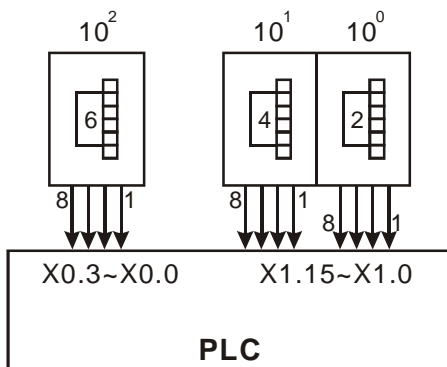
When SM605 is ON, the data involved in the instruction is binary values. When the instruction SMOV is executed, the binary values in D10 and D20 are not transformed into the binary-coded decimal numbers, and the digit which is transferred is composed of four bits.

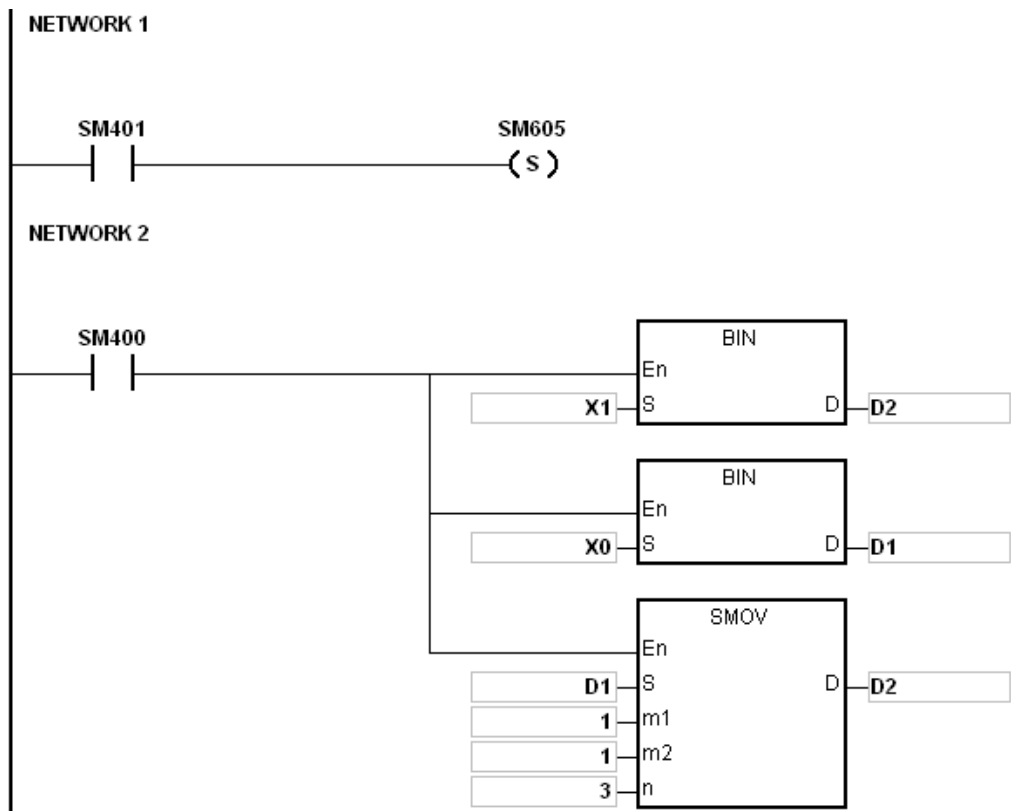


Suppose the number in D10 is 16#1234, and the number in D20 is 16#5678. After the instruction is executed, the number in D10 is unchanged, and the number in D20 is 16#5128.

Example 3:

1. The instruction can be used to combine the values of the DIP switches which 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 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 the first digit of the number in D1.
3. The instruction SMOV can be used 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 instruction SMOV.





Additional remark:

1. Suppose the data involved in the instruction is binary-coded decimal numbers. If the number in S is not within the range between 0 and 9999, or if the number in D is not within the range between 0 and 9999, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.
2. If m1 is less than 1, or if m1 is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If m2 is less than 1, or if m2 is larger than m1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. If n is less than m2, or if n is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand	Description
FC	MOVB	P	S, n, D	Transferring several bits

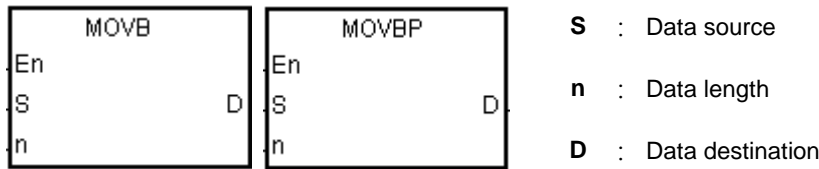
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●													
n		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●	●	●	●	●	●	●	●	●			●				
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●	●	●	●	●	●	●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:

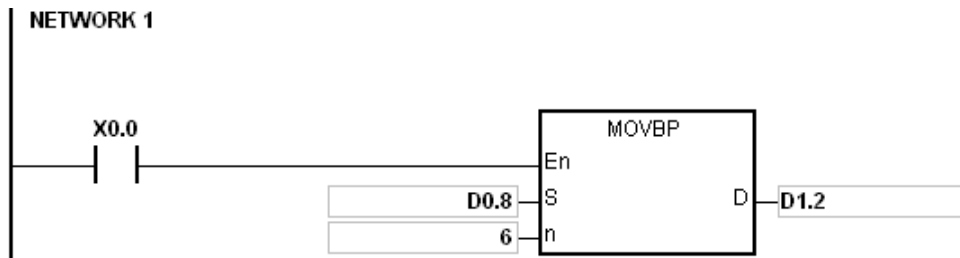


Explanation:

1. When the instruction is executed, n pieces of data in devices starting from the device specified by S are transferred to the devices starting from the device specified by D.
2. When S or D is T, C or HC/AC, only the state of the device is transferred, and the current value of the device is not transferred.
3. The operand n should be within the range between 1 and 256. When n is less than 1, or when n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

Example:

When X0.0 is ON, the data in D0.8~D0.13 is transferred to D1.2~D1.7.



Additional remark:

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.

3.7 Jump Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>CJ</u>	–	✓	Conditional jump	3
FC	<u>JMP</u>	–	–	Unconditional jump	3
FC	<u>GOEND</u>	–	–	Jumping to the end of the program	1

FB/FC	Instruction			Operand				Description					
FC		CJ	P	S				Conditional jump					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S																	

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

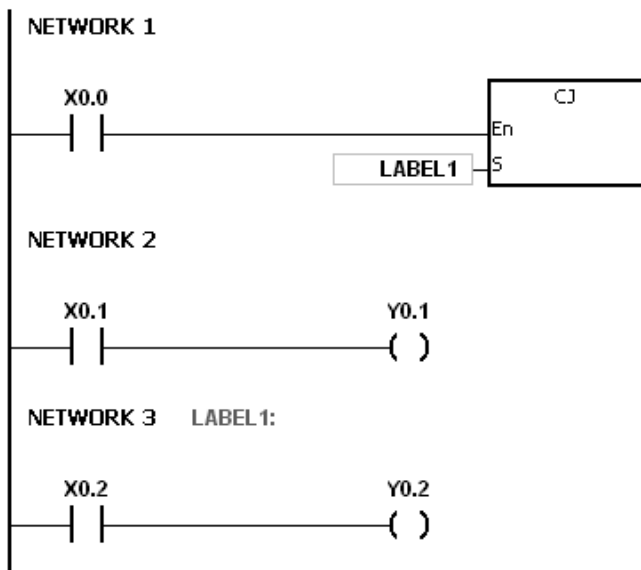


Explanation:

1. When some part of the program in the PLC does not need to be executed, you can use CJ or CJP to shorten the scan time. Besides, when a dual output is used, you also can use CJ or CJP.
2. If the program specified by the label is prior to the instruction CJ, the watchdog timer error will occur, and the PLC will stop running. Please use the instruction carefully.
3. The instruction CJ can specify the same label repeatedly.
4. When the instruction is executed, the actions of the devices are as follows.
 - The state of Y, the state of M, and the state of S remain the same as those before the execution of the jump.
 - The timer stops counting.
 - If the instruction which is used to reset the timer is driven before the jump is executed, the timer will still be in the condition of being reset during the execution of the jump.
 - The general applied instructions are not executed.

Example 1:

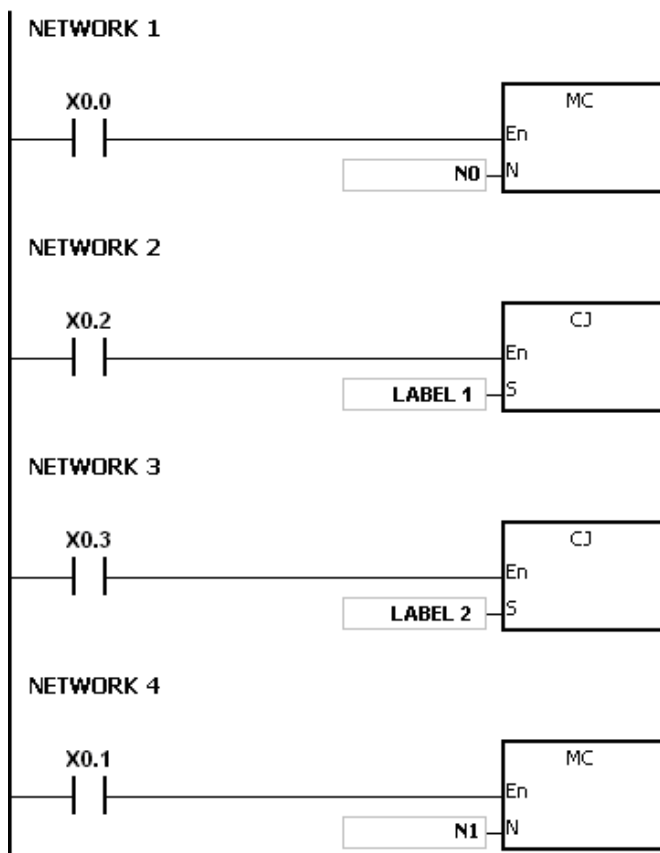
1. When X0.0 is ON, the execution of the program jumps from address 0 to address N (LABEL1:).
2. When X0.0 is OFF, the execution of the program starts from address 0, and the instruction CJ is not executed.

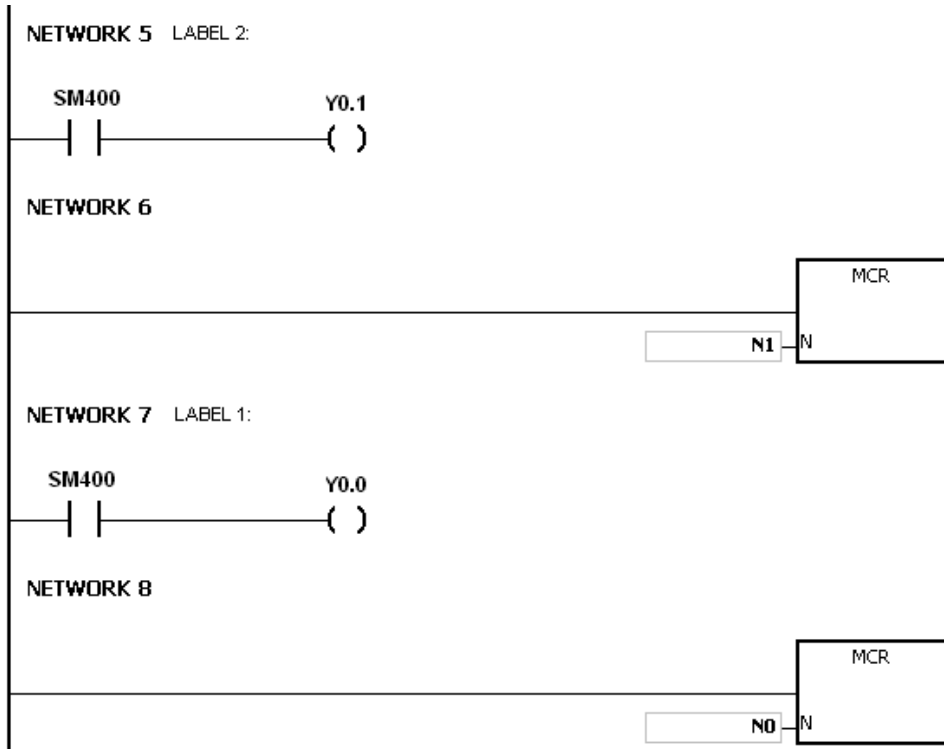


Example 2:

1. The instruction CJ between the instruction MC and the instruction MCR can be used 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 instruction MC is executed, the previous state of the switch contact is put onto the top of the stack inside the PLC. The stack is controlled by the PLC, and can not be changed by users. When the instruction MCR is executed, the previous state of the switch contact is popped 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, and the items can be popped from the stack until the stack is empty. Therefore, when CJ or CJP is used with MC and MCR, you have to be careful of the pushing of the item onto the stack and the popping of the item from the stack.



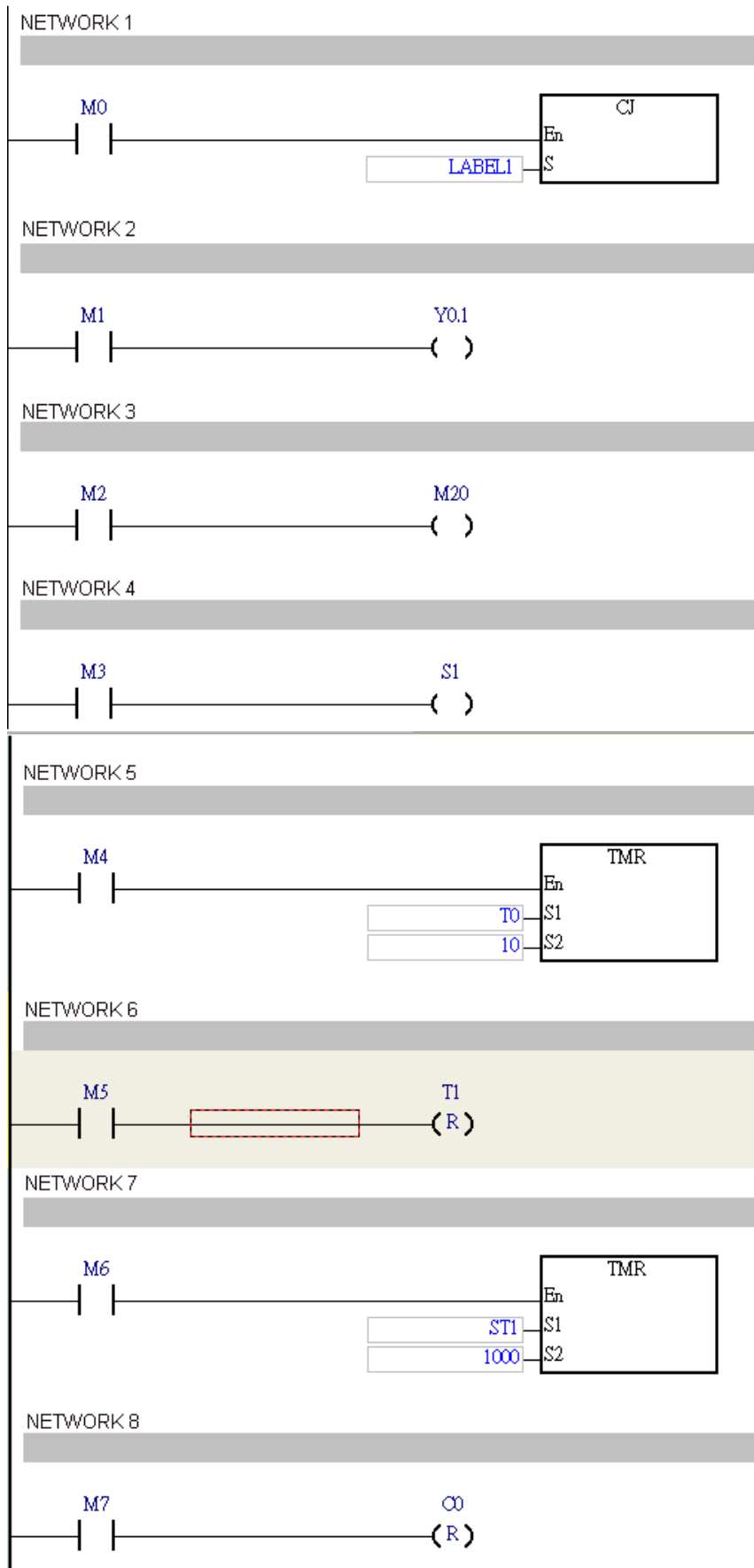


Example 3:

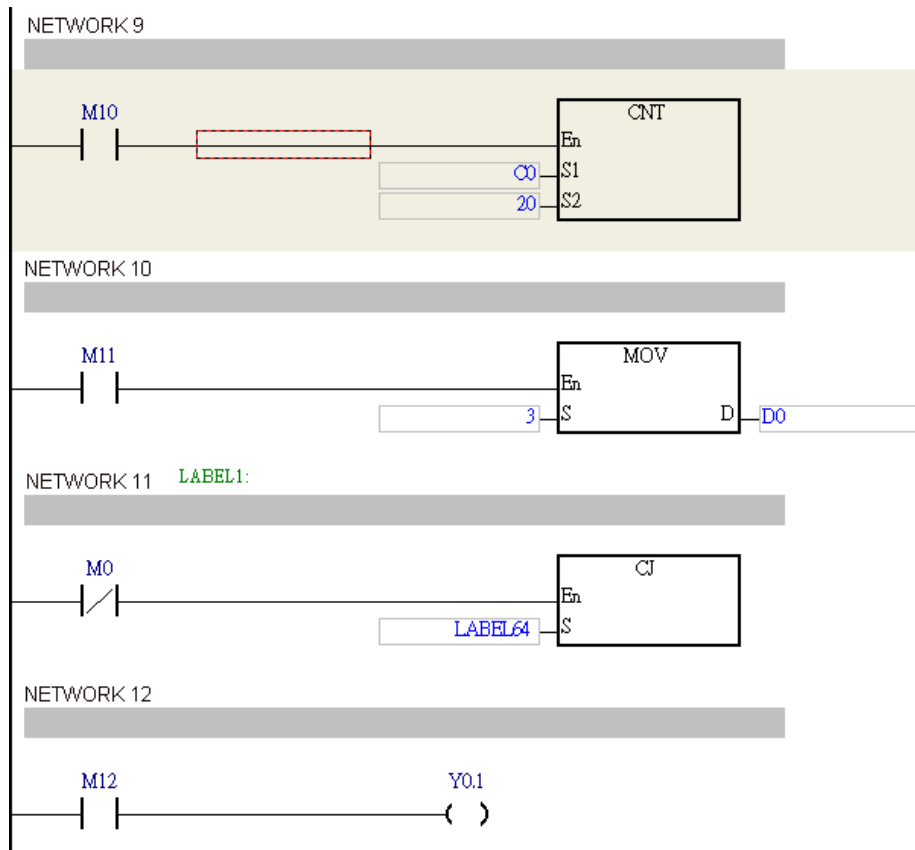
The states of the devices are listed below.

Device	State of the contact before CJ execution	State of the contact during CJ execution	State of the output coil during CJ execution
Y, M, and S	M1, M2, and M3 are OFF.	M1, M2, and M3 are switched from OFF to ON.	Y0.1 ^{*1} , M20, and S1 are OFF.
	M1, M2, and M3 are ON.	M1, M2, and M3 are switched from ON to OFF.	Y0.1 ^{*1} , M20, and S1 are ON.
Timer	M4 is OFF.	M4 is switched from OFF to ON.	The timer is not enabled.
	M4 is ON.	M4 is switched from ON to OFF	The timer stops counting immediately. When M0 is switched from ON to OFF, the timer is reset to 0.
Accumulative timer	M6 is OFF.	M6 is switched from OFF to ON.	ST1 is not enabled.
	M6 is ON.	M6 is switched from ON to OFF.	If the instruction CJ is executed after the accumulative timer is enabled, the accumulative timer stops counting.
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 is switched OFF, C0 keeps counting.
Applied instruction	M11 is OFF.	M11 is switched from OFF to ON	The applied instruction is not executed.
	M11 is ON.	M11 is switched from ON to OFF	The applied instruction which is skipped is not executed.

*1: Y0.1 is a dual output. When M0 is OFF, Y0.1 is controlled by M1. When M0 is ON, Y0.1 is controlled by M12.



3



Additional remark:

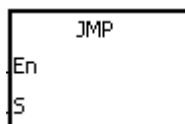
Please refer to ISPSOft User Manual for more information about the use of the label.

3

FB/FC	Instruction			Operand				Description					
FC		JMP		S				Unconditional jump					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S																	

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:

S : Jump destination

Explanation:

1. The execution of the program jumps to the part of the program specified by the pointer without any condition.
2. If the program specified by the label is prior to the instruction JMP, the watchdog timer error will occur, and the PLC will stop running. Please use the instruction carefully.
3. When the instruction is executed, the actions of the devices are as follows.
 - The state of Y, the state of M, and the state of S remain the same as those before the execution of the jump.
 - The timer stops counting.
 - If the instruction which is used to reset the timer is driven before the jump is executed, the timer will still be in the condition of being reset during the execution of the jump.
 - The general applied instructions are not executed.

FB/FC	Instruction		Operand	Description
FC		GOEND	—	Jumping to END

Pulse instruction	16-bit instruction (1 step)	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



Explanation:

1. When the condition is met, the execution of the program jumps to END in the program.
2. Function blocks and interrupt tasks do not support the instruction. Besides, the instruction can not be between the instruction FOR and the instruction NEXT.
3. When the instruction GOEND is executed, the instructions skipped are not executed, the data in all devices is unchanged, and the states of all devices are also unchanged.

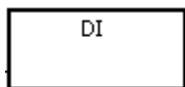
3.8 Program Execution Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>DI</u>	–	–	Disabling the interrupt	1
FC	<u>EI</u>	–	–	Enabling the interrupt	1
FC	<u>IMASK</u>	–	–	Controlling the interrupt	3

FB/FC	Instruction		Operand	Description
FC		DI	-	Disabling the interrupt

Pulse instruction	16-bit instruction (1 step)	32-bit instruction
-	AH Motion CPU	-

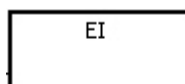
Graphic expression:



FB/FC	Instruction			Operand	Description
FC		EI		-	Enabling the interrupt

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S																	

Pulse instruction	16-bit instruction (1 step)	32-bit instruction
-	AH Motion CPU	-

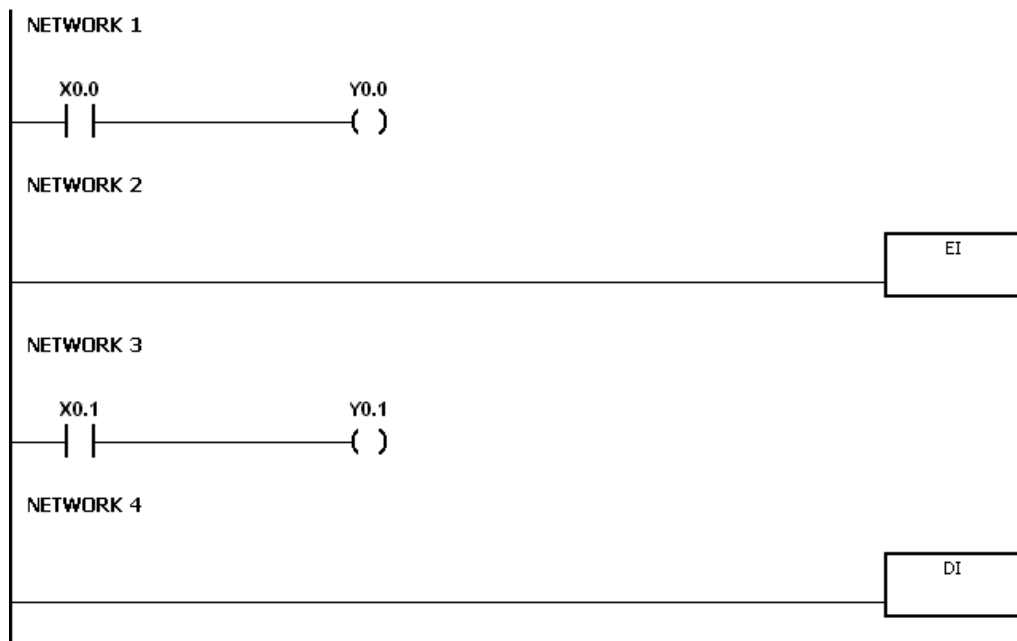
Graphic expression:**Explanation:**

1. The use of the instruction EI indicates that the interrupt task is allowed to be used in the program. (Please refer to *ISPSoft User Manual* for more information about Task.)
2. The interrupt task is allowed to be used between the instruction EI and the instruction DI in the program. When there is no part of the program in which the interrupt is disabled, you can choose not to use the instruction DI.
3. During the execution of one interrupt task, other interrupts generated will not be executed, but will be memorized. Not until the execution of the present interrupt task is complete will the next interrupt task be executed.
4. When several interrupts occur, the interrupt task which should be executed first has higher priority. When several interrupts occur simultaneously, the interrupt task whose pointer number is smaller is executed first.
5. When the interrupt task occurring between DI and EI can not be executed immediately, the interrupt request is memorized once, and the interrupt task is executed in the part of the program in which the execution of the interrupt task is allowed.
6. When the immediate I/O signal is required in the execution of the interrupt task, you can use the instruction REF in the program to refresh the state of the I/O.

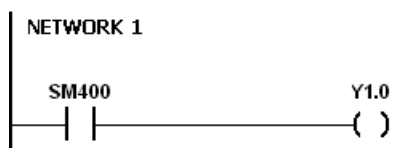
Example:

If the PLC runs and the part of the program Cyclic_0 between the instruction EI and the instruction DI is scanned, the interrupt task is executed when it is enabled. When the execution of the interrupt task is complete, the main program is executed.

The program Cyclic_0:



The interrupt task:



Additional remark:

There are 256 interrupt tasks, i.e. task I0~task I255.

1. The I/O interrupts (I0~I31)

The I/O interrupts are used by the special high-speed module. The interrupt conditions and the interrupt numbers are set in HWCONFIG in ISPSOft, and the interrupt programs are downloaded to the PLC. If the interrupt conditions are satisfied when the PLC runs, the corresponding interrupt programs will be executed.

2. The communication interrupts (I32 and I33)

The communication interrupt can be used as the instruction RS, that is, the receiving of the specific character triggers the interrupt, or can be used as the general interrupt.

Please refer to the explanation of the instruction RS for more information.

COM1: I32

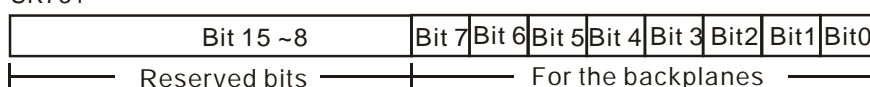
COM2: I33

3. 24 V low voltage interrupt

Whether the external 24 V voltage is normal can be checked by the terminals VS+ and VS- on AHPS05-5A. If the external 24 V voltage is abnormal, you can execute the corresponding program by means of the interrupt subroutine I34.

Note: If the external 24 V voltage of a backplane is abnormal, the corresponding bit in SR731 will be set to ON. After the external 24 V voltage of the backplane returns to normal, the bit will be set to OFF. The high 8 bits in SR731 are reserved bits.

SR731



For example:

- (a) If the external 24 V voltage of the local main backplane is abnormal, bit 0 in SR731 will be set to ON.
- (b) If the external 24 V voltage of the first local extension backplane is abnormal, bit 1 in SR731 will be set to ON.

4. The external interrupts (I40~I251)

If a peripheral device, e.g. a special I/O module, sends an interrupt request, the PLC will execute the specific interrupt task.

5. The timed interrupts (I252~I255)

Timed interrupt 0 (I252): The default value is 100 milliseconds (1~1000 milliseconds).

Timed interrupt 1 (I253): The default value is 40 milliseconds (1~1000 milliseconds).

Timed interrupt 2 (I254): The default value is 20 milliseconds (1~1000 milliseconds).

Timed interrupt 3 (I255): The default value is 10 milliseconds (1~1000 milliseconds).

The timed interrupt task is executed every specific period of time. For example, the timed interrupt task is executed every 10 milliseconds.

- The priority order is as follows.

Interrupt number	Description	Priority order
I0	I/O interrupt 0	1
I1	I/O interrupt 1	2
I2	I/O interrupt 2	3
I3	I/O interrupt 3	4
I4	I/O interrupt 4	5
I5	I/O interrupt 5	6
I6	I/O interrupt 6	7
I7	I/O interrupt 7	8
I8	I/O interrupt 8	9
I9	I/O interrupt 9	10
I10	I/O interrupt 10	11
I11	I/O interrupt 11	12
I12	I/O interrupt 12	13
I13	I/O interrupt 13	14
I14	I/O interrupt 14	15
I15	I/O interrupt 15	16
I16	I/O interrupt 16	17
I17	I/O interrupt 17	18

Interrupt number	Description	Priority order
I18	I/O interrupt 18	19
I19	I/O interrupt 19	20
I20	I/O interrupt 20	21
I21	I/O interrupt 21	22
I22	I/O interrupt 22	23
I23	I/O interrupt 23	24
I24	I/O interrupt 24	25
I25	I/O interrupt 25	26
I26	I/O interrupt 26	27
I27	I/O interrupt 27	28
I28	I/O interrupt 28	29
I29	I/O interrupt 29	30
I30	I/O interrupt 30	31
I31	I/O interrupt 31	32
I32	Reserved	33
I33	Reserved	34
I34	Reserved	35
I35~I39	Reserved	36~40
I40~I251	External interrupt	41~252
I252	Timed interrupt 0 Default value: 100 ms (1~1000 ms)	253
I253	Timed interrupt 1 Default value: 40 ms (1~1000 ms)	254
I254	Timed interrupt 2 Default value: 20 ms (1~1000 ms)	255
I255	Timed interrupt 3 Default value: 10 ms(1~1000 ms)	256

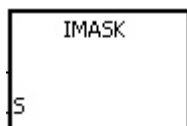
FB/FC	Instruction		Operand	Description
FC		IMASK	S	Controlling the interrupt

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S : Data source

Explanation:

1. The values of the bits in S~S+15 determine whether the interrupts are enabled or disabled. When the value of the bit is 1 and the instruction EI is executed, the corresponding interrupt is executed. When the value of the bit is 0, the corresponding interrupt can not be executed.
2. When the instruction is executed, the values in S~S+15 are transferred to SR623~SR638.
3. When the instruction is not executed, the values of the bits in SR623~SR638 determine whether the interrupts are enabled or disabled.

S	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
S +1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
S +2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
S +3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
S +4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
S +5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
S +6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
S +7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
S +8	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
S +9	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
S +10	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
S +11	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
S +12	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
S +13	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
S +14	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
S +15	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

Additional remark:

If S~S+15 exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3.9 I/O Refreshing Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>REF</u>	–	✓	Refreshing the I/O	5

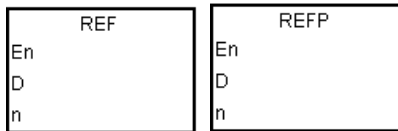
FB/FC	Instruction			Operand	Description
FC	REF	P		D, n	Refreshing the I/O

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	○	○						○					○				
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



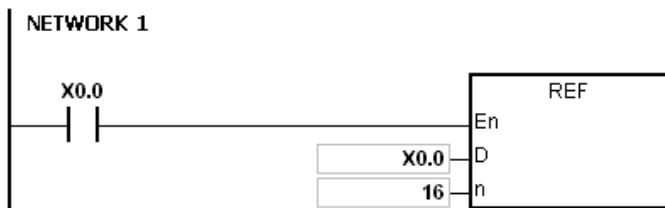
D : I/O point whose state is refreshed
n : Number of I/O points whose states are refreshed. n =1~256

Explanation:

The I/O states are not refreshed until the instruction END is executed. When the scanning of the program starts, the states of the external inputs are read and stored in the memory. After the instruction END is executed, the states of the outputs in the memory is sent to the output terminals. Therefore, you can use this instruction when you need the latest I/O data in the operation process.

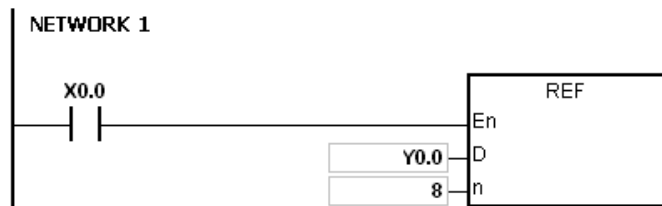
Example 1:

When X0.0 is ON, the PLC refreshed the states of the inputs X0.0~X0.15 immediately without any delay.



Example 2:

When X0.0 is ON, the output signals from Y0.0~Y0.7 are sent to the output terminals. The output signals are refreshed immediately without the need to wait for the execution of the instruction END.



Additional remark:

- 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 larger than 256, or if n is less than 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3.10 Convenience Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>ALT</u>	–	✓	Alternating between ON and OFF	3
FC	<u>TTMR</u>	–	–	Teaching timer	5
FC	<u>STMR</u>	–	–	Special timer	7
FC	<u>RAMP</u>	–	–	Ramp signal	9
FC	<u>MTR</u>	–	–	Matrix input	9
FC	<u>ABSD</u>	<u>DABSD</u>	–	Absolute drum sequencer	9
FC	<u>INCD</u>	–	–	Incremental drum sequencer	9
FC	–	<u>DPID</u>	–	PID algorithm	35
FC	–	<u>DPIDE</u>	–	PID algorithm	43

FB/FC	Instruction			Operand				Description					
FC		ALT	P	D				Alternating between ON and OFF					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●	●	●				●		●							

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



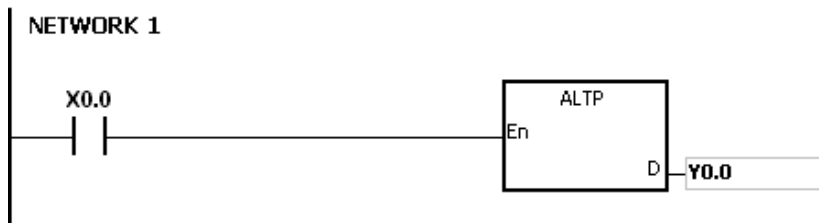
D : Destination device

Explanation:

1. When the instruction ALT is executed, the state of the device specified by D alternate between ON and OFF.
2. Generally, the pulse instruction ALTP is used.

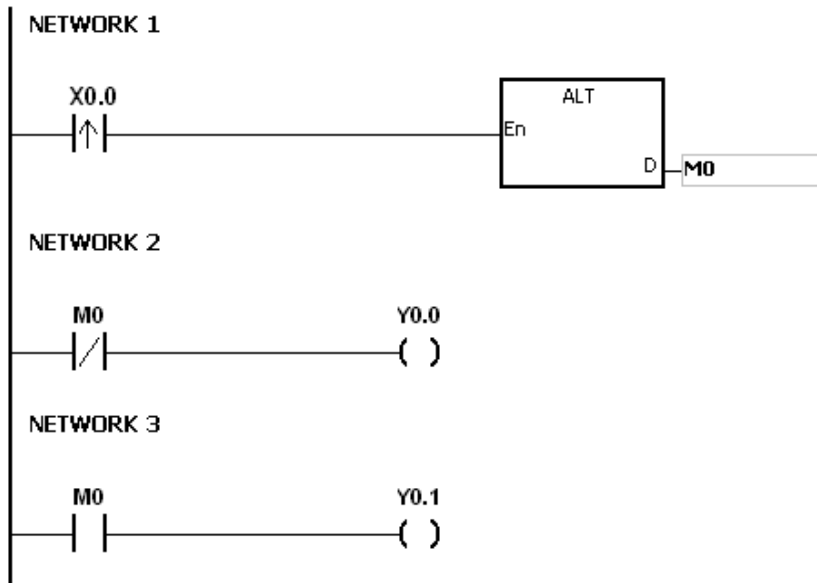
Example 1:

When X0.0 is switched from OFF to ON for the first time, Y0.0 is ON. When X0.0 is switched from OFF to ON for the second time, Y0.0 is OFF.



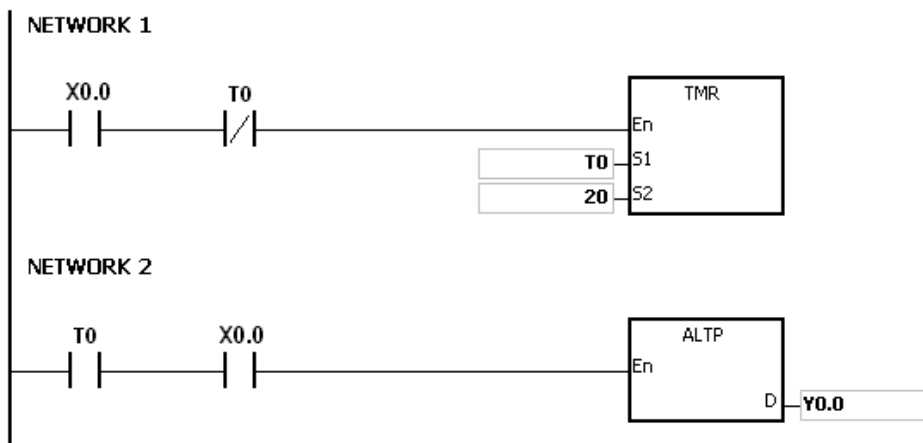
Example 2:

In the beginning, M0 is OFF. Therefore, Y0.0 is ON, and Y0.1 is OFF. When X0.0 is switched from OFF to ON for the first time, M0 is ON. Therefore, Y0.1 is ON, and Y0.0 is OFF. When X0.0 is switched from OFF to ON for the second time, M0 is OFF. Therefore, Y0.0 is ON, and Y0.1 is OFF.



3 Example 3:

When X0.0 is ON, T0 generates a pulse every two seconds. The output Y0.0 alternates between ON and OFF according to the pulses generated by T0.



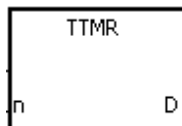
FB/FC	Instruction		Operand				Description				
FC		TTMR	n, D				Teaching timer				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
n	●	●						●	●		●		●	○	○		
D	●	●						●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



n : Multiplier

D : Device in which the time is stored

Explanation:

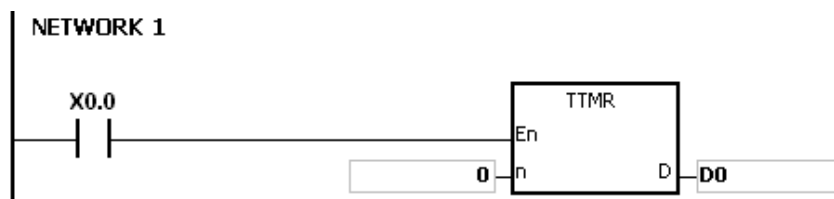
1. A second is taken as the timing unit. The time for which the button switch has been turned ON is multiplied by n, and the product is stored in D. D+1 is for system use only. When the instruction is executed, the value in D+1 can not be altered. Otherwise, the time will be counted incorrectly.
2. When the conditional contact is ON, D is reset to 0.
3. Setting the multiplier: When n is 0, D takes 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 takes 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 takes 10 milliseconds as the timing unit.

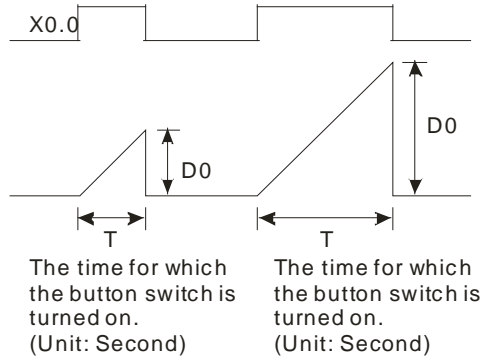
n	D
K0 (unit: 1 second)	1×T
K1 (unit: 100 milliseconds)	10×T
K2 (unit: 10 milliseonds)	100×T

4. When the on-line editing is used, please reset the conditional contact to initialize the instruction.
5. The operand n should be within the range between 0 and 2.

Example 1:

1. The time for which the button switch X0.0 has been turned ON is multiplied by n, and the product is stored in D0.
2. When X0.0 is switched OFF, the value in D0 is unchanged.





Additional remark:

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, or if n is larger than 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 will be ARRAY [2] of WORD/INT.

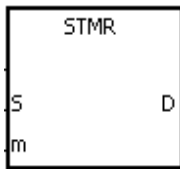
FB/FC	Instruction		Operand				Description				
FC		STMR	S, D, m				Special timer				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S					○												
D	●	●	●	●				●	●	●			●				
m	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S : Timer number. **S** =T0~T2047

D : Output device

m : Setting value of the timer

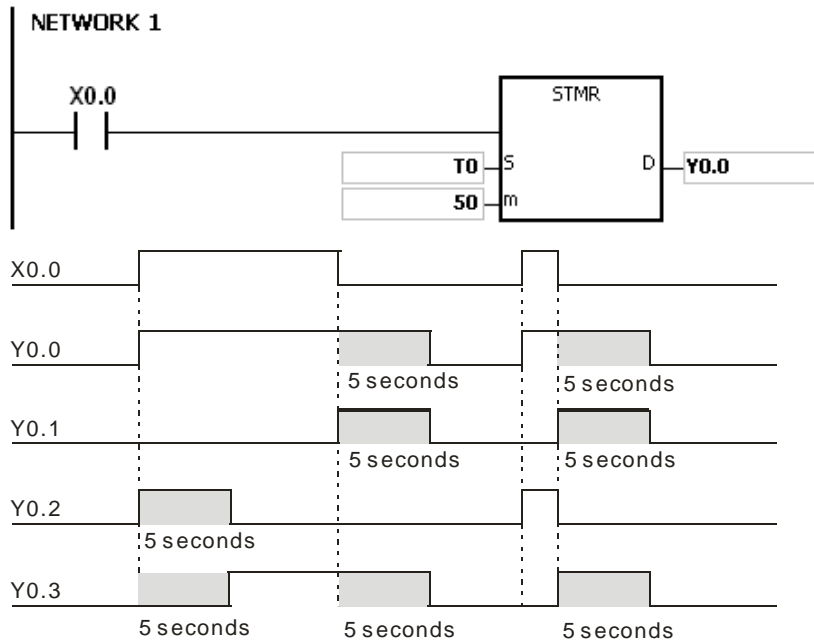
Explanation:

- The instruction STMR is used to generate the off-delay relay, the one-shot circuit, and the flashing circuit.
- The timer specified by the instruction TMR takes 100 milliseconds as the timing unit.
- The timer specified by the instruction STMR can not be used repeatedly.
- D occupies four consecutive devices.
- Before the instruction is executed, please reset D~D+3.
- 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.
 - When the on-line editing is used, please reset the conditional contact to initialize the instruction.
 - The operand m should be within the range between 1 and 32767.

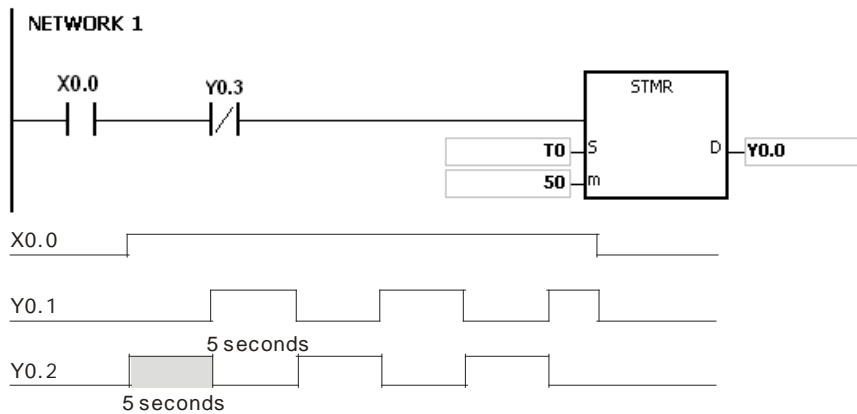
Example:

- When X0.0 is ON, the instruction STMR specifies the timer T0, and the setting value of T0 is five seconds.
- Y0.0 is the off-delay contact. When X0.0 is switched from OFF to ON, Y0.0 is ON. Five minutes after X0.0 is switched from ON to OFF, Y0.0 is OFF.
- When X0.0 is switched from ON to OFF, Y0.0 is ON for five seconds.

4. When X0.0 is switched from OFF to ON, Y0.2 is ON for five seconds.
5. Five seconds after X0.0 is switched from OFF to ON, Y0.3 is ON. Five seconds after X0.0 is switched from ON to OFF, Y0.3 is OFF.



6. When the conditional contact X0.0 is followed by the normally-closed contact Y0.0, the flashing currents pass through Y0.1 and Y0.2. When X0.0 is switched OFF, Y0.0, Y0.1, and Y0.3 are switched OFF, and T0 is reset to 0.



Additional remark:

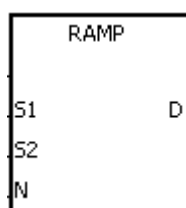
1. If D+3 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If m is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If you declare the operand D in ISPSOft, the data type will be ARRAY [4] of BOOL.

FB/FC	Instruction		Operand				Description				
FC		RAMP	S₁, S₂, D, N				Ramp signal				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●						●	●		●		●				
D	●	●						●	●		●		●				
N	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

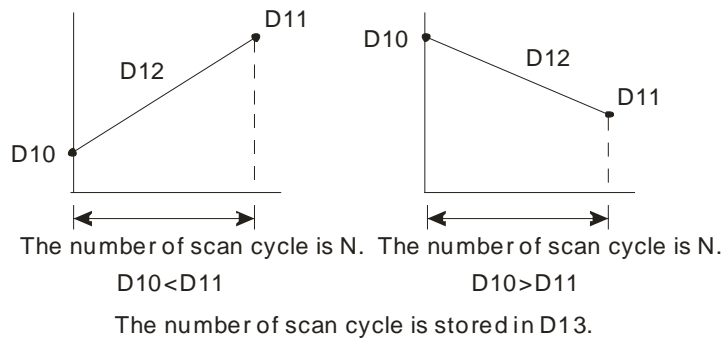
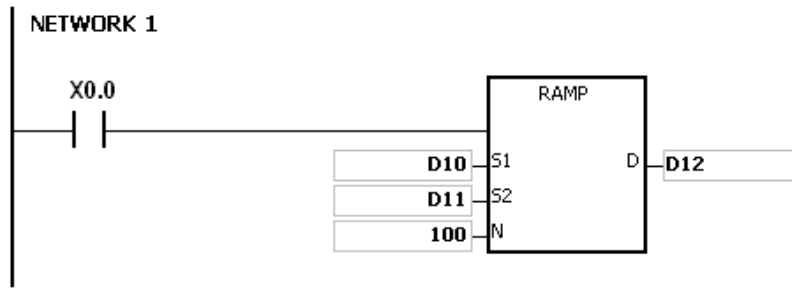
Graphic expression:**S₁** : Initial value of the ramp signal**S₂** : Final value of the ramp signal**N** : Number of scan cycles**D** : Duration of the ramp signal**Explanation:**

- The instruction is used to get the slope. The slope is linear, and has an absolute relationship with the scan time.
- The initial value of the ramp signal and the final value of the ramp signal are written into **S₁** and **S₂** respectively in advance. When the driving contact is ON, the value in **D** increases from the setting value in **S₁** to the setting value in **S₂**. The number of scan cycles is stored in D+1. When the value in **D** is equal to that in **S₂**, or when the value on D+1 is equal to n, SM687 is ON.
- When the conditional contact is not enabled, the value in **D**, **D+1** is 0, and SM687 is OFF.
- When the on-line editing is used, please reset the conditional contact to initialize the instruction.
- Please refer to ISPSOFT User Manual for more information related to the fixing of the scan time.
- The operand n should be within the range between 1 and 32767.

Example:

When the instruction is used with the analog signal output, the action of cushioning the start/stop can be executed.

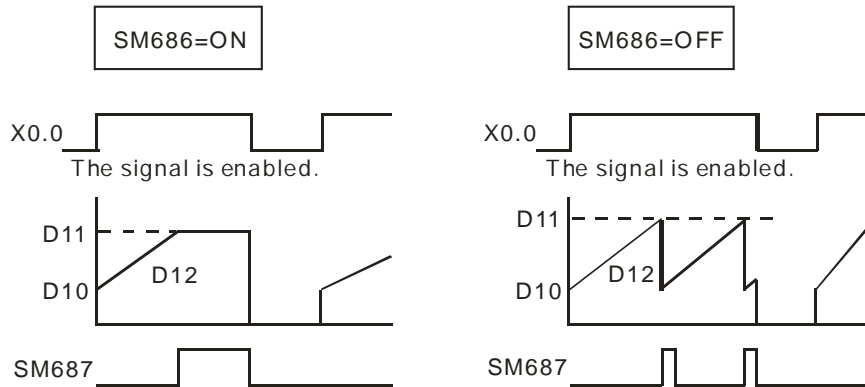
- Suppose the instruction is being executed. When X0.0 is switched OFF, the execution of the instruction stops. When X0.0 is ON again, SM687 is OFF, D12 is reset to the setting value in D10, D13 is reset to 0, and the calculation is restarted.
- When SM686 is OFF, SM687 is ON, D12 is reset to the setting value in D10, and D13 is reset to 0.



3

Additional remark:

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. If you declare the operand D in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
4. When SM686 is ON/OFF, the value in D12 changes as follows.



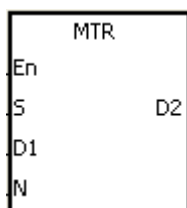
FB/FC	Instruction		Operand				Description			
FC		MTR	S, D₁, D₂, N				Matrix input			

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●													
D₁	●													
D₂	●													
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	○																
D₁		○															
D₂	○	○	○	○				○	○			○					
N	●	●	●	●				●	●		●	●	○	○			

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



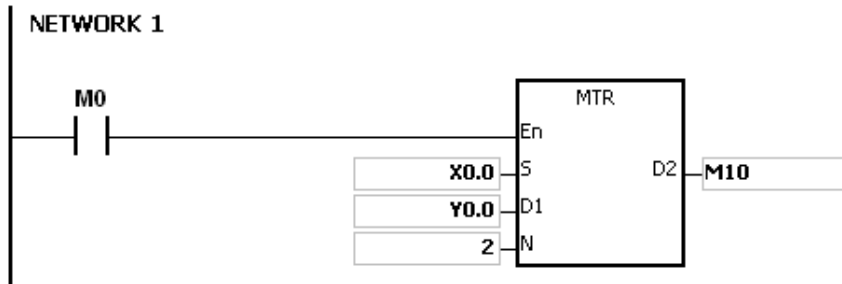
- S** : Initial input device in the matrix scan
- D₁** : Initial output device in the matrix scan
- D₂** : Initial corresponding device in the matrix scan
- N** : Number of rows which are scanned

Explanation:

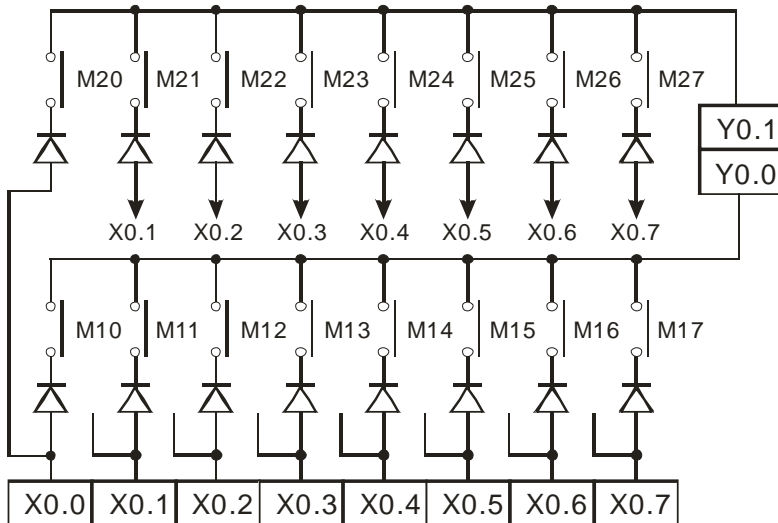
1. S specifies the initial input device in the matrix scan. The eight devices starting from the device specified by S are the input devices in the matrix scan.
2. D1 specifies the transistor output device Y as the initial device in the matrix scan. When the conditional contact is OFF, the states of the n devices starting from D1 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 to the nth 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 D1 to form the n rows of switches. The states of the n rows of switches are read in the matrix scan, and stored in the devices starting from the device specified by D2.
5. When the instruction is used, you can connect at most 8 rows of input switches in parallel to get 64 inputs (8×8=64).
6. The interval between the time when the instruction is executed and the next time when it is executed should be longer than the time it takes for the states of the I/O points on the module to be refreshed. Otherwise, the correct states of the inputs can not be read.
7. Generally, the conditional contact used in the instruction is the normally-open contact SM400.
8. The operand n should be within the range between 2 and 8.

Example 1:

- When M0 is ON, the instruction MTR is executed. The states of the two rows of switches are read in order, and stored in the internal relays M10~M17 and M20~M27 respectively.

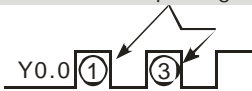


- The diagram below is the external wiring diagram of the 2-by-8 matrix input circuit which is composed of X0.0~X0.7 and Y0.0~Y0.7. The corresponding internal relays of the 16 switches are M10~M17 and M20~M27.



- The eight input devices starting from X0.0 are connected to the two output devices starting from Y0.0 to form the two rows of switches. The states of the two rows of switches are read in the matrix scan, and stored in the devices starting from M10 specified by D₂. That is, the states of the first row of switches are stored in M10~M17, and the states of the second row of switches are stored in M20~M27.

The first row of input signals are read.



The second row of input signals are read.



Additional remark:

- If S+7, D1+n-1, or D2+(n*8)-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If n is less than 2, or if n is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
- If you declare the operand S in ISPSOft, the data type will be ARRAY [8] of BOOL.

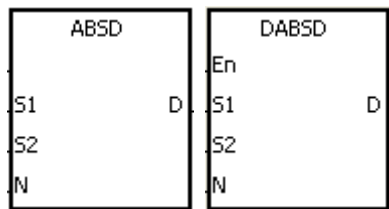
FB/FC	Instruction			Operand				Description				
FC	D*	ABSD		S ₁ , S ₂ , D, N				Absolute drum sequencer				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D	●/●*													
N		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●		●				
D	●	●	●	●				●	●	●			●				
N	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



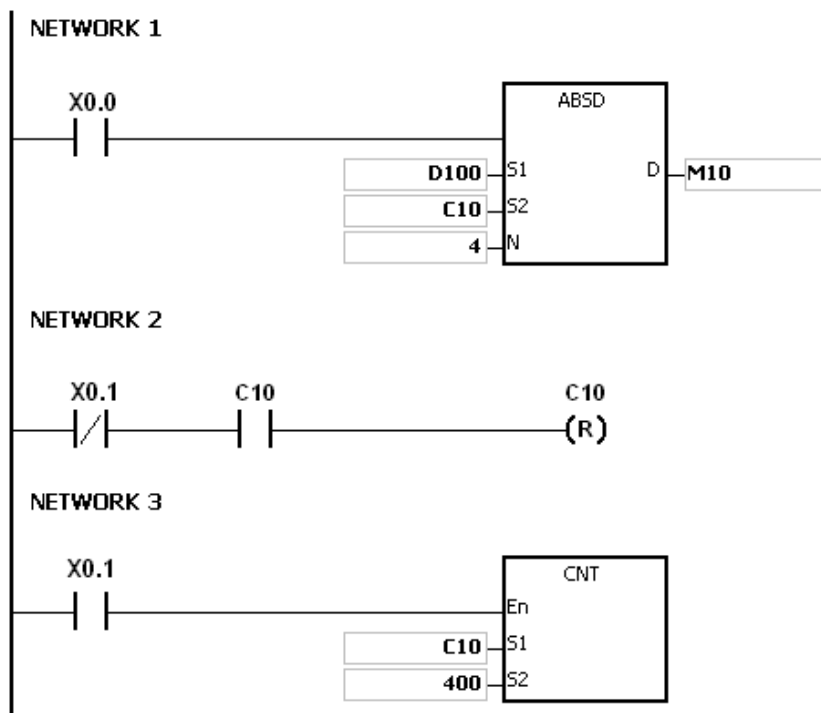
- S₁ : Initial device in the comparison
- S₂ : Comparison value
- D : Comparison result
- N : Number of comparison groups

Explanation:

- The instruction ABSD is used to generate multiple pulses corresponding to the current values of the counter.
- Only the instruction DABSD can use the 32-bit counter.
- When the instruction ABSD is used, n should be within the range between 1 and 256. When the instruction DABSD is used, n should be within the range between 1 and 128.

Example 1:

- Before the instruction ABSD is executed, the instruction MOV is used to write 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.
- When X0.0 is ON, the current value of the counter C10 is compared with the maximum values and the minimum values in D100~D107, and the comparison results are stored in M10~M13.
- When X0.0 is OFF, the original states of M10~M13 are unchanged.

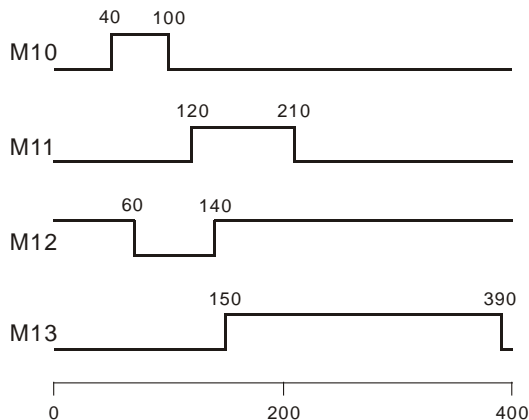


4. When the current value of C10 is within the range between the minimum value and the maximum value, M10~M13 are ON. Otherwise, M10~M13 are OFF.

Minimum value	Maximum value	Current value of C10	Output
D100=40	D101=100	$40 \leq C10 \leq 100$	M10=ON
D102=120	D103=210	$120 \leq C10 \leq 210$	M11=ON
D104=140	D105=170	$140 \leq C10 \leq 170$	M12=ON
D106=150	D107=390	$150 \leq C10 \leq 390$	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	$40 \leq C10 \leq 100$	M10=ON
D102=120	D103=210	$120 \leq C10 \leq 210$	M11=ON
D104=140	D105=60	$60 \leq C10 \leq 140$	M12=OFF
D106=150	D107=390	$150 \leq C10 \leq 390$	M13=ON



Additional remark:

1. If $S+2*n-1$ used in the instruction ABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If $S+4*n-1$ used in the instruction DABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If $D+n-1$ used in the instruction ABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If $D+2*n-1$ used in the instruction DABSD exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. If n used in the instruction ABSD 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.
6. If n used in the instruction DABSD is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

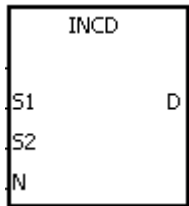
FB/FC	Instruction			Operand				Description						
FC	INCD			S₁, S₂, D, N				Incremental drum sequencer						

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●					●							
D	●													
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●		●	●		●		●				
D	●	●	●	●				●	●	●			●				
N	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S₁ : Initial device in the comparison
- S₂ : Counter number
- D : Comparison result
- N : Number of comparison groups

Explanation:

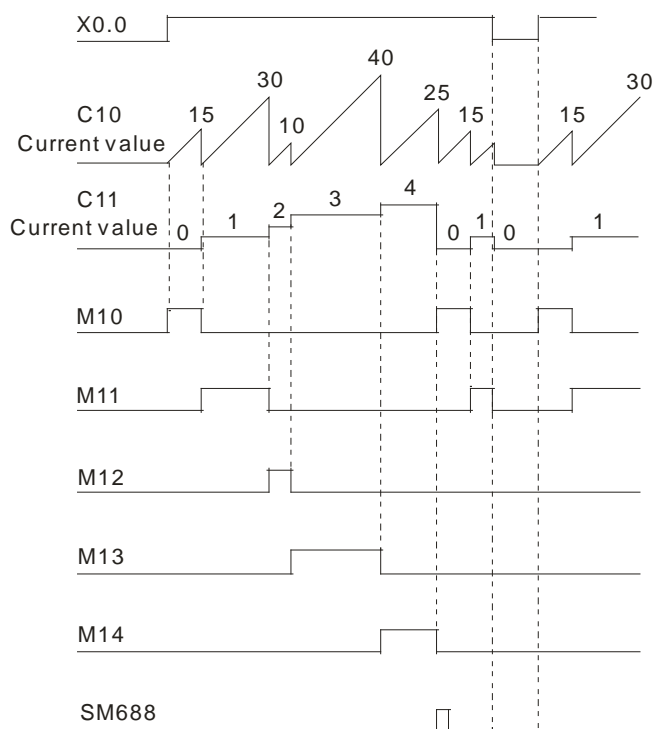
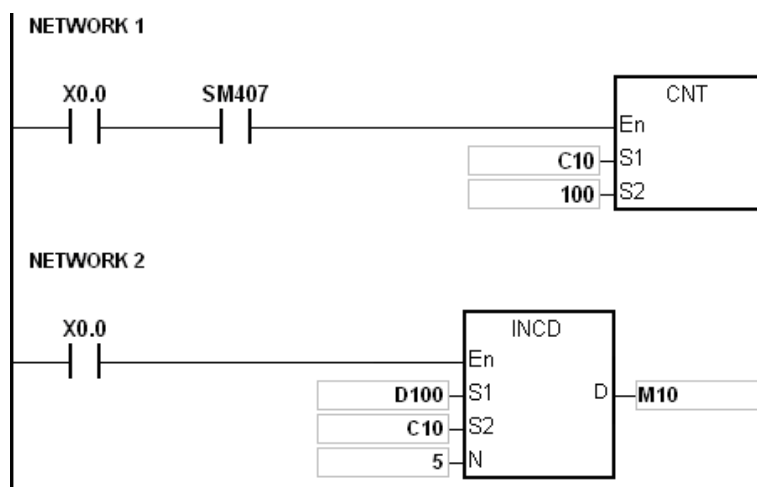
1. The instruction INCD is used to generate multiple pulses for a pair of counters.
2. The current value of S2 is compared with the setting value in S1. When the current value matches the setting value, the current value of S2 is reset to 0, and the current comparison group number is stored in S2+1.
3. After the comparison between the current values of S2 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 S2 is 0, the value in S2+1 is 0, D~D+n-1 are OFF, and SM688 is OFF.
5. When the on-line editing is used, please reset the conditional contact to initialize the instruction.
6. The operand n should be within the range between 1 and 256.

Example:

1. Before the instruction INCD is executed, the instruction MOV is used to write the setting values in D100~D104. The values in D100~D104 are 15, 30, 10, 40, and 25 respectively.
2. The current values of C10 is compared with the setting values in D100~D104. When the current value matches the setting value, C10 is reset to 0, and counts again.
3. The current comparison group number is stored in C11.
4. When the value in C11 changes by 1, M10~M14 act correspondingly. Refer to the timing diagram below.
5. When the comparison between the current values of C10 and the values in D100~D104 is complete, SM688

is ON for a scan cycle.

6. When X0.0 is switched from ON to OFF, C10 and C11 are reset to 0, and M10~M14 are switched OFF. When X0.0 is ON again, the execution of the instruction starts from the beginning.



Additional remark:

1. If S2+1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If S1+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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
5. If you declare the operand S2 in ISPSOft, the data type will be ARRAY [2] of WORD/INT.

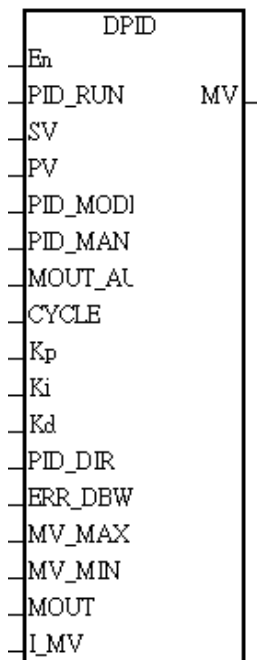
FB/FC	Instruction			Operand						Description				
FC	D*	PID		Refer to below table						PID algorithm				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
PID_RUN	●*													
SV										●*				
PV										●*				
PID_MODE			●*					●*						
PID_MAN	●*													
MOUT_AUTO	●*													
CYCLE			●*					●*						
Kp										●*				
Ki										●*				
Kd										●*				
PID_DIR	●*													
ERR_DBW										●*				
MV_MAX										●*				
MV_MIN										●*				
MOUT										●*				
I_MV										●*				
MV										●*				

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
PID_RUN	●	●	●					●	●	●			●				
SV								●	●				●				
PV								●	●				●				
PID_MODE								●	●				●				
PID_MAN	●	●	●					●	●	●			●				
MOUT_AUTO	●	●	●					●	●	●			●				
CYCLE								●	●				●				
Kp								●	●				●				
Ki								●	●				●				
Kd								●	●				●				
PID_DIR	●	●	●					●	●	●			●				
ERR_DBW								●	●				●				
MV_MAX								●	●				●				
MV_MIN								●	●				●				
MOUT								●	●				●				
I_MV								●	●				●				
MV								●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AH Motion CPU

Graphic expression:



PID_RUN	:	Enabling the PID algorithm
SV	:	Target value (SV)
PV	:	Process value (PV)
PID_MODE	:	PID control mode
PID_MAN	:	PID A/M mode (PID_MAN)
MOUT_AUTO	:	MOUT_AUTO
CYCLE	:	Sampling time (CYCLE)
K_p	:	Proportional gain (K _p)
K_i	:	Integral gain (K _i)
K_d	:	Derivative gain (K _d)
PID_DIR	:	PID forward/reverse direction (PID_DIR)
ERR_DBW	:	Range within which the error value is count as 0 (ERR_DBW)
MV_MAX	:	Maximum output value (MV_MAX)
MV_MIN	:	Minimum output value (MV_MIN)
MOUT	:	Manual output value (MOUT)
I_MV	:	Accumulated integral value (I_MV)
MV	:	Output value (MV)

Explanation:

- The instruction is used to implement the PID algorithm. After the sampling time is reached, the PID algorithm is implemented. PID stands for Proportional, Integral, Derivative. The PID control is widely applied to mechanical equipments, pneumatic equipments, and electronic equipments.
- The setting of the parameters is as follows.

Interface	Function	Setting range	Description
PID_RUN	Enabling the PID algorithm	True: The PID algorithm is implemented. False: The output value (MV) is reset to 0, and the PID algorithm is not implemented.	
SV	SV	Range of single-precision floating-point numbers	Target value
PV	PV	Range of single-precision floating-point numbers	Process value

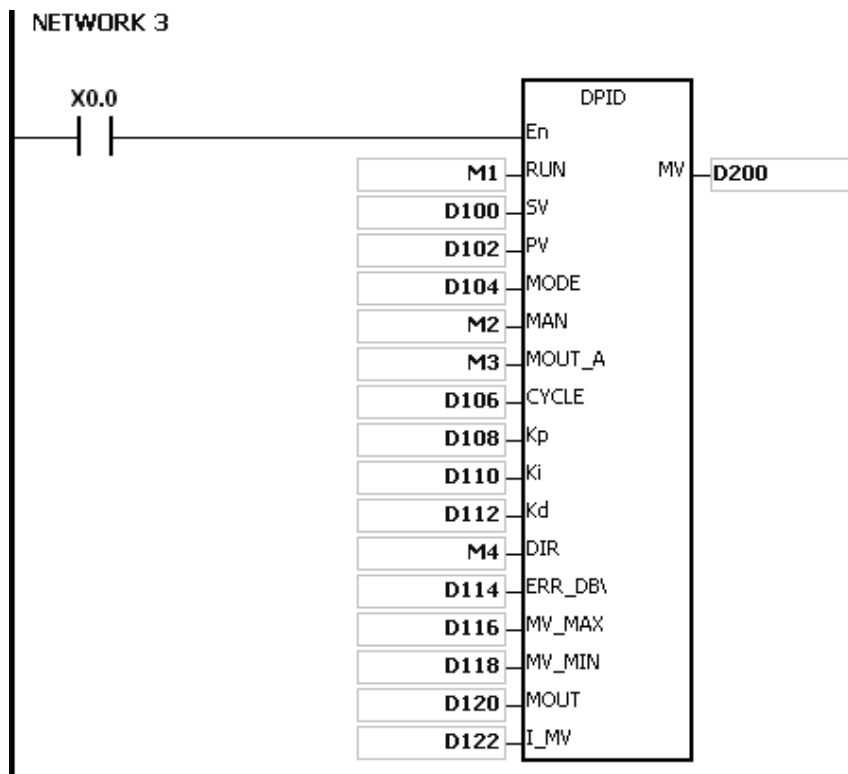
Interface	Function	Setting range	Description
PID_MODE	PID control mode		<p>0: Automatic control When PID_MAN is switched from ON to OFF, the output value (MV) then is involved in the automatic algorithm.</p> <p>1: The parameters are tuned automatically for the temperature control. When the tuning of the parameters is complete, the device is automatically set to 0, and is filled in with appropriate parameters K_P, K_I, and K_D.</p> <p>2: Automatic control When PID_MAN is switched from ON to OFF, the MV involved in the internal algorithm is involved in the automatic algorithm. If the setting value exceeds the range, it will be counts as 0.</p>
PID_MAN	PID A/M mode		<p>True: Manual The MV is output according to the MOUT, but it is still within the range between the MV_MIN and the MV_MAX. When PID_MODE is set to 1, the setting is ineffective.</p> <p>False: Automatic The MV is output according to the PID algorithm, and the output value is within the range between MV_MIN and MV_MAX.</p>
MOUT_AUTO	MOUT automatic change mode		<p>True: Automatic The MOUT varies with the MV.</p> <p>False: Normal The MOUT deos not vary with the MV.</p>
Cycle	Sampling time (T_S)	1~2,000 (unit: 10ms)	When the instruction is scanned, the PID algorithm is implemented according to the sampling time, and the MV is refreshed. If T_S is less than 1, it will be count as 1. If T_S is larger than 2000, it will be count as 2000. When the instruction PID is used in the interval interrupt task, the sampling time is the same as the interval between the timed interrupt tasks.
K_p	Proportional gain (K_p)	Range of positive single-precision floating-point numbers	It is the magnified proportional value of the error between the SV and the PV. If the magnified proportional value of the error is less than 0, the K_p will be count as 0.

Interface	Function		Setting range	Description
K_i	Integral gain (K _i)		Range of positive single-precision floating-point numbers	It is the integral gain (K _i). If the integral gain is less than 0, the K _i will be count as 0.
K_d	Derivative gain (K _d)		Range of positive single-precision floating-point numbers	It is the derivative gain (K _d). If the derivative gain is less than 0, the K _d will be count as 0.
PID_DIR	PID forward/reverse direction		True: Reverse action (E=SV-PV) False: Forward action (E=P _V -SV)	
ERR_DBW	Range within which the error value is count 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 is not enabled. For example, the E within the range between -5 and 5 will be count as 0 if the setting value is 5 or -5.
MV_MAX	Maximum output value		Range of single-precision floating-point numbers	Suppose MV_MAX is set to 1,000. When the MV is larger than 1,000, 1,000 is output. The value in MV_MAX should be larger than that in MV_MIN . Otherwise, the maximum MV and the minimum MV will be reversed.
MV_MIN	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 output.
MOUT	Manual output value		It is used with the PID_MAN mode. You set the MV directly.	
I_MV (It occupies six consecutive 32-bit devices.)	I_MV	The accumulated integral value is temporarily stored in it.	Range of single-precision floating-point numbers	The accumulated integral value is only for reference. It still can be cleared or modified according to users' need. When the MV is larger 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 PV is temporarily stored in it.	The previous PV is only for reference. It still can be modified according to users' need.	
	I_MV+2	For system use only		

Interface	Function	Setting range	Description
	I		
	I_MV+5		
MV	MV		The MV is within the range between the MV_MIN and the MV_MAX.

Example:

1. Before the instruction PID is executed, the setting of the parameters should be complete.
2. When X0.0 is ON, the instruction is executed. When M1 is ON, the PID algorithm is implemented. When M1 is OFF, the MV is 0, and the MV is stored in D200. When X0.0 is switched OFF, the instruction is not executed, and the previous data is unchanged.



Additional remark:

1. The instruction can be used several times, but the registers specified by I_MV~I_MV+5 can not be the same.
2. I_MV occupies 12 registers. I_MV used in the instruction PID in the above example occupies D122~D133.
3. The instruction PID only can be used in the cyclic task and the interval interrupt task. When the instruction PID is used in the interval interrupt task, the sampling time is the same as the interval between the timed interrupt tasks.
4. When the instruction is scanned, the PID algorithm is implemented according to the sampling time, and the MV is refreshed. When the instruction is used in the interrupt task, the sampling time is the same as the interval between the timed interrupt tasks. The PID algorithm is implemented according to the interval between the timed interrupt tasks.
5. Before the PID algorithm is implemented, the process value used in the instruction PID has to be a stable value. When you need the input value in the module to implement the PID algorithm, you have to notice the time it takes for the analog input to be converted into the digital input.

● **The PID algorithm:**

1. When PID_MODE is set to 0 or 2, the PID control mode is the automatic control mode.
2. When PID_MODE is set to 1, the PID control mode is the auto tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode becomes the automatic control mode.

The PID algorithm includes the forward action and the reverse action. Whether the action is the forward one or the reverse one depends on the setting of **PID_DIR**.

The PID algorithm is as follows.

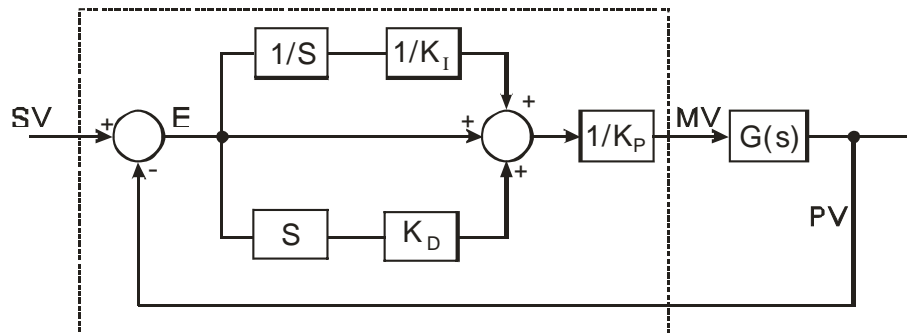
$$MV = K_p E(t) + K_I \int_0^t E(t) dt + K_D * \frac{dE(t)}{dt}$$

$E(t)S$ represents the derivative value of $E(t)$, and $E(t)\frac{1}{S}$ represents the integral value of $E(t)$.

Action direction	PID algorithm
Forward action	$E(t) = PV(t) - SV(t)$
Reverse action	$E(t) = SV(t) - PV(t)$

- (a) Control diagram: S represents the derivative action, and is defined as (Current $E(t)$ -previous $E(t)$)/Sampling time. $1/S$ represents the integral action, and is defined as (Previous integral value+Error value)×Sampling time. $G(S)$ represents the plant.

The instruction PID is inside the dotted line.



- (b) The symbols:

MV : Output value

$E(t)$: Error value

Forward action $E(t) = PV - SV$

Reverse action $E(t) = SV - PV$

K_p : Proportional gain

PV : Process value

SV : Target value

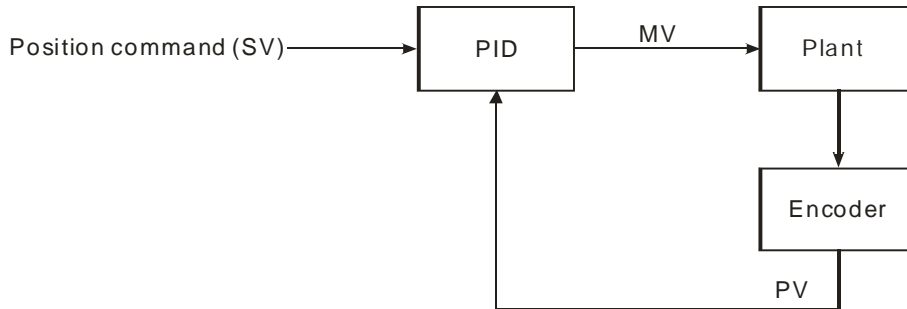
K_D : Derivative gain

K_I : Integral gain

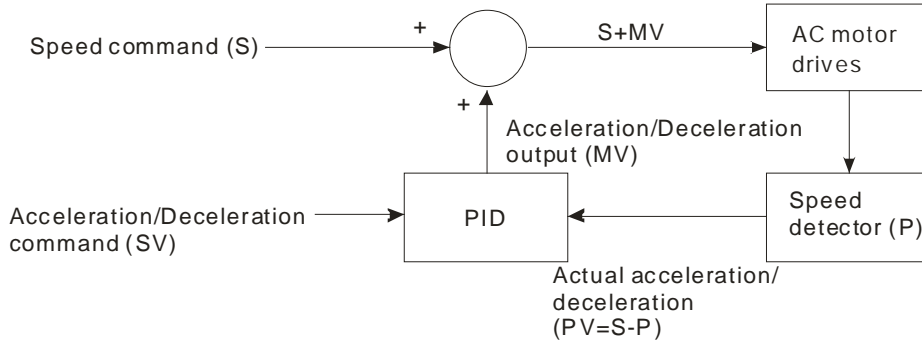
● **Points to note:**

- (a) Because the instruction PID can be used in a lot of controlled environments, you have to choose the control function appropriately. For example, to prevent the improper control from occurring, PID_MODE can not be used in the motor controlled environment when it is set to 1.
- (b) When you tune the parameters KP, KI, and KD (PID_MODE is set to 0 or 2), you have to tune the KP first (according to the experience), and then set set the KI and the KD to 0. When you can handle the control, you can increase the KI and the KD, as illustrated in example four below. When the KP is 100, 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 th proportional gain is larger than 100%, the error value is increased.
- (c) To prevent the parameters which have been tuned automatically from disappearing after a power cut, you have to store the parameters in the latched data registers when is PID_MODE set to 1. The parameters which have been tuned automatically are not necessarily suitable for every controlled environment. Therefore, you can modify the parameters which have been tuned automatically. However, it is suggested that you only modify the KI or the KD.
- (d) The instruction should be used with many parameters. To prevent the improper control from occurring, please do not set the parameters randomly.

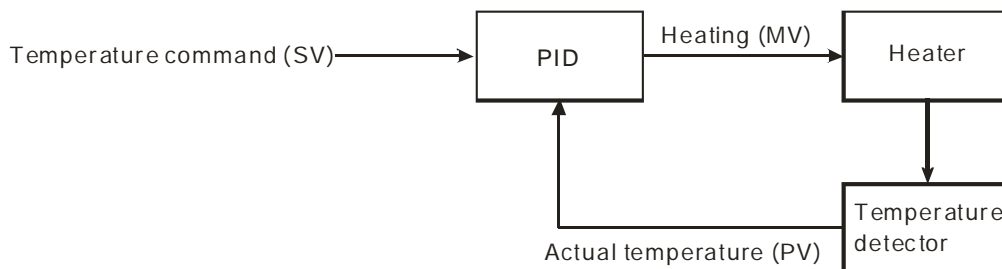
Application 1: The use of the instruction PID in the position control (PID_MODE is set to 0 or 2.)



Application 2: The instruction PID is used with the AC motor drives. (PID_MODE is set to 0 or 2.)



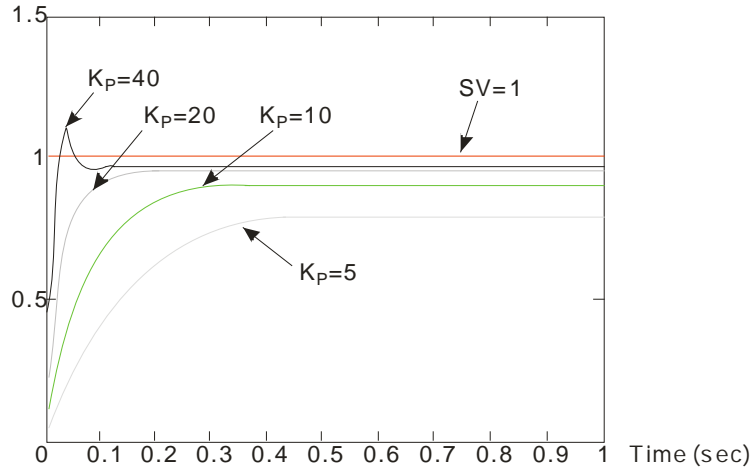
Application 3: The use of the instruction PID in the temperature control (PID_MODE is set to 0 or 2.)



Application 4: The steps of tuning the parameters used with the instruction PID

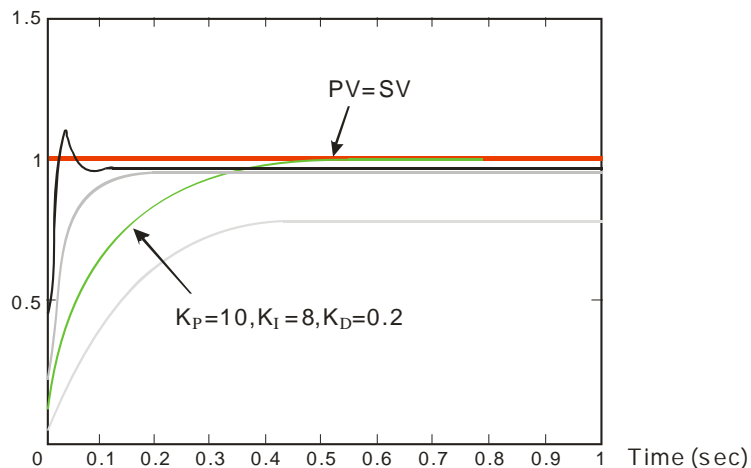
Suppose that the transfer function of the plant is the first-order function $G(s) = \frac{b}{s+a}$, the SV is 1, the sampling time T_s is 10 milliseconds. It is suggested that the steps of tuning the parameters are as follows.

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 diagram below.



Step 2: When the K_P is 40, there is overreaction. Thus, the K_P is not chosen. When the K_P is 20, the reaction curve of the PV is close to the SV, and there is no overreaction. However, due to the fast start-up, the transient output value (MV) is big. The K_P is not chosen, either. When the K_P is 10, the reaction curve of the PV approaches the SV smoothly. Therefore, the K_P is chosen. When the K_P is 5, the reaction is slow. Thus, the K_P is not chosen.

Step 3: After the K_P is set to 10, increase the K_I . For example, the K_I is set to 1, 2, 4, and 8 successively. The K_I should not be larger than the K_P . Then, increase the K_D . For example, the K_D is set to 0.01, 0.05, 0.1, and 0.2 successively. The K_D should not be larger than ten percent of the K_P . Finally, the relation between the PV and the SV is present in the following diagram.



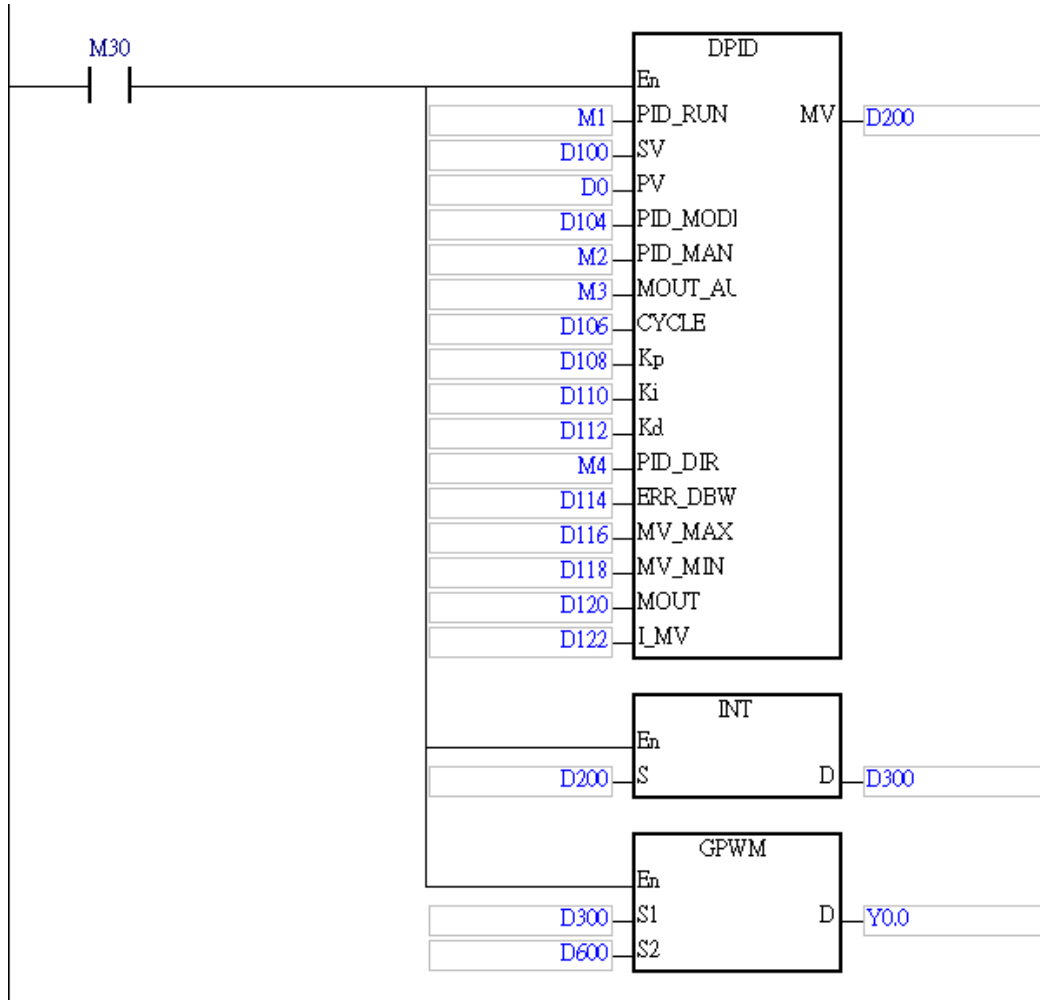
Note: The example is only for reference. You have to tune the parameters properly according to the practical condition of the control system.

Application Case: Using the auto tuning function to control the temperaturePurpose:

Using the auto tuning function to calculate the most appropriate parameters for the PID operation

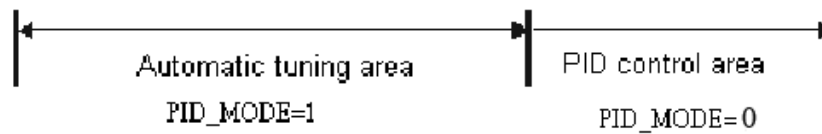
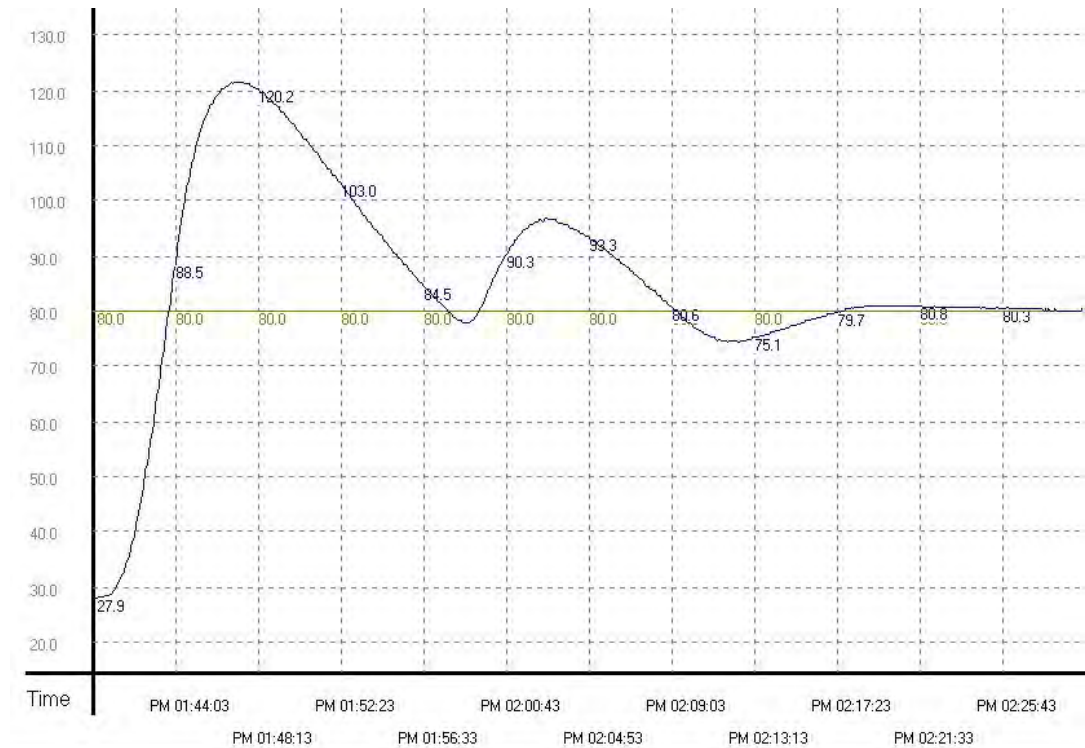
Explanation:

If you are not familiar with the characteristics of the temperature environment for the first time, you can use the auto tuning function to make an initial adjustment (**PID_MODE** is set to 1). After the tuning of the parameters is complete, **PID_MODE** will be set to 0. In this case, the controlled environment is an oven. The program example is as below.

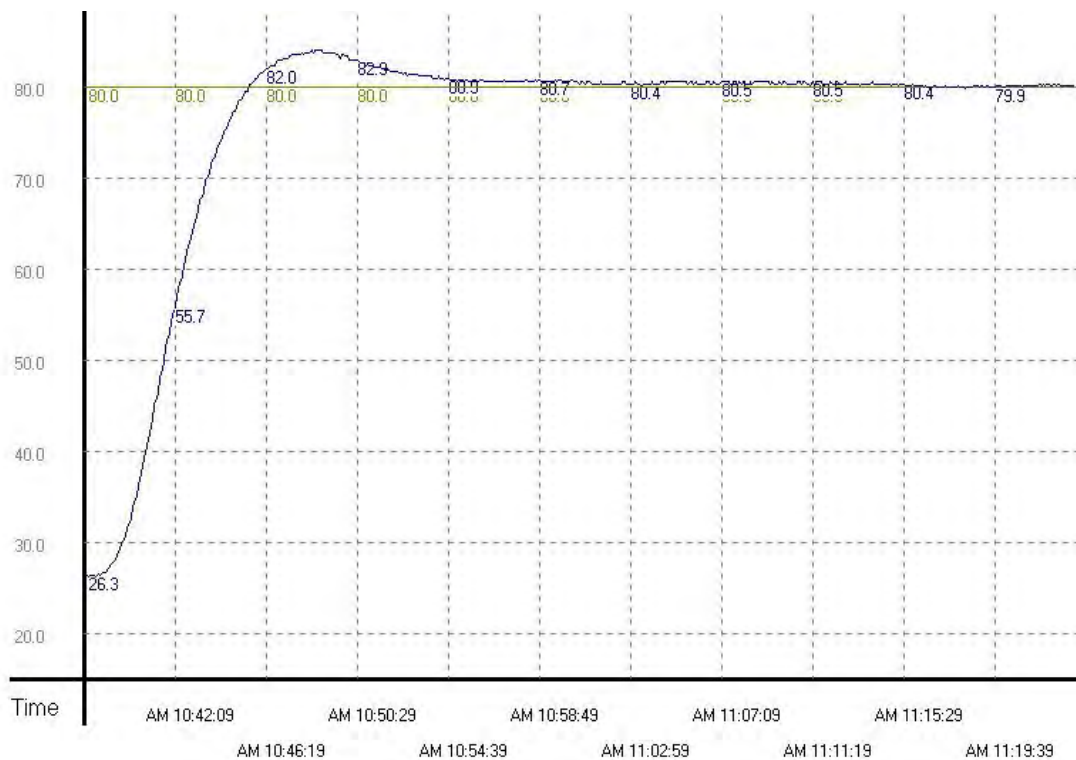


3

Performance curve 1: Using the auto tuning function as the initial PID mode

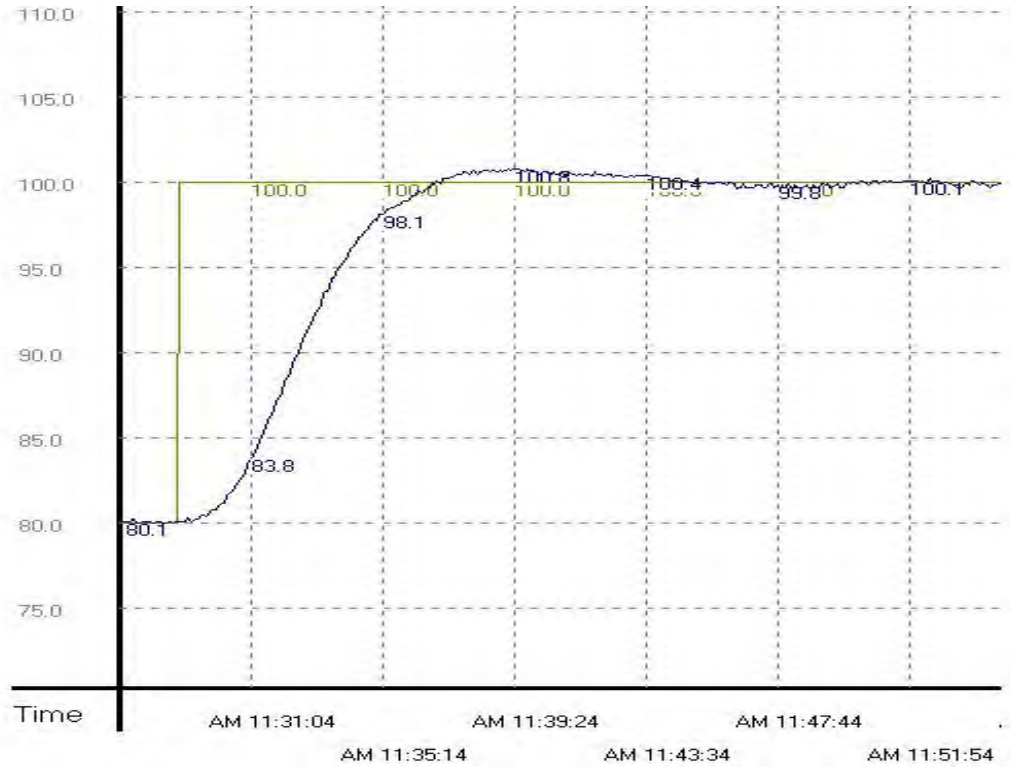


Performance curve 2: Using the parameters tuned by auto tuning function to control the temperature



As the diagram above shows, after the parameters are tuned automatically, you can get a good temperature control results. It only takes about twenty minutes to control the temperature. When the target temperature changes from 80°C to 100°C, the result is as below.

Performance curve 3: Changing target temperature from 80°C to 100°C



As the diagram above shows, the parameters tuned previously still can be used to control the temperature within similar time interval.

FB/FC	Instruction			Operand						Description				
FC	D*	PIDE		Refer to below table						PID algorithm				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
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										●*				
BIAS										●*				
I_MV										●*				
MV										●*				

装置	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
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								●	●				●				●

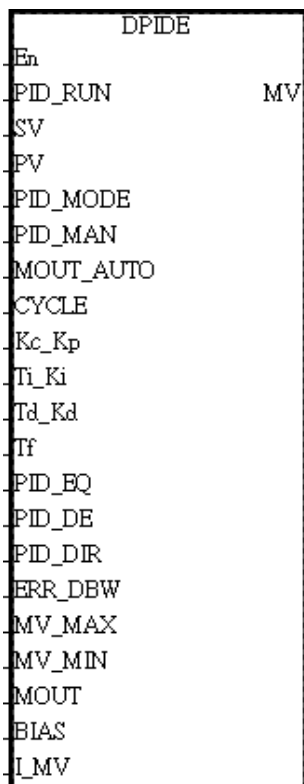
MV_MAX								●	●					●				●
MV_MIN								●	●					●				●
MOUT								●	●					●				
BIAS								●	●					●				●
I_MV								●	●					●				
MV								●	●					●				

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AH Motion CPU

Graphic expression:

- PID_RUN** : Enabling the PID algorithm
- SV** : Target value (SV)
- PV** : Present value (PV)
- PID_MODE** : PID control mode
- PID_MAN** : PID automatic/manual mode
- MOUT_AUTO** : MOUT automatic change mode
- CYCLE** : Sampling time (millisecond)
- Kc_Kp** : Proportional gain
- Ti_Ki** : Integral gain (second or 1/second)
- Td_Kd** : Derivative gain (second)
- Tf** : Derivative action time constant (second)
- PID_EQ** : Selection of a PID formula
- PID_DE** : Selection of the calculation of the PID derivative error
- PID_DIR** : PID forward/reverse direction
- ERR_DBW** : Error dead bandwidth
- MV_MAX** : Maximum output value
- MV_MIN** : Minimum output value
- MOUT** : Manual output value
- BIAS** : Feedforward output value
- I_MV** : Accumulated integral value
- MV** : Output value

3



Explanation:

- The instruction is used to implement an advanced PID algorithm. The PID algorithm is implemented only when the instruction is executed. PID stands for Proportional, Integral, Derivative. PID control is widely applied to mechanical equipment, pneumatic equipment, and electronic equipment.

2. The setting of the parameters is described below.

Interface	Data type	Function	Setting range	Description
PID_RUN	BOOL	Enabling the PID algorithm		True: The PID algorithm is implemented. False: The output value (MV) is reset to 0, and the PID algorithm is not implemented.
SV	REAL	SV	Range of single-precision floating-point values	Target value
PV	REAL	PV	Range of single-precision floating-point values	Present value
PID_MODE	DWORD/ DINT	PID control mode		0: Automatic control When PID_MAN is turned from true to false, the output value (MV) then is involved in the automatic algorithm. 1: Parameters are tuned automatically. After the tuning of the parameters is complete, the value will be changed to 0 automatically, and the appropriate parameters Kc_Kp, Ti_Ki, Td_Kd, and Tf will be calculated.
PID_MAN	BOOL	PID A/M mode		True: Manual The MV is output according to the MOUT, but it is still in the range of the MV_MIN to the MV_MAX. When PID_MODE is set to 1, the setting is ineffective. False: Automatic The MV is output according to the PID algorithm, and the output value is in the range of MV_MIN to MV_MAX.
MOUT_AUTO	BOOL	MOUT automatic change mode		True: Automatic The MOUT varies with the MV. False: Normal The MOUT does not vary with the MV.

Interface	Data type	Function	Setting range	Description
CYCLE	DWORD/ DINT	Sampling time (T_s)	1~40,000 (unit: ms)	When the instruction is scanned, the PID algorithm is implemented according to the sampling time, and the MV is refreshed. (The CPU module does not automatically judge time and execute the instruction according to the sampling time.) If CYCLE is less than 1, it will be count as 1. If CYCLE is greater than 40,000, it will be count as 40,000. When the instruction is used in the time interrupt, the CPU module automatically implements the PID algorithm according to the interval between the timed interrupts, and the setting of CYCLE does not work.
Kc_Kp	REAL	Proportional gain (K_c or K_p) (The selection of K_c or K_p depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values	It is a proportional gain. If a proportional gain is less than 0, Kc_Kp will be count as 0. If Kc_Kp is equal to 0 when the independent formula is used, the proportional control is not used.
Ti_Ki	REAL	Integral gain (T_i or K_i) (The selection of T_i or K_i depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values (Unit: T_i =Second; K_i =1/second)	It is an integral gain. If an integral gain is less than 0, Ti_Ki will be count as 0. If Ti_Ki is equal to 0, the integral control is not used.
Td_Kd	REAL	Derivative gain (T_d or K_d) (The selection of T_d or K_d depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values (Unit: Second)	It is a derivative gain. If a derivative gain is less than 0, Td_Kd will be count as 0. If Td_Kd is equal to 0, the derivative control is not used.

3

Interface	Data type	Function	Setting range	Description
Tf	REAL	Derivative action time constant (T _i)	Range of positive single-precision floating-point values (Unit: Second)	It the derivative action time constant. If the derivative action time constant is less than 0, Tf will be count as 0. If Tf is equal to 0, the derivative action time control is not used. (Derivative smoothing)
PID_EQ	BOOL	Selection of a PID formula	True: Dependent formula False: Independent Formula	
PID_DE	BOOL	Selection of the calculation of the PID derivative error	True: Using the variations in the PV to calculate the control value of the derivative (Derivative of the PV) False: Using 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	Error dead bandwidth: Range within which an error (E) is count as 0	Range of single-precision floating-point values	An error (E) is equal to SV-PV or PV-SV. If the setting value is 0, the function will not be enabled, otherwise the CPU module will check whether the present error is less than the absolute value of ERR_DBW, and check whether the present error meets the cross status condition. If the present error is less than the absolute value of ERR_DBW, and meets the cross status condition, the present error will be count as 0, and the PID algorithm will be implemented, otherwise the present error will be brought into the PID algorithm according to the normal processing.
MV_MAX	REAL	Maximum output value	Range of single-precision floating-point values	Example: After MV_MAX is set to 1,000, an MV will be 1,000 if it exceeds 1,000. MV_MAX has to be greater than MV_MIN , otherwise the

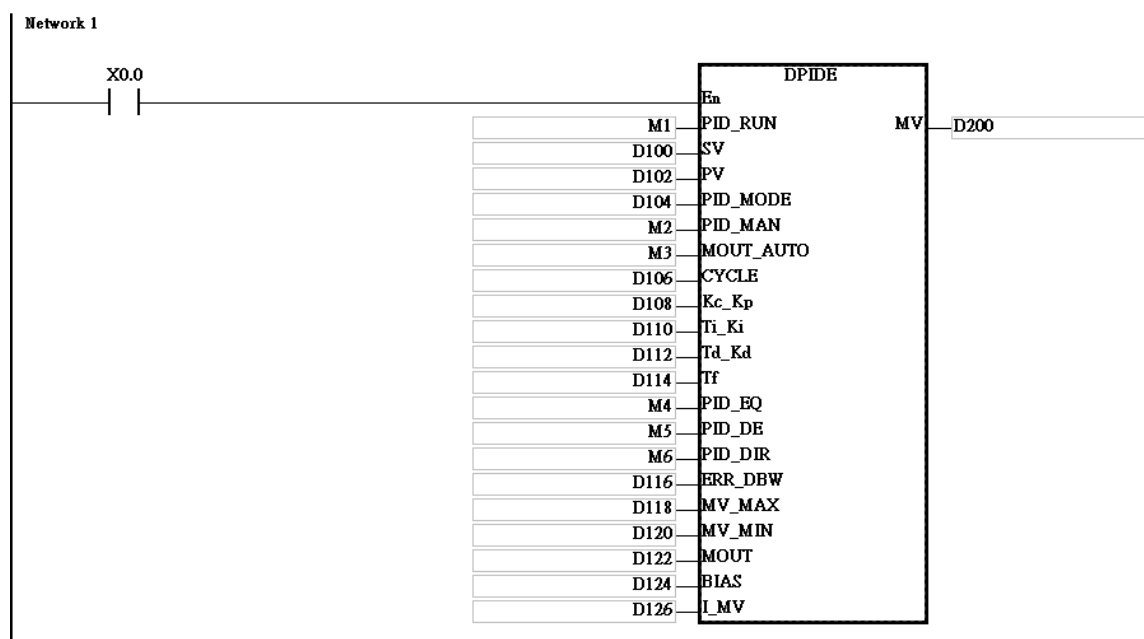
Interface	Data type	Function		Setting range	Description	
					maximum output value set and the minimum output value set will be interchanged.	
MV_MIN	REAL	Minimum output value		Range of single-precision floating-point values	Example: After MV_MIN is set to -1,000, an MV will be -1,000 if it is less than -1,000. MV_MIN has to be less than MV_MAX , otherwise the maximum output value set and the minimum output value set will be interchanged.	
MOUT	REAL	Manual output value		Range of single-precision floating-point values	Mout and PID_MAN are used together. If PID_MAN is set to true, the MV will be output according to the MOUT, but it will be still in the range of the MV_MIN to the MV_MAX.	
BIAS	REAL	Feedforward output value		Range of single-precision floating-point values	It is used for the PID feedforward.	
I_MV (It occupies ten consecutive 32-bit devices.)	REAL	I_MV	Accumulated integral value temporarily stored	Range of single-precision floating-point values	An accumulated integral value is usually for reference. Users can still clear or modify it according to their 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	Previous error temporarily stored	The system records the previous error.		
		I_MV+2~I_MV+5	For system use only			
		I_MV+6	The system records the previous PV.			
		I_MV+7~	For system use only			

3

Interface	Data type	Function	Setting range	Description
		I_MV+9		
MV	REAL	MV		The MV is in the range of the MV_MIN to the MV_MAX.

Example:

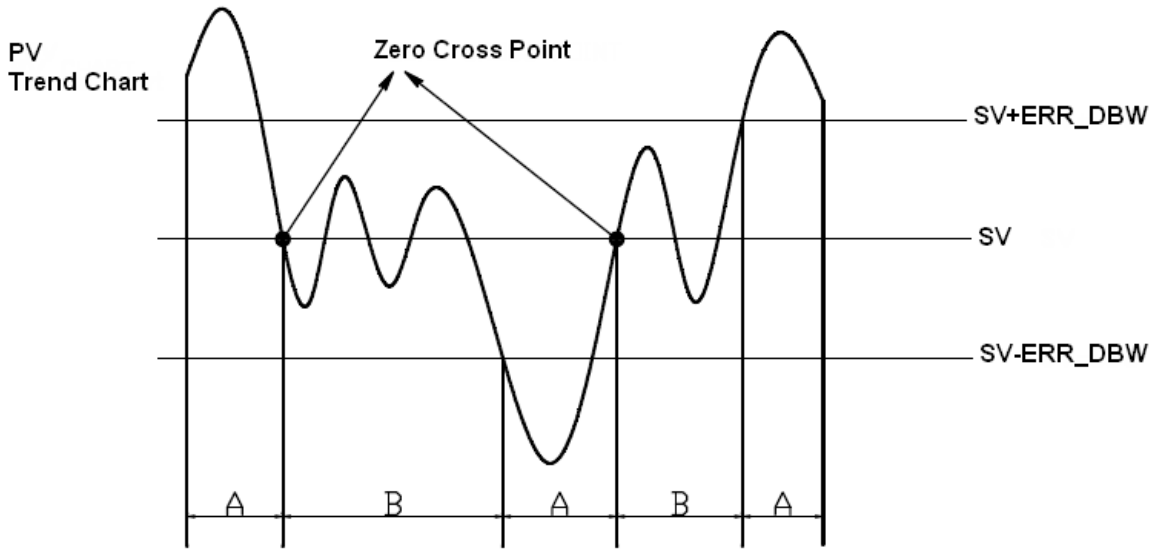
1. Before the instruction DPIDE is executed, the setting of the parameters should be complete.
2. When X0.0 is ON, the instruction is executed. When M1 is ON, the DPIDE algorithm is implemented. When M1 is OFF, the MV is 0, and the MV is stored in D200. When X0.0 is switched OFF, the instruction is not executed, and the previous data is unchanged.

**Additional remark:**

1. The instruction can be used several times, but the registers specified by I_MV~I_MV+9 can not be the same.
2. I_MV occupies 20 word registers. I_MV used in the instruction DPIDE in the example above occupies D126~D145.
3. The instruction DPIDE can only be used in the cyclic task and the time interrupt. When the instruction DPIDE is used in the time interrupt, the sampling time is the same as the interval between the time interrupts.
4. When the instruction DPIDE is scanned, the PID algorithm is implemented according to the sampling time, and the MV is directly refreshed. Whether the scan time reaches the sampling time is not calculated automatically. When the instruction is used in the time interrupt, the sampling time is the same as the interval between the time interrupts. The PID algorithm is implemented according to the interval between the time interrupts.
5. Before the PID algorithm is implemented, the present value used in the instruction DPIDE has to be a stable value. When users need the input value in the module to implement the PID algorithm, they have to notice the time it takes for the analog input to be converted into the digital input.
6. If the PV is in the range indicated by ERR_DBW, the CPU module will bring the error into the PID algorithm until the PV reaches the SV. The cross status condition will not be met until the PV crosses the zero cross point indicated by the SV. If the cross status condition is met, the error will be count as 0 until the PV is out of the range indicated by ERR_DBW. If PID_DE is set to true, the variations in the PV will be used to calculate the control value of the derivative, and the CPU module will count the Delta PV as 0 after the cross status

condition is met. (Delta PV=Current PV-Previous PV)

7. In the PV trend chart shown below, the CPU module implements the PID algorithm normally in the A sections A. In the B sections, the CPU module counts the error or the Delta PV as 0 when it implements the PID algorithm.



PID algorithms:

1. When **PID_MODE** is set to 0, the PID control mode is the automatic control mode.
- **Independent formula & derivative of of the E (PID_EQ=False & PID_DE=False)**

$$CV = K_p E + K_i \int_0^t E dt + K_d \frac{dE}{dt} + BIAS$$

$$E = SV - PV \quad \text{or} \quad E = PV - SV$$

- **Independent formula & derivative of the PV (PID_EQ=False & PID_DE=Ture)**

$$CV = K_p E + K_i \int_0^t E dt - K_d \frac{dPV}{dt} + BIAS$$

$$E = SV - PV$$

or

$$CV = K_p E + K_i \int_0^t E dt + K_d \frac{dPV}{dt} + BIAS$$

$$E = PV - SV$$

- **Dependent formula & derivative of the E (PID_EQ=True & PID_DE=False)**

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t E dt + T_d \frac{dE}{dt} \right] + BIAS$$

$$E = SV - PV \quad \text{or} \quad E = PV - SV$$

- **Dependent formula & derivative of the PV (PID_EQ=True & PID_DE=True)**

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t E dt - T_d \frac{dPV}{dt} \right] + BIAS$$

$$E = SV - PV$$

or

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t E dt + T_d \frac{dPV}{dt} \right] + BIAS$$

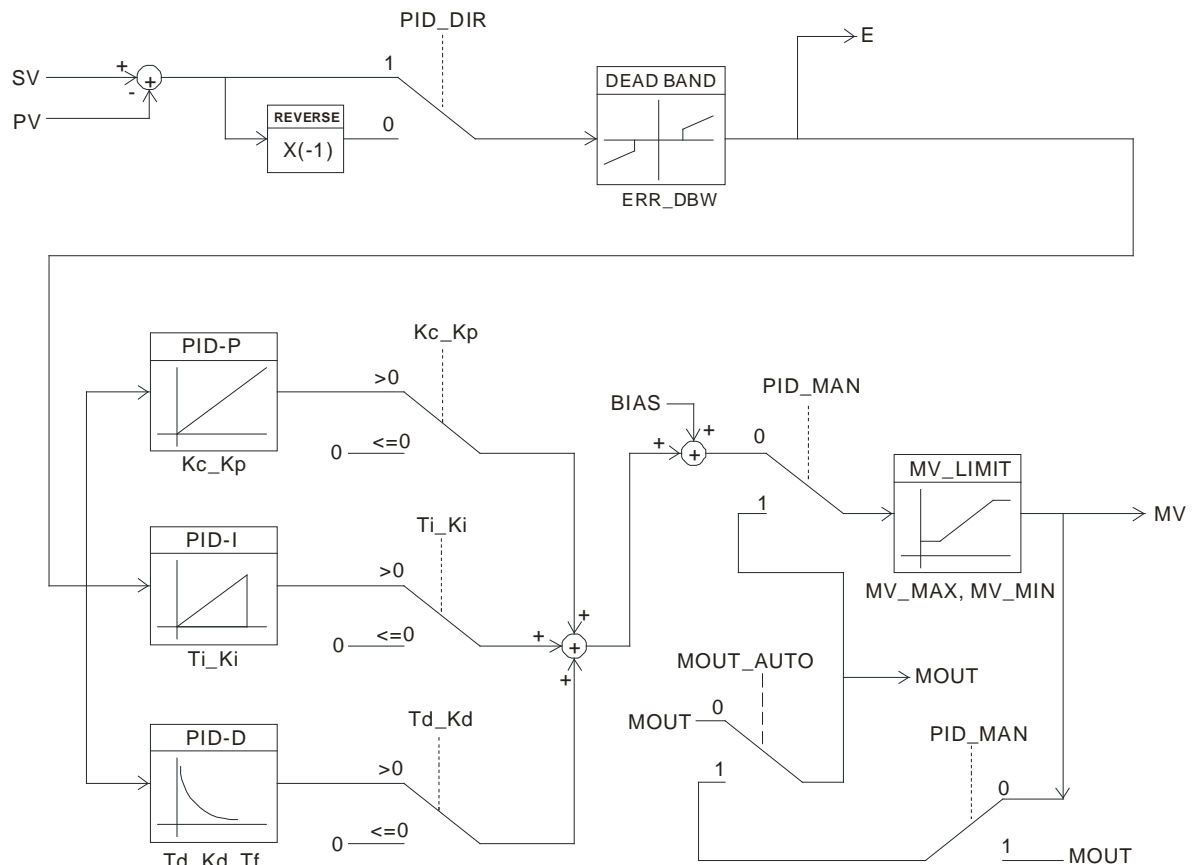
$$E = PV - SV$$

※The CV values in the formulas above are the MV used in DPIDE.

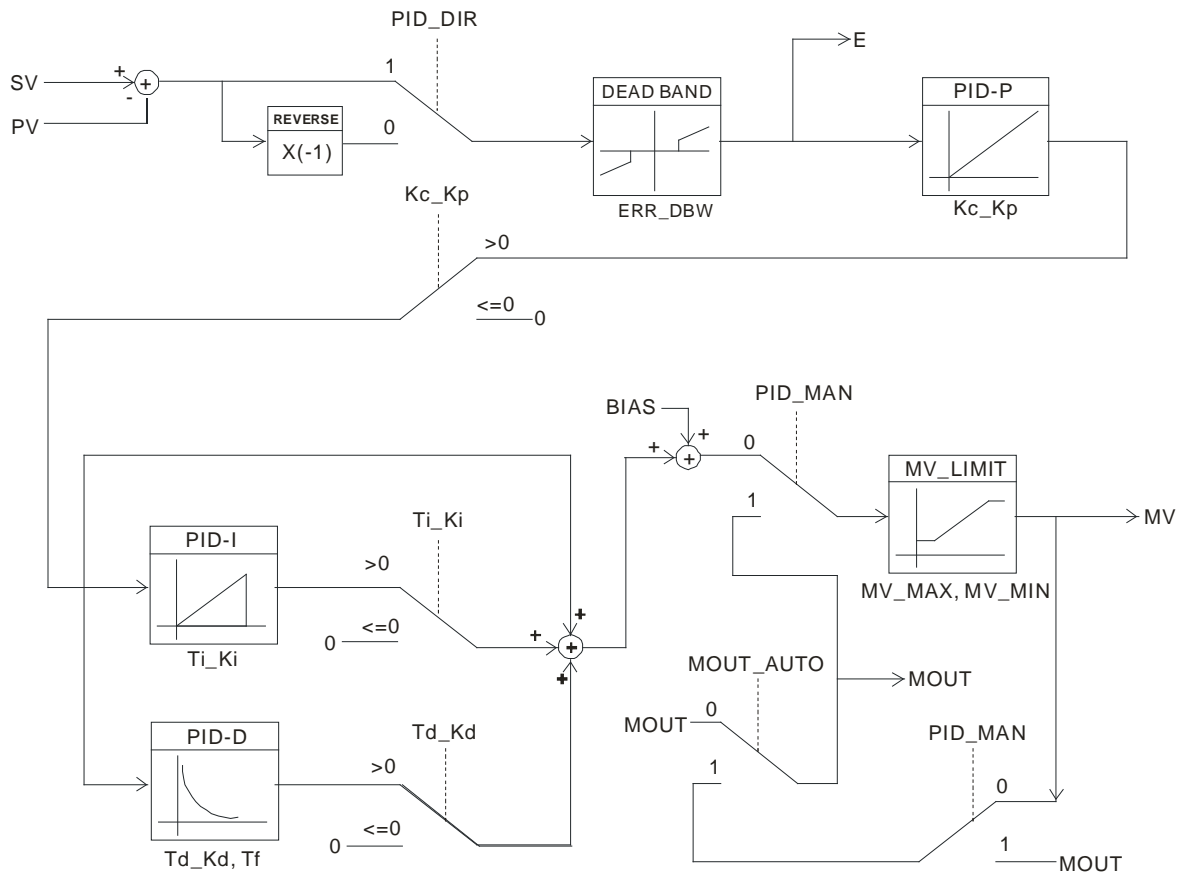
2. When **PID_MODE** is set to 1, the PID control mode is the automatic tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode becomes the automatic control mode.

PID control diagrams:

PID Block Diagram (Independent)



PID Block Diagram (Dependent)



Suggestion:

- Owing to the fact that the instruction DPIDE can be used in a lot of controlled environments, users have to select control functions appropriately. For example, the MV switches between the maximum output value and the minimum output value when **PID_MODE** is set to 1. Please do not use DPIDE in the environment controlled by a motor which reacts rapidly, otherwise the violent change of the system resulting from the automatic tuning of the parameters may hurt the staff or damage the system.
- When users tune the parameters $K_c K_p$, $T_i K_i$, and $T_d K_d$ (**PID_MODE** is set to 0), they have to tune $K_c K_p$ first (according to their experiences), and then set $T_i K_i$ and $T_d K_d$ to 0. When the users can handle the control, they can increase $T_i K_i$ and $T_d K_d$. When $K_c K_p$ is 1, it means that the proportional gain is 100%. That is, the error is increased by a factor of one. When the proportional gain is less than 100%, the error is decreased. When the proportional gain is greater than 100%, the error is increased.
- To prevent the parameters which have been tuned automatically from disappearing after a power cut, it is suggested that users should store the parameters in latching data registers if **PID_MODE** is set to 1. The parameters which have been tuned automatically are not necessarily suitable for every controlled environment. Therefore, the users can modify the parameters which have been tuned automatically. However, it is suggested that users only modify $T_i K_i$ or $T_d K_d$.
- The action of the instruction depends on many parameters. To prevent improper control from occurring, please do not set parameters randomly.

3.11 Logic Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>WAND</u>	<u>DAND</u>	✓	Logical AND operation	7
FC	<u>MAND</u>	–	✓	Matrix AND operation	9
FC	<u>WOR</u>	<u>DOR</u>	✓	Logical OR operation	7
FC	<u>MOR</u>	–	✓	Matrix OR operation	9
FC	<u>WXOR</u>	<u>DXOR</u>	✓	Logical exclusive OR operation	7
FC	<u>MXOR</u>	–	✓	Matrix exclusive OR operation	9
FC	<u>WXNR</u>	<u>DXNR</u>	✓	Logical exclusive NOR operation	7
FC	<u>MXNR</u>	–	✓	Matrix exclusive NOR operation	9
FC	<u>LD&</u>	<u>DLD&</u>	–	S1&S2	5
FC	<u>LD </u>	<u>DLD </u>	–	S1 S2	5
FC	<u>LD^</u>	<u>DLD^</u>	–	S1^S2	5
FC	<u>AND&</u>	<u>DAND&</u>	–	S1&S2	5
FC	<u>AND </u>	<u>DAND </u>	–	S1 S2	5
FC	<u>AND^</u>	<u>DAND^</u>	–	S1^S2	5
FC	<u>OR&</u>	<u>DOR&</u>	–	S1&S2	5
FC	<u>OR </u>	<u>DOR </u>	–	S1 S2	5
FC	<u>OR^</u>	<u>DOR^</u>	–	S1^S2	5

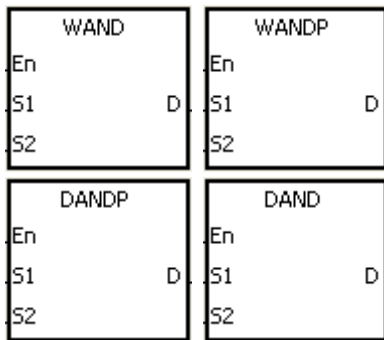
FB/FC	Instruction			Operand	Description
FC	W D*	AND	P	S ₁ , S ₂ , D	Logical AND operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●	●*				●	●*						
S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●	●	●	●		●	○	●	○	○		
S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

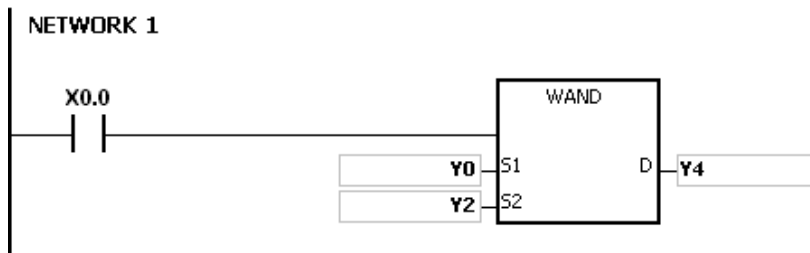
D : Operation result

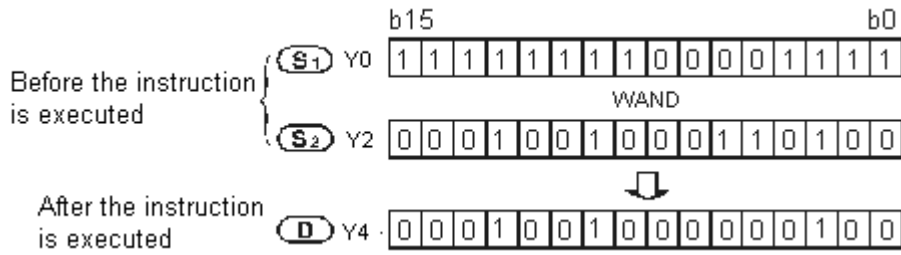
Explanation:

1. The logical operator AND takes the binary representations in S₁ and S₂, and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in D.
2. Only the instruction DAND 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:

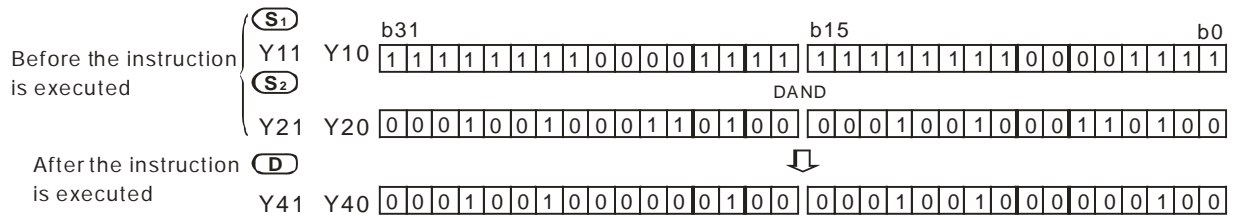
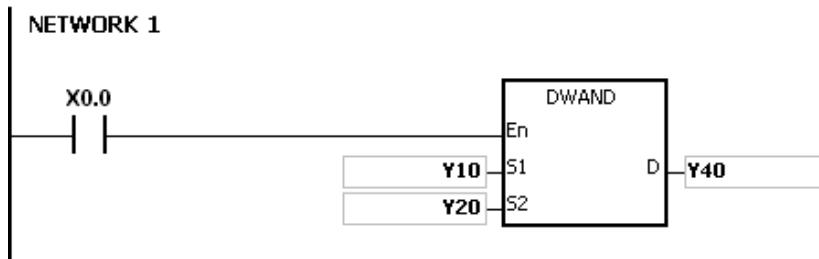
When X0.0 is ON, the logical operator AND takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in Y4.





Example 2:

When X0.0 is ON, the logical operator AND takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical AND operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).



FB/FC	Instruction		Operand		Description
FC	MAND	P	S₁, S₂, D, N		Matrix AND operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁, S₂	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●				●				
N	●	●			●	●	●	●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



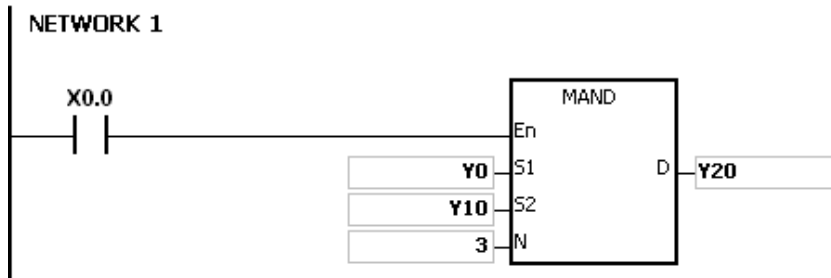
- S₁** : Matrix source 1
- S₂** : Matrix source 2
- D** : Operation result
- N** : Length of the array

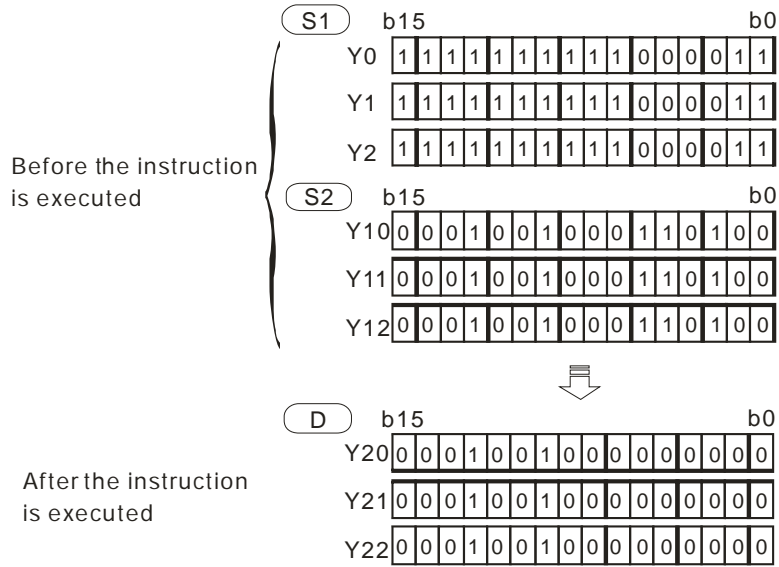
Explanation:

- The operator AND takes the n rows of binary representations in S1 and the n rows of binary representations in S2, and performs the matrix AND operation on each pair of corresponding bits. The operation result is stored in D.
- The result in each position is 1 if the first bit is 1 and the second bit is 1. Otherwise, the result is 0.
- The operand n should be within the range between 1 and 256.

Example:

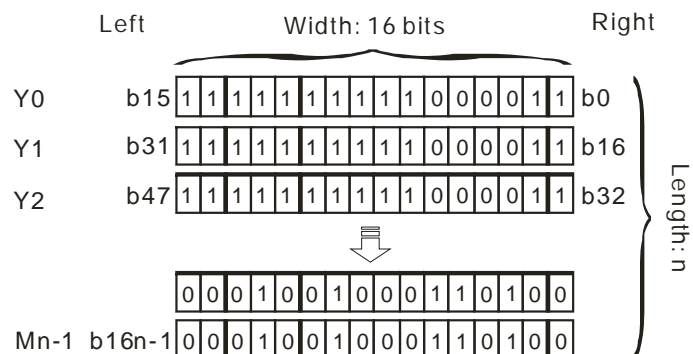
When X0.0 is ON, the operator AND takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix AND operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.





Additional remark:

1. If S1+n-1, S2+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 less than 1, or if n is larger than 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 16xn bits in a matrix, and the matrix operation is performed on one bit at a time.
 - The matrix instruction takes the 16xn bits in a matrix as a string of bits, rather than takes them as values. The matrix operation is performed on one specified bit.
 - The matrix instruction mainly processes the one-to-many status or the many-to-many status, such as the moving, the copying, the comparing, and the searching. It is a handy and important applied instruction.
 - When the matrix instruction is executed, you need a 16-bit register to specify a certain bit among the 16n bits in the matrix for the operation. The 16-bit register is called the pointer, and is specified by users. The value in the register is within the range between 0 and 16n-1, and corresponds to the bit within the range between b0 and b16n-1.
 - The shift of the specified data, or the rotation of the specified data can be involved in the matrix operation. Besides, the bit number decreases from the left to the right, as illustrated below.



- The width of the matrix (C) is 16 bits.
- Pr represents the pointer. When the value in Pr is 15, b15 is specified.

Example: The following matrix is composed of the three 16-bit devices Y0, Y1, and Y2. The data in Y0 is 16#AAAA, the data in Y1 is 16#5555, and the data in Y2 is 16#AAFF.

C ₁₅	C ₁₄	C ₁₃	C ₁₂	C ₁₁	C ₁₀	C ₉	C ₈	C ₇	C ₆	C ₅	C ₄	C ₃	C ₂	C ₁	C ₀	
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	Y0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	Y1
1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	Y2

Example: The following matrix is composed of the three 16-bit devices X 0, X 1, and X 2. The data in X 0 is 16#37, the data in X 1 is 16#68, and the data in X 2 is 16#45.

C ₁₅	C ₁₄	C ₁₃	C ₁₂	C ₁₁	C ₁₀	C ₉	C ₈	C ₇	C ₆	C ₅	C ₄	C ₃	C ₂	C ₁	C ₀	
0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	X0
0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	X1
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	X2

3

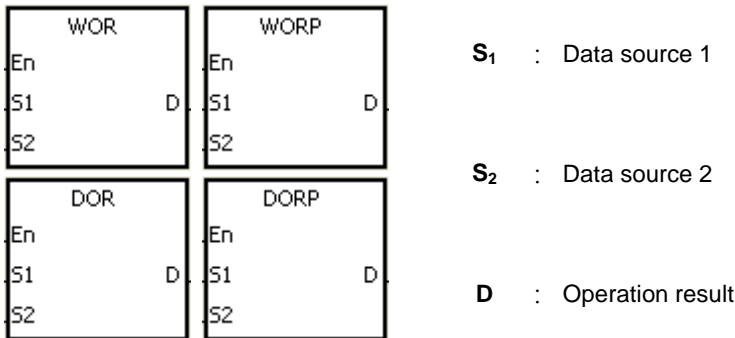
FB/FC	Instruction			Operand	Description
FC	W D*	OR	P	S ₁ , S ₂ , D	Logical OR operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

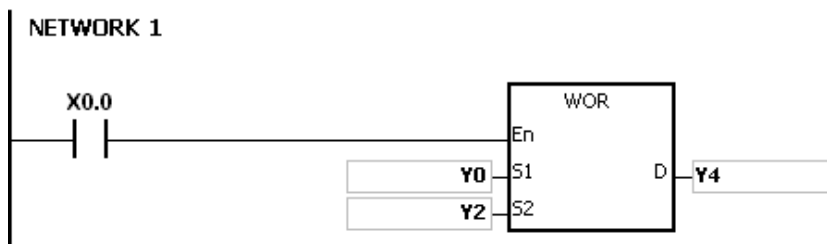


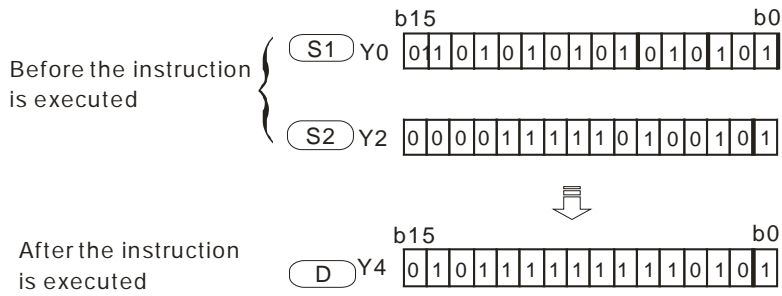
Explanation:

1. The logical operator OR takes the binary representations in **S₁** and **S₂**, and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in **D**.
2. Only the instruction DOR can use the 32-bit counter.
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:

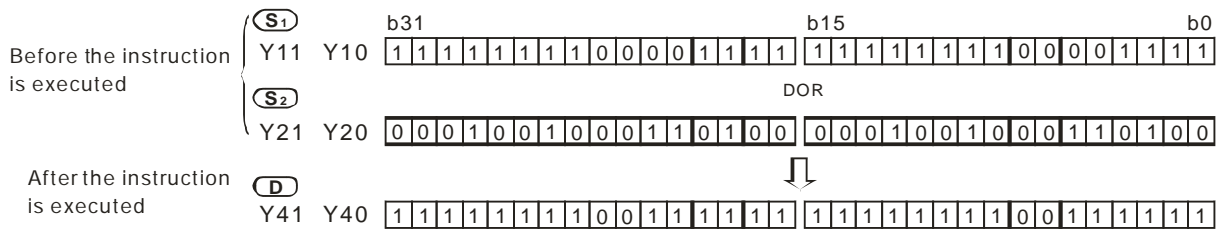
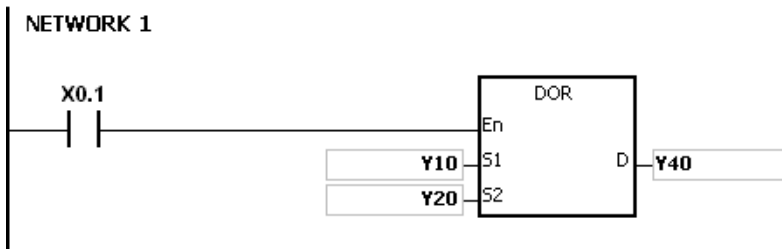
When X0.0 is ON, the logical operator OR takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in Y4.





Example 2:

When X0.1 is ON, the logical operator OR takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical inclusive OR operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).



3

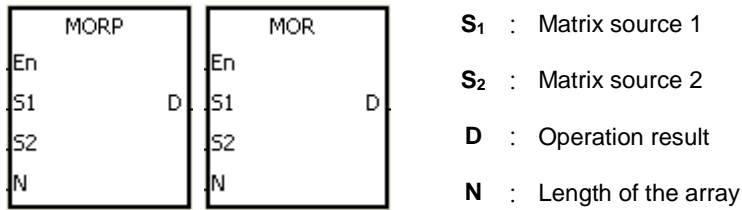
FB/FC	Instruction		Operand	Description
FC	MOR	P	S₁, S₂, D, N	Matrix OR operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●				●				
N	●	●			●	●	●	●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

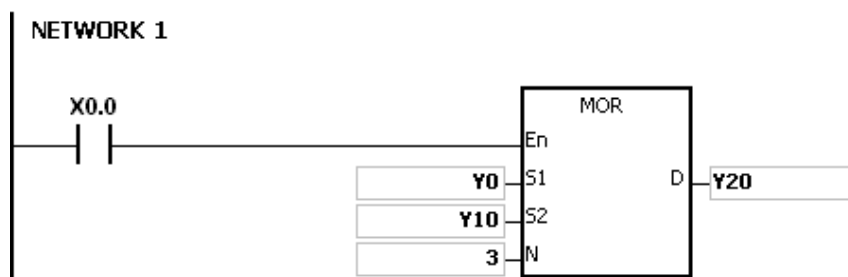


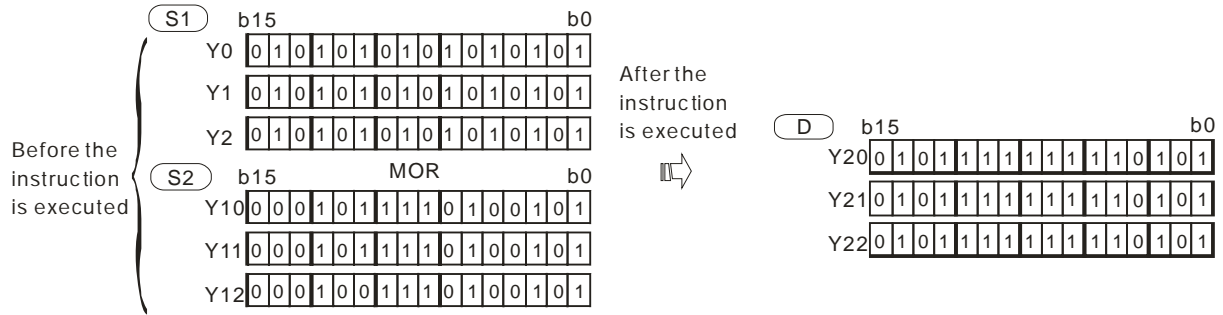
Explanation:

1. The operator OR takes the n rows of binary representations in S1 and the n rows of binary representations in S2, and performs the matrix OR operation on each pair of corresponding bits. The operation result is stored 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 should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator OR takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.





Additional remark:

1. If S1+n-1, S2+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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

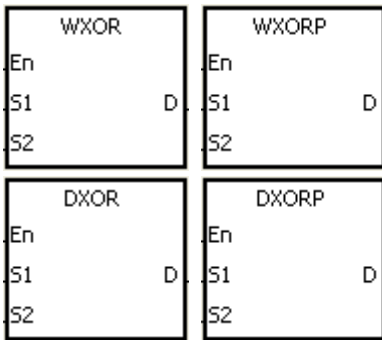
FB/FC	Instruction			Operand	Description
FC	W D*	XOR	P	S ₁ , S ₂ , D	Logical exclusive OR operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

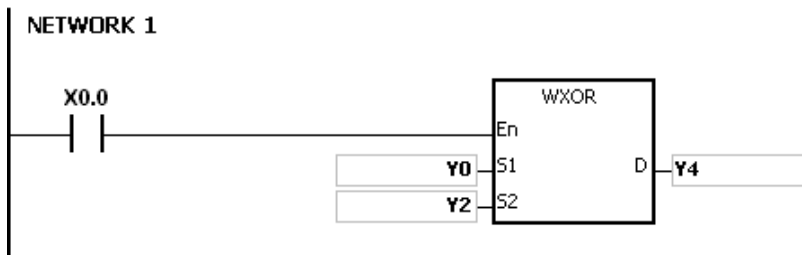
D : Operation result

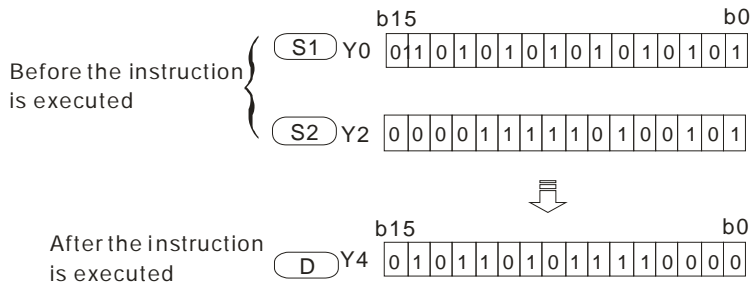
Explanation:

1. The logical operator XOR takes the binary representations in S₁ and S₂, and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in D.
2. Only the instruction DXOR can use the 32-bit counter.
3. The result in each position is 1 if the two bits are different, and 0 if they are the same.

Example 1:

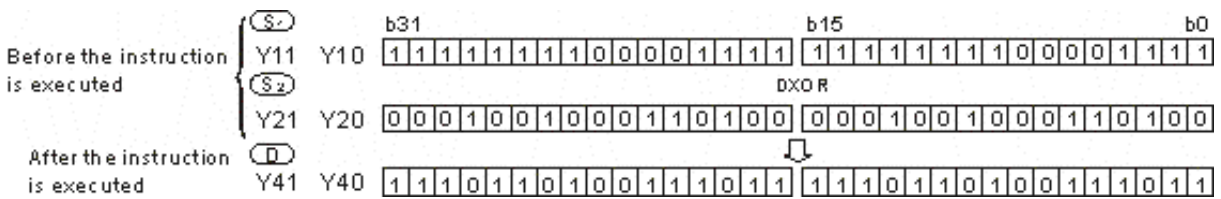
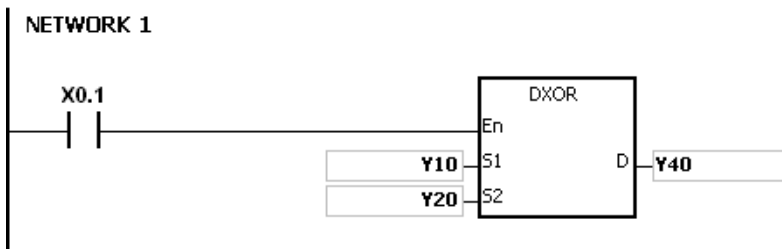
When X0.0 is ON, the logical operator XOR takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the exclusive OR operation on each pair of corresponding bits. The operation result is stored in Y4.





Example 2:

When X0.1 is ON, the logical operator XOR takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical exclusive OR operation on each pair of corresponding bits. The operation result is stored in (Y41, Y40).



3

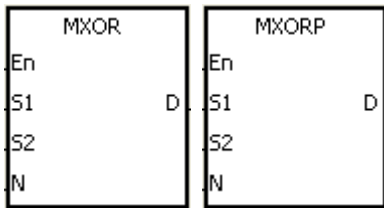
FB/FC	Instruction		Operand				Description					
FC		MXOR	P	S₁, S₂, D, N				Matrix exclusive OR operation				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●				●				
N	●	●			●	●	●	●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



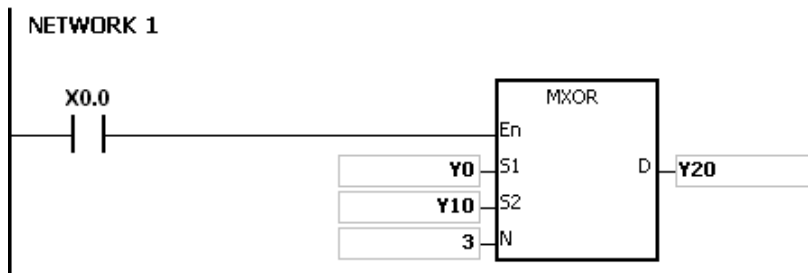
- S₁** : Matrix source 1
- S₂** : Matrix source 2
- D** : Operation result
- N** : Length of the array

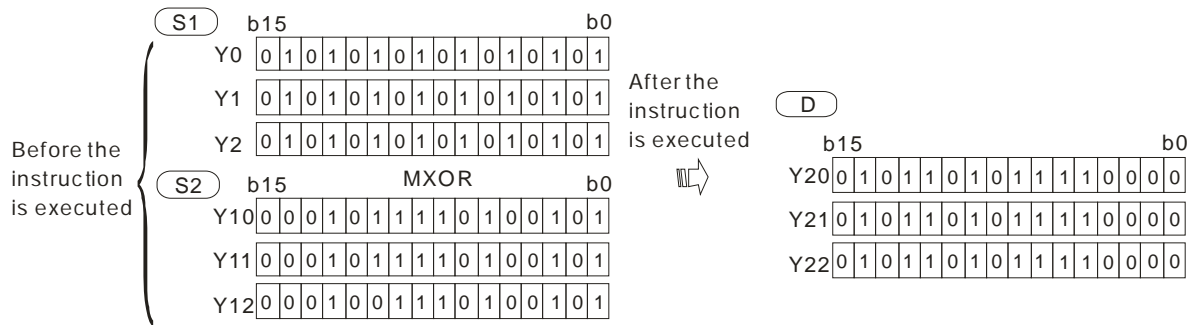
Explanation:

- The operator XOR takes the **n** rows of binary representations in **S₁** and the **n** rows of binary representations in of **S₂**, and performs the matrix exclusive OR operation on each pair of corresponding bits. The operation result is stored in **D**.
- The result in each position is 1 if the two bits are different, and 0 if they are the same.
- The operand **n** should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator XOR takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix exclusive OR operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.





Additional remark:

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 less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

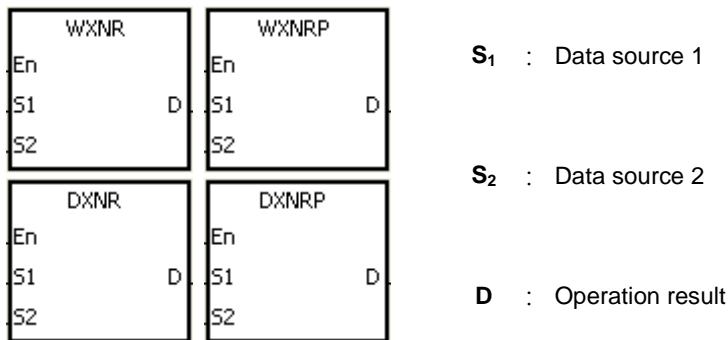
FB/FC	Instruction			Operand	Description
FC	W D*	XNR	P	S ₁ , S ₂ , D	Logical exclusive NOR operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

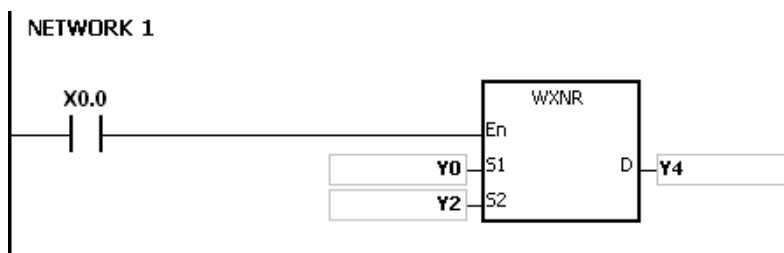


Explanation:

1. The logical operator XNR takes the binary representations in **S₁** and **S₂**, and performs the logical exclusive NOR operation on each pair of corresponding bits. The operation result is stored in **D**.
2. Only the instruction DXNR can use the 32-bit counter.
3. The result in each position is 1 if the two bits are the same, and 0 if they are different.

Example 1:

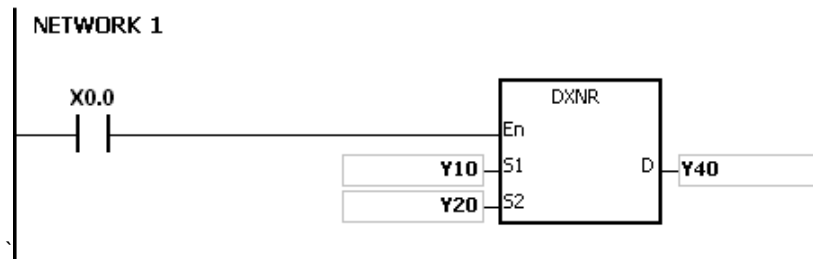
When X0.0 is ON, the logical operator XNR takes the data in the 16-bit device Y0 and the 16-bit device Y2, and performs the logical exclusive NOR operation on each pair of corresponding bits. The operation result is stored in Y4.



Example 2:

When X0.0 is ON, the logical operator XNR takes the data in the 32-bit device (Y11, Y10) and the 32-bit device (Y21, Y20), and performs the logical exclusive NOR operation on each pair of corresponding bits. The operation result is

stored in (Y41, Y40).



3

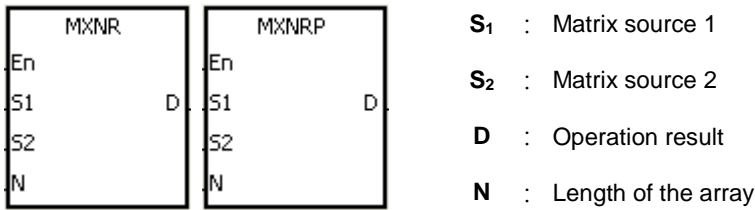
FB/FC	Instruction		Operand	Description
FC	MXNR	P	S₁, S₂, D, N	Matrix exclusive NOR operation

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							
N		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●				●				
N	●	●			●	●	●	●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:

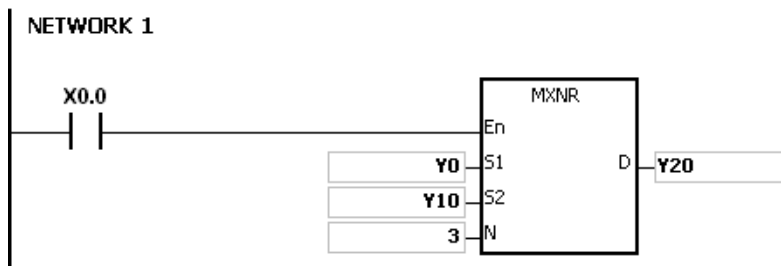


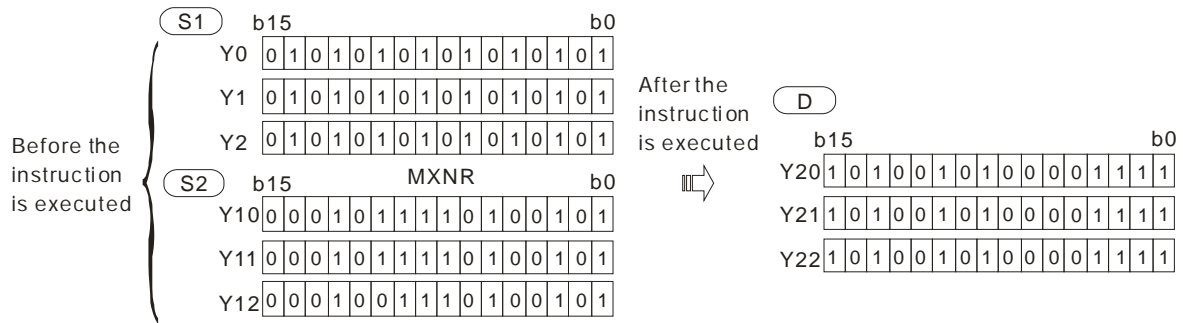
Explanation:

1. The operator XNR takes the n rows of binary representations in S1 and the n rows binary representations in of S2, and performs the matrix exclusive NOR operation on each pair of corresponding bits. The operation result is stored in D.
2. The result in each position is 1 if the two bits are the same, and 0 if they are different.
3. The operand n should be within the range between 1 and 256.

Example:

When X0.0 is ON, the operator XNR takes the data in the 16-bit devices Y0~Y2 and the data in 16-bit devices Y10~Y12, and performs the matrix exclusive NOR operation on each pair of corresponding bits. The operation result is stored in the 16-bit devices Y20~Y22.





Additional remark:

1. If S1+n-1, S2+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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

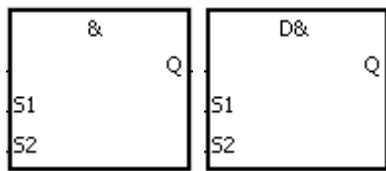
FB/FC	Instruction			Operand	Description
FC	D*	LD #		S ₁ , S ₂	Contact type of logical operation LD #

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●				●	●	●	●		●	○		○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking LD& and DLD& for example

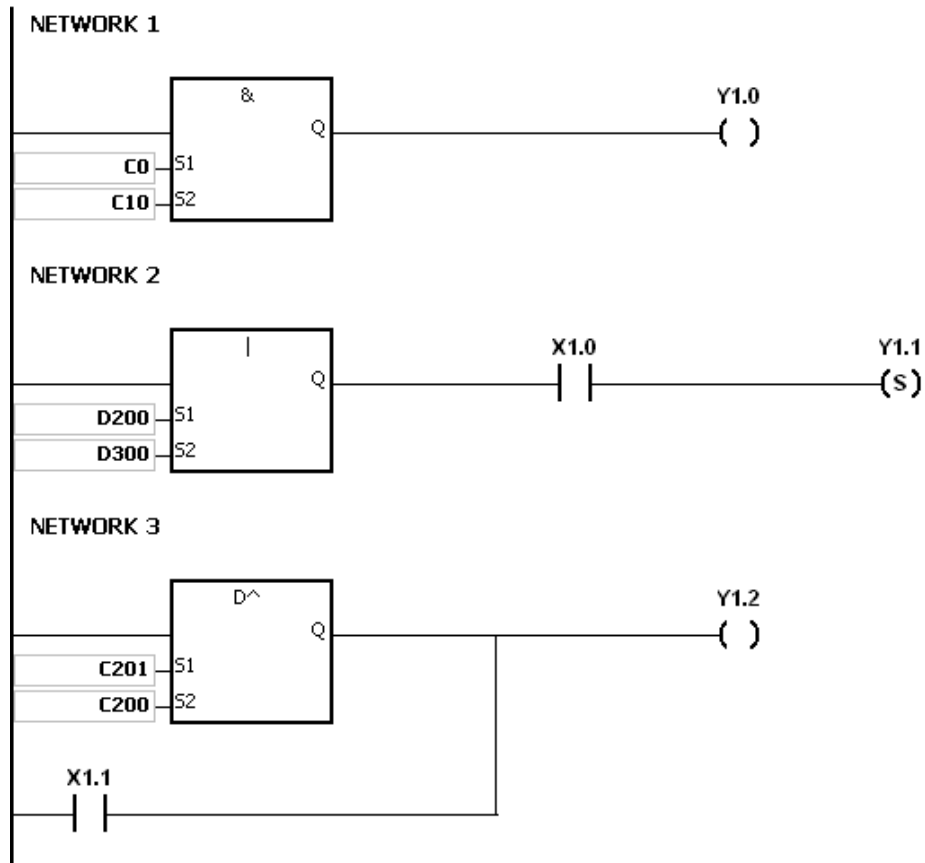
Explanation:

1. The instruction is used to compare the data in S₁ with that in S₂. 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.
3. The instruction LD # can be connected to the mother line directly.
4. &: Logical AND operation
5. |: Logical OR operation
6. ^: Logical exclusive OR operation

16-bit instruction	32-bit instruction	Comparison operation result	
		ON	OFF
LD&	DLD&	S ₁ &S ₂ ≠ 0	S ₁ &S ₂ = 0
LD	DLD	S ₁ S ₂ ≠ 0	S ₁ S ₂ = 0
LD^	DLD^	S ₁ ^S ₂ ≠ 0	S ₁ ^S ₂ = 0

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, Y1.0 is ON.
2. The logical operator OR takes the data in D200 and D300, and performs the logical OR operation on each pair of corresponding bits. When the operation result is not 0 and X1.0 is ON, Y1.1 is ON.
3. The logical operator XOR takes the data in C201 and C200, and performs the logical exclusive OR operation on each pair of corresponding bits. When the operation result is not 0, or when X1.1 is ON, Y1.2 is ON.



Additional remark:

If S₁ or S₂ is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

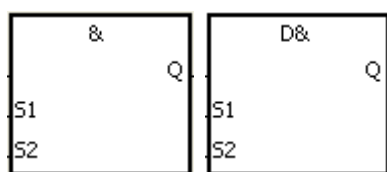
FB/FC	Instruction			Operand	Description
FC	D*	AND #		S ₁ , S ₂	Contact type of logical operation AND #

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●				●	●	●	●		●	○		○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking AND& and DAND& for example

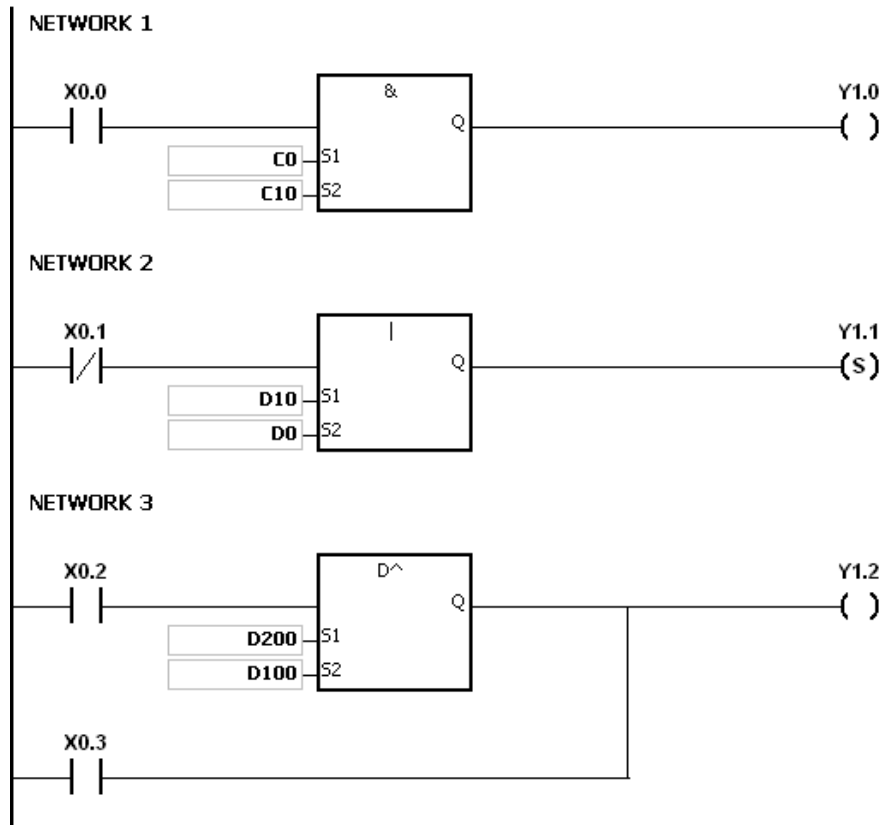
Explanation:

1. The instruction is used to compare the data in S₁ with that in S₂. 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 DAND# can use the 32-bit counter.
3. The instruction AND# and the contact are connected in series.
4. &: Logical AND operation
5. |: Logical OR operation
6. ^: Logical exclusive OR operation

16-bit instruction	32-bit instruction	Comparison operation result	
		ON	OFF
AND&	DAND&	S ₁ &S ₂ ≠ 0	S ₁ &S ₂ = 0
AND	DAND	S ₁ S ≠ 0	S ₁ S ₂ = 0
AND^	DAND^	S ₁ ^S ₂ ≠ 0	S ₁ ^S = 0

Example:

1. When X0.0 is ON, the logical operator AND takes the data in C0 and C10, and performs the logical AND operation on each pair of corresponding bits. When the operation result is not 0, Y1.0 is ON.
2. When X0.1 is OFF, the logical operator OR takes the data in D10 and D0, and performs the logical OR operation on each pair of corresponding bits. When the operation result is not 0, Y1.1 keeps ON.
3. When X0.2 is ON, the logical operator XOR takes the data in (D200, D201) and (D100, D101) and performs the logical exclusive OR operation on each pair of corresponding bits. When the operation result is not 0, or when X0.3 is ON, Y1.2 is ON.



Additional remark:

If **S₁** or **S₂** is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

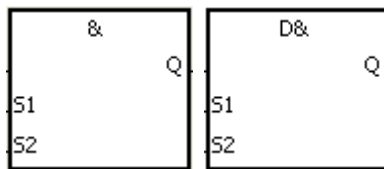
FB/FC	Instruction			Operand	Description
FC	D*	OR #		S ₁ , S ₂	Contact type of logical operation OR #

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●				●	●	●	●		●	○		○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Data source 1

S₂ : Data source 2

Taking OR& and DOR& for example

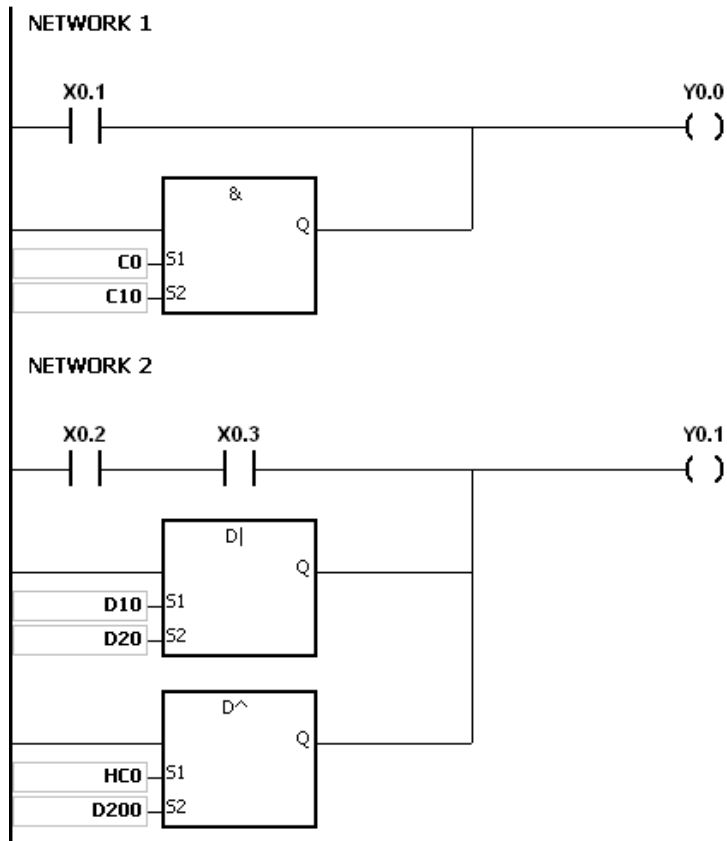
Explanation:

- The instruction is used to compare the data in S₁ with that in S₂. 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.
- Only the instruction DOR # can use the 32-bit counter.
- The instruction OR # and the contact are connected in parallel.
- &: Logical AND operation
- |: Logical OR operation
- ^: Logical exclusive OR operation

16-bit instruction	32-bit instruction	Comparison operation result	
		ON	OFF
OR&	DOR&	S ₁ &S ₂ ≠ 0	S ₁ &S ₂ = 0
OR	DOR	S ₁ S ₂ ≠ 0	S ₁ S ₂ = 0
OR^	DOR^	S ₁ ^S ₂ ≠ 0	S ₁ ^S ₂ = 0

Example:

- When X0.1 is ON, Y0.0 is ON. Besides, when the logical operator AND performs the logical AND operation on each pair of corresponding bits in C0 and C10 and the operation result is not 0, Y0.0 is ON.
- When X0.2 and X0.3 are ON, Y0.1 is ON. When the logical operator OR performs the logical OR operation on each pair of corresponding bits in the 32-bit register (D11, D10) and the 32-bit register (D21, D20) and the operation result is not 0, Y0.1 is ON. Besides, when the logical operator XOR performs the logical exclusive OR operation on each pair of corresponding bits in the 32-bit counter HC0 and the 32-bit register (D201, D200) and the operation result is not 0, Y0.1 is ON.



Additional remark:

If **S₁** or **S₂** is illegal, the condition of the instruction is not met, SM0 is ON, and the error in SR0 is 16#2003.

3.12 Rotation Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>ROR</u>	<u>DROR</u>	✓	Rotating to the right	5
FC	<u>RCR</u>	<u>DRCR</u>	✓	Rotating to the right with the carry flag	5
FC	<u>ROL</u>	<u>DROL</u>	✓	Rotating to the left	5
FC	<u>RCL</u>	<u>DRCL</u>	✓	Rotating to the left with the carry flag	5
FC	<u>MBR</u>	–	✓	Rotating the matrix bits	7

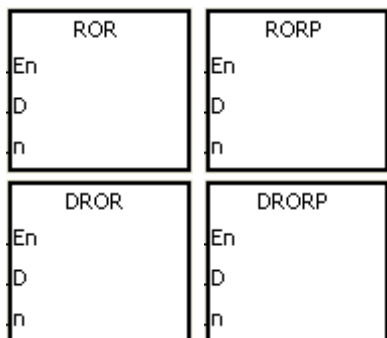
FB/FC	Instruction			Operand	Description
FC	D*	ROR	P	D, n	Rotating to the right

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



D : Device which is rotated

n : Number of bits forming a group

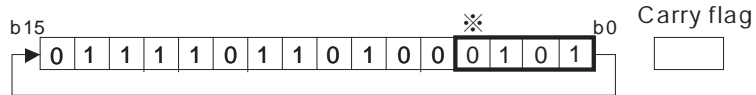
Explanation:

1. The values of the bits in the device specified by D are divided into groups (n bits as a group), and these groups are rotated to the right.
2. Only the instruction DROR can use the 32-bit counter.
3. The operand n used in the 16-bit instruction should be within the range between 1 and 16. The operand n used in the 32-bit instruction should be within the range between 1 and 32.
4. Generally, the pulse instructions RORP and DRORP are used.

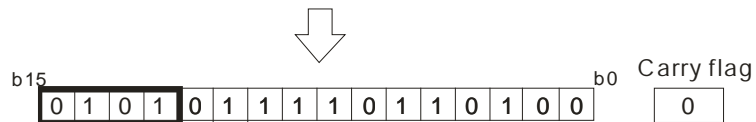
Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the right. (The value of the bit marked ※ is transmitted to the carry flag SM602.)





After the rotation is executed



Additional remark:

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 n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

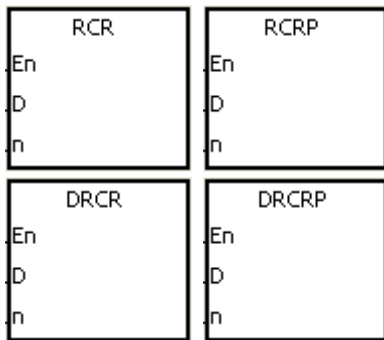
FB/FC	Instruction			Operand	Description
FC	D*	RCR	P	D, n	Rotating to the right with the carry flag

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3 Graphic expression:



D : Device which is rotated

n : Number of bits forming a group

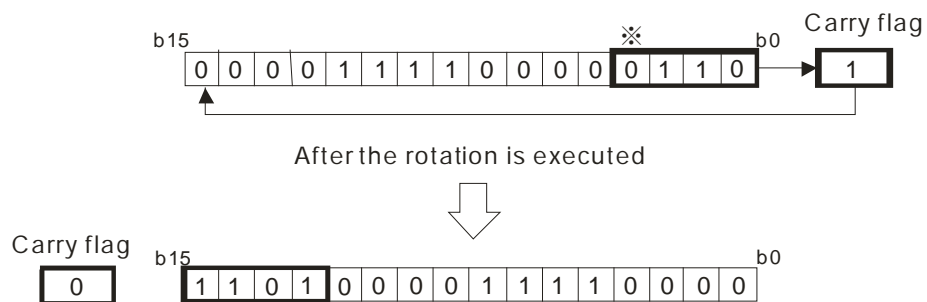
Explanation:

1. The values of the bits in the device specified by D are divided into groups (n bits as a group), and these groups are rotated to the right with the carry flag SM602.
2. Only the 32-bit instructions can use the 32-bit counter.
3. The operand n used in the 16-bit instruction should be within the range between 1 and 16. The operand n used in the 32-bit instruction should be within the range between 1 and 32.
4. Generally, the pulse instructions RCRP and DRCRP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the right with the carry flag SM602. (The value of the bit marked ※ is transmitted to the carry flag SM602.)





Additional remark:

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 n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	D*	ROL	P	D, n	Rotating to the left

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



D : Device which is rotated

n : Number of bits forming a group

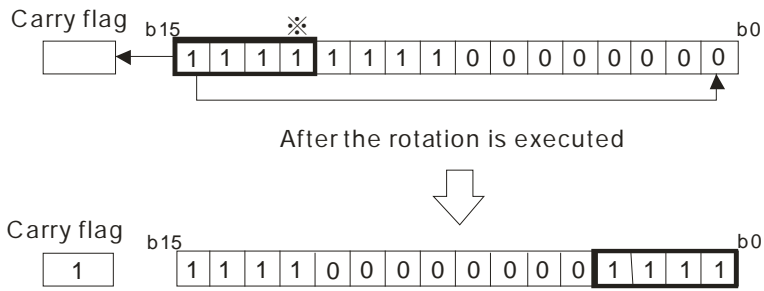
Explanation:

1. The values of the bits in the device specified by D are divided into groups (n bits as a group), and these groups are rotated to the left.
2. Only the 32-bit instructions can use the 32-bit counter.
3. The operand n used in the 16-bit instruction should be within the range between 1 and 16. The operand n used in the 32-bit instruction should be within the range between 1 and 32.
4. Generally, the pulse instructions ROLP and DROLP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the left. (The value of the bit marked ※ is transmitted to the carry flag SM602.)



**Additional remark:**

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 n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

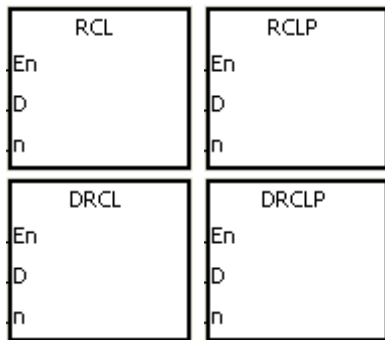
FB/FC	Instruction			Operand	Description
FC	D*	RCL	P	D, n	Rotating to the left with the carry flag

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○	●				
n	●	●						●	●		●		●	○	○		

Pulse instruction	32-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3 Graphic expression:



D : Device which is rotated

n : Number of bits forming a group

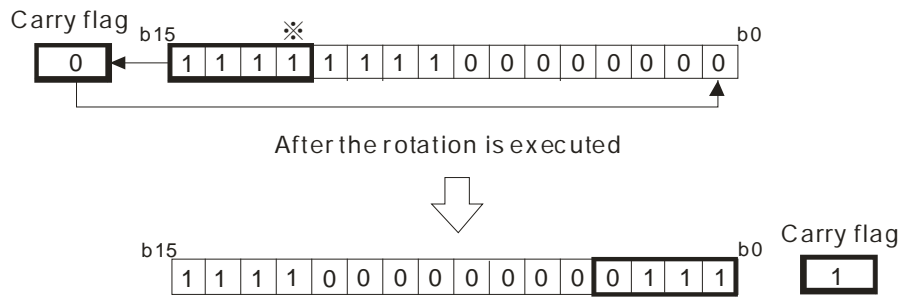
Explanation:

1. The values of the bits in the device specified by D are divided into groups (n bits as a group), and these groups are rotated to the left with the carry flag SM602.
2. Only the 32-bit instructions can use the 32-bit counter.
3. The operand n used in the 16-bit instruction should be within the range between 1 and 16. The operand n used in the 32-bit instruction should be within the range between 1 and 32.
4. Generally, the pulse instructions RCLP and DRCLP are used.

Example:

When X0.0 is switched from OFF to ON, the values of the bits in D10 are divided into groups (four bits as a group), and these groups are rotated to the left with the carry flag SM602. (The value of the bit marked ※ is transmitted to the carry flag SM602.)



**Additional remark:**

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 n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

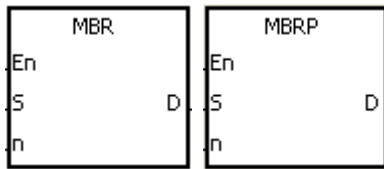
FB/FC	Instruction			Operand	Description
FC		MBR	P	S, D, n	Rotating the matrix bits

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
N	●	●			●	●	●	●	●		●		●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



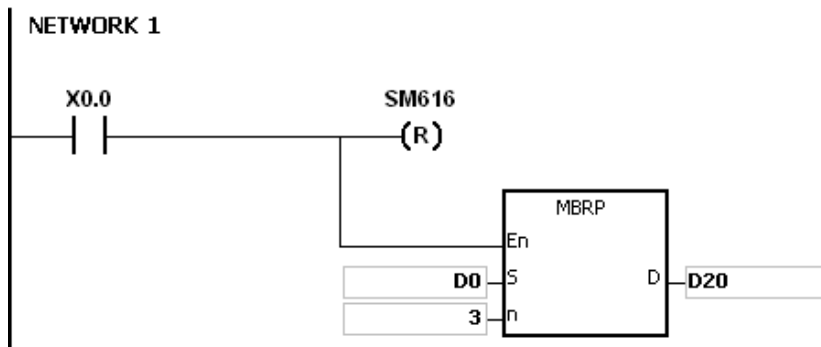
S : Matrix source
D : Operation result
n : Length of the array

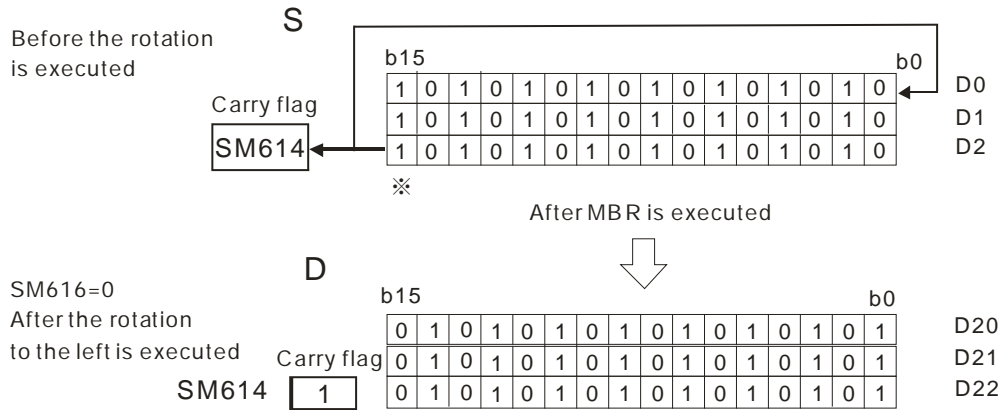
Explanation:

1. The values of the **n** rows of bits in **S** are rotated to the right or to the left. When SM616 is OFF, the values of the bits are rotated to the left. When SM616 is ON, the values of the bits are rotated to the right. The vacancy resulting from the rotation is filled by the value of the bit rotated last, and the operation result is stored in **D**. The value of the bit rotated last not only fills the vacancy, but also is transmitted to the carry flag SM614.
2. The operand **n** should be within the range between 1 and 256.
3. Generally, the pulse instruction MBRP is used.

Example 1:

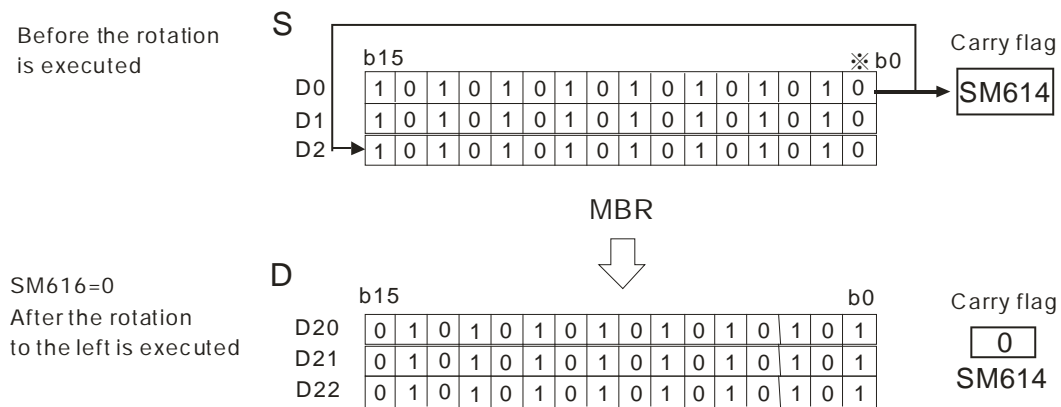
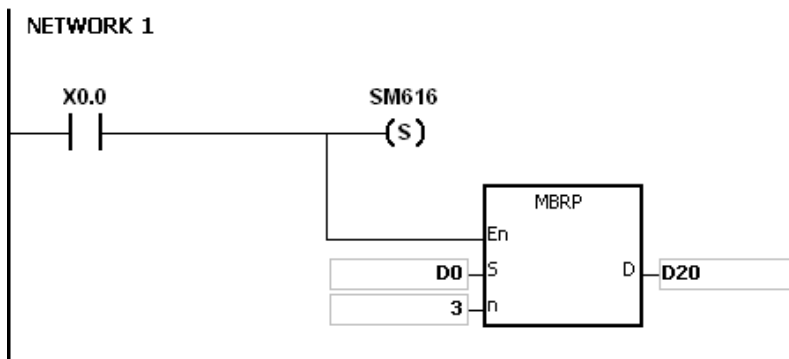
When X0.0 is ON, SM616 is OFF. The values of the bits in the 16-bit registers D0~D2 are rotated to the left, and the operation result is stored in the 16-bit registers D20~D22. The value of the bit marked ※ is transmitted to the carry flag SM614.





Example 2:

When X0.0 is ON, SM616 is ON. The values of the bits in the 16-bit registers D0~D2 are rotated to the right, and the operation result is stored in the 16-bit registers D20~D22. The value of the bit marked ※ is transmitted to the carry flag SM614.



Additional remark:

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 less than 1, or if **n** is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. The flags:
 - SM614: It is the carry flag for the matrix rotation/shift/output.
 - SM616: It is the direction flag for the matrix rotation/shift.

3.13 Basic Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>RST</u>	–	–	Resetting the contact or clearing the register	3
FC	<u>TMR</u>	–	–	16-bit timer	5
FC	<u>TMRH</u>	–	–	16-bit timer	5
FC	<u>CNT</u>	–	–	16-bit counter	5
FC	–	<u>DCNT</u>	–	32-bit counter	5

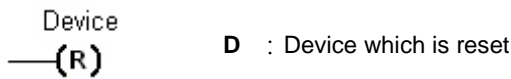
FB/FC	Instruction			Operand				Description					
FC		RST		D				Resetting the contact or clearing the register					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●	●	●	●	●	●	●	●	●	●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic Expression:



Explanation:

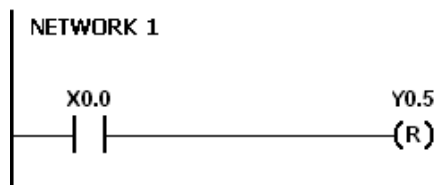
1. When the instruction RST is driven, the action of the device specified is as follows.

Device	status
Bit	The coil and the contact are set to OFF.
T, C, and HC/AC	The timer and the counter are reset to 0, and the coil and the contact are set to OFF.
Word	The value is cleared to 0.

2. If the instruction RST is not executed, the status of the device specified is unchanged.
3. The instruction supports the direct output.

Example:

When X0.0 is ON, Y0.5 is set to OFF.



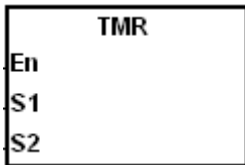
FB/FC	Instruction			Operand			Description					
FC	TMR			S₁, S₂			16-bit timer					

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁		●					●							
S₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁					●												
S₂	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

3 Graphic expression:



S₂ : Timer number

S₂ : Setting value of the timer

Explanation:

Please refer to the explanation of the instruction TMRH for more information.

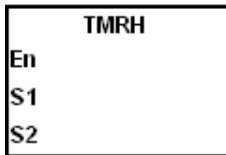
FB/FC	Instruction			Operand				Description					
FC		TMRH		S₁, S₂				16-bit timer					

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁					○												
S ₂	○	○						○	○		○		○	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S₁ : Timer number

S₂ : Setting value of the timer

Explanation:

- The timer used in the instruction TMR takes 100 milliseconds as the timing unit, and the timer used in the instruction TMRH takes 1 millisecond as the timing unit.
- The timers for the subroutine's exclusive use are T1920~T2047.
- The values of the timers used in TMR and TMRH should be within the range between 0 and 32767.
- If the same timer is used repeatedly in the program, including in the different instructions TMR and TMRH, the setting value is the one that the value of the timer matches first.
- As long as you add the letter S in front of the device T, the timer used in the instruction TMR becomes the accumulative timer. When the conditional contact is OFF, the accumulative timer value is not reset. When the conditional contact is ON, the timer counts from the current value.
- If the same timer is used repeatedly in the program, it is OFF when one of the conditional contacts is OFF.
- If the same timer is used repeatedly as the timer for the subroutine's exclusive use and the accumulative timer in the program, it is OFF when one of the conditional contacts is OFF.
- When the timer is switched from ON to OFF and the conditional contact is ON, the timer is reset and counts again.
- 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 state of the contact is as follows.

Normally open (NO) contact	ON
Normally closed (NC) contact	OFF

Example 1:

When X0.0 is ON, the setting value 50 is loaded to the timer T0. When the value of T0 matches 50, the contact of T0 is ON.



Example 2:

When X0.0 is ON, the setting value 50 is loaded to the timer T0. When the value of T0 is 25 and X0.0 is switched from OFF to ON, T0 counts up from 25 to 50, and the contact of T0 is ON.



Example 3:

When X0.0 is ON, the setting value 1000 is loaded to the timer T5. When the value of T5 matches 1000, the contact of T5 is ON.



Example 4:

When X0.0 is ON, the setting value 1000 is loaded to the timer T5. When the value of T5 is 500 and X0.0 is switched from OFF to ON, T0 counts up from 50 to 1000, and the contact of T5 is ON.



Points to Note :

If you declare the operand **S₁** in ISPSOft, the data type will be TIMER.

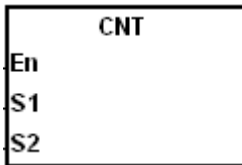
FB/FC	Instruction			Operand				Description				
FC		CNT		S₁, S₂				16-bit counter				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁						○											
S ₂	○	○						○	○		○		○	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S₁ : Counter number

S₂ : Setting value of the counter

Explanation:

- When the instruction CNT is executed, the coil of the counter is ON, and the value of the counter increases by one. When the value of the counter matches the setting value, the state of the contact is as follows.

Normally open (NO) contact	ON
Normally closed (NC) contact	OFF

- After the value of the counter matches the setting value, if there is still a pulse input signal of the counter, the state of the contact and the value of the counter remain unchanged. If you want to clear the value of the counter, you can use the instruction RST.

Example:

When SM408 is ON for the first time, the setting value 10 is loaded to the counter C0, and C0 begins to count. After SM408 is switched from OFF to ON ten times, the value of C0 is 10, and the contact of C0 is ON.

After the contact of C0 is ON, the value of C0 does not increase although SM408 still turns from OFF to ON.



Points to Note:

If you declare the operand S1 in ISPSOft, the data type will be COUNTER.

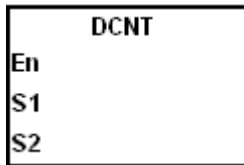
FB/FC	Instruction			Operand			Description		
FC	DCNT			S₁, S₂			32-bit counter		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁			●					●						
S₂			●					●						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁							○										
S₂	○	○						○	○		○		○	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	-	AH Motion CPU

Graphic expression:



S₁ : Counter value

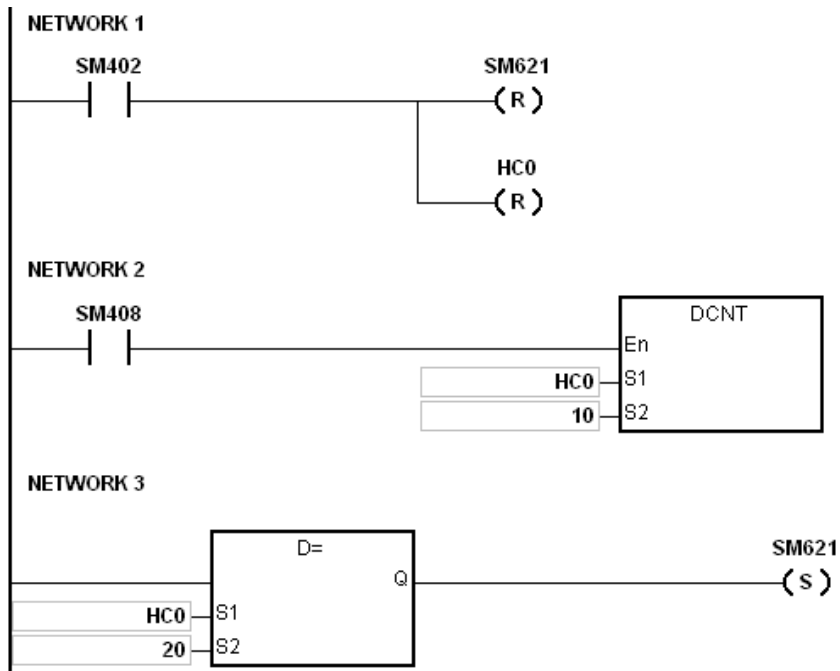
S₂ : Setting value of the counter

Explanation:

1. The instruction DCNT can be used to enable the 32-bit counter within the range between HC0 and HC63.
2. When the instruction DCNT is executed, the switch between the 32-bit general-purpose addition counters and the 32-bit general-purpose subtraction counters depends on states of the special auxiliary relays SM621~SM684.
3. When the instruction DCNT is not executed, the counter stops counting, and the original value of the counter is not cleared. You can use the instruction RST to clear the value of the counter and reset the contact.

Example:

1. When the PLC runs, SM621 is OFF, and the value of HC0 is cleared. When SM408 is ON for the first time, the setting value 10 is loaded to HC0, and HC0 begins to count up.
2. After SM408 is switched from OFF to ON ten times, the value of HC0 is 10, and the contact of HC0 is ON.
3. After HC0 is ON, the value of HC0 keeps increasing because SM408 is still switched from OFF to ON.
4. When the value of HC0 is 20, SM621 is ON. After SM408 is switched from OFF to ON ten times, the contact of HC0 is OFF.
5. After the contact of HC0 is OFF, the value of HC0 keeps decreasing because SM408 is still switched from OFF to ON.

**Additional remark:**

1. Please refer to the usage of 32-bit counters in chapter 2 for more information related to SM621~SM684.
2. If you declare the operand S1 in ISPSOft, the data type will be COUNTER.

3.14 Shift Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>SFTR</u>	–	✓	Shifting the states of the devices to the right	9
FC	<u>SFTL</u>	–	✓	Shifting the states of the devices to the left	9
FC	<u>WSFR</u>	–	✓	Shifting the data in the word devices to the right	9
FC	<u>WSFL</u>	–	✓	Shifting the data in the word devices to the left	9
FC	<u>SFWR</u>	–	✓	Shifting the data and writing it into the word device	7
FC	<u>SFRD</u>	–	✓	Shifting the data and reading it from the word device	7
FC	<u>SFPO</u>	–	✓	Reading the latest data from the data list	5
FC	<u>SFDEL</u>	–	✓	Deleting the data from the data list	7
FC	<u>SFINS</u>	–	✓	Inserting the data into the data list	7
FC	<u>MBS</u>	–	✓	Shifting the matrix bits	7
FC	<u>SFR</u>	–	✓	Shifting the values of the bits in the 16-bit registers by n bits to the right	5
FC	<u>SFL</u>	–	✓	Shifting the values of the bits in the 16-bit registers by n bits to the left	5
FC	<u>BSFR</u>	–	✓	Shifting the states of the n bit devices by one bit to the right	5
FC	<u>BSFL</u>	–	✓	Shifting the states of the n bit devices by one bit to the left	5
FC	<u>NSFR</u>	–	✓	Shifting n registers to the right	5
FC	<u>NSFL</u>	–	✓	Shifting n registers to the left	5

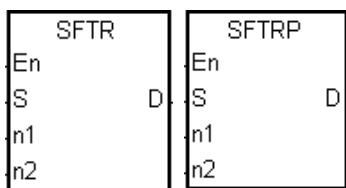
FB/FC	Instruction		Operand	Description
FC	SFTR	P	S, D, n ₁ , n ₂	Shifting the states of the devices to the right

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●													
D	●													
n ₁ , n ₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●	●	●				●	●	●			●				
D	●	●	●	●				●	●	●			●				
n ₁ , n ₂	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



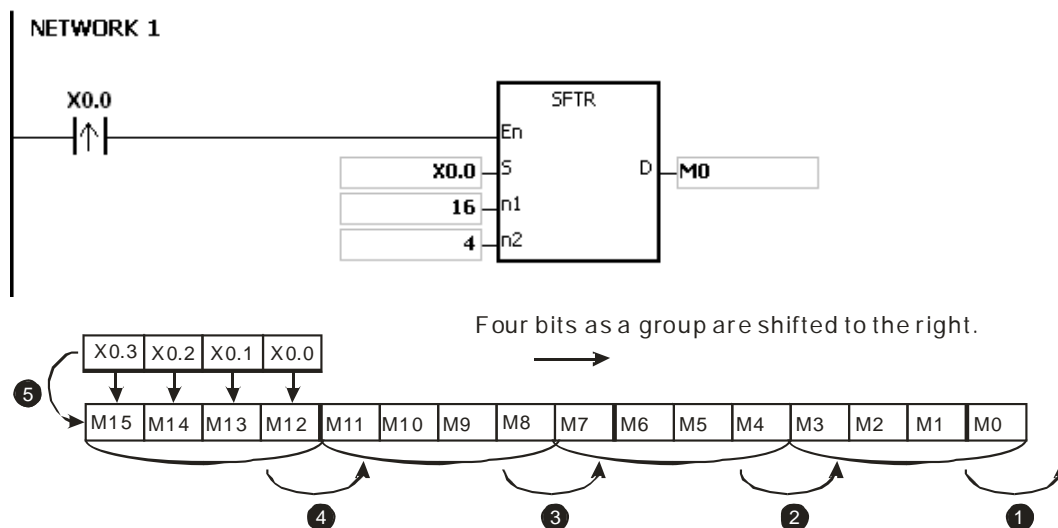
- S** : Initial device in which the value is shifted
- D** : Initial device in which the value is shifted
- n₁** : Length of the data which is shifted
- n₂** : Number of bits forming a group

Explanation:

1. The states of the n₁ bit devices starting from D are divided into groups (n₂ bits as a group), and these groups are shifted to the right. The states of the n₂ bit devices starting from S are shifted to the devices starting from D to fill the vacancy.
2. Generally, the pulse instruction SFTRP is used.
3. The operand n₁ should be within the range between 1 and 1024. The operand n₂ should be within the range between 1 and n₁.

Example 1:

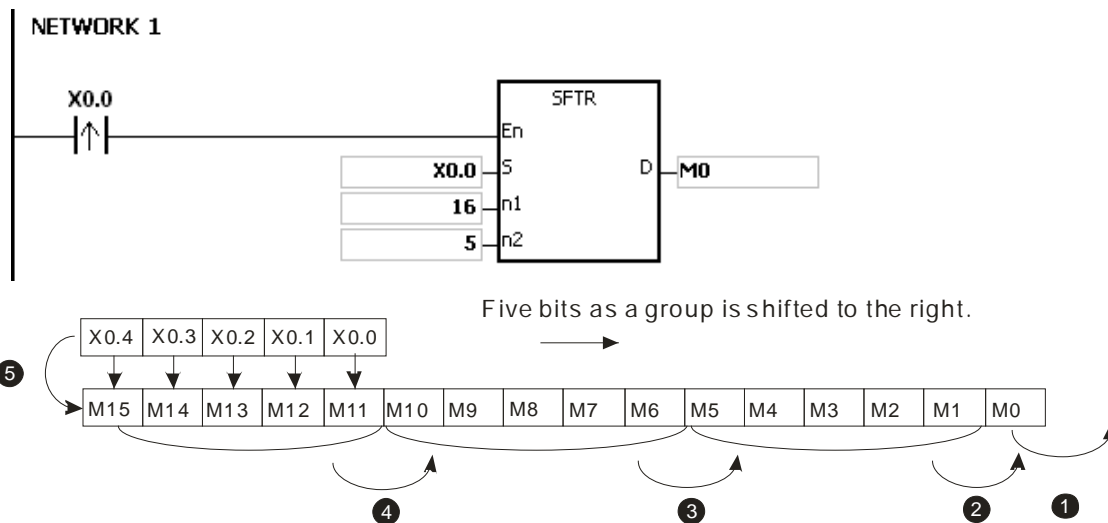
1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (four bits as a group), and these groups are shifted to the right.
2. The shift of the states of the bit devices to the right during a scan is illustrated as follows.
 - ① M3~M0 → Being carried
 - ② M7~M4 → M3~M0
 - ③ M11~M8 → M7~M4
 - ④ M15~M12 → M11~M8
 - ⑤ X0.3~X0.0 → M15~M12



Example 2:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (five bits as a group), and these groups are shifted to the right.
2. The shift of the states of the bit devices to the right during a scan is illustrated as follows.

- ① M0 → Being carried
- ② M5 → M0
- ③ M10~M6 → M5~M1
- ④ M15~M11 → M10~M6
- ⑤ X0.4~X0.0 → M15~M11



Additional remark:

1. If S+n2-1 or D+n1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If n1 is less than 1, or if n1 is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If n2 is less than 1, or if n2 is larger than n1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

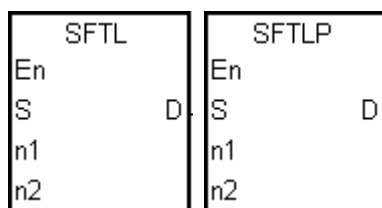
FB/FC	Instruction			Operand				Description					
FC		SFTL	P	S, D, n₁, n₂				Shifting the states of the devices to the left					

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●													
D	●													
n₁, n₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●	●	●				●	●	●			●				
D	●	●	●	●				●	●	●			●				
n₁, n₂	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



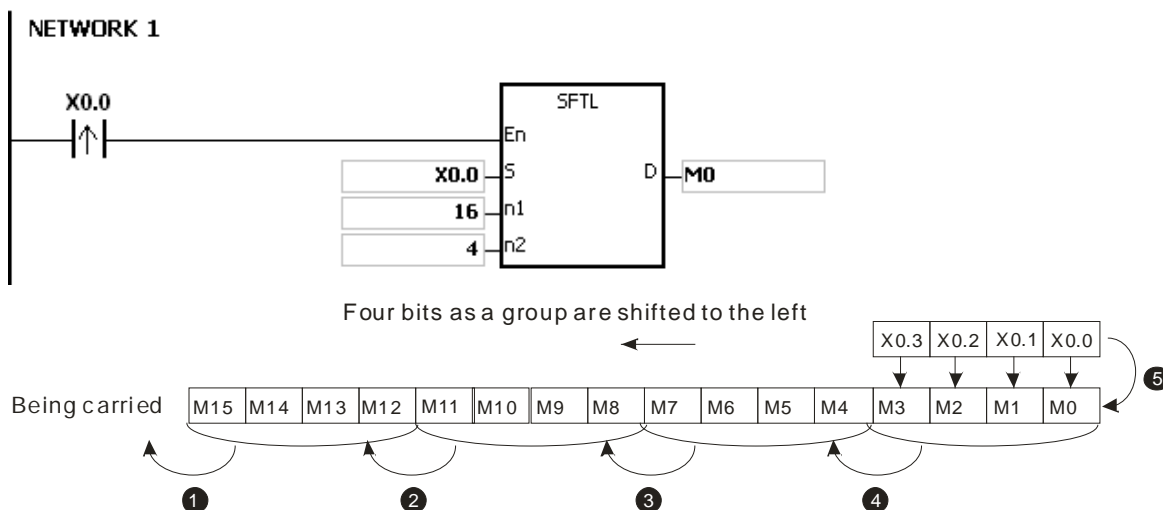
- S** : Initial device in which the value is shifted
- D** : Initial device in which the value is shifted
- n₁** : Length of the data which is shifted
- n₂** : Number of bits forming a group

Explanation:

1. The states of the n₁ bit devices starting from D are divided into groups (n₂ bits as a group), and these groups are shifted to the left. The states of the n₂ bit devices starting from S are shifted to the devices starting from D to fill the vacancy.
2. Generally, the pulse instruction SFTLP is used.
3. The operand n₁ should be within the range between 1 and 1024. The operand n₂ should be within the range between 1 and n₁.

Example 1:

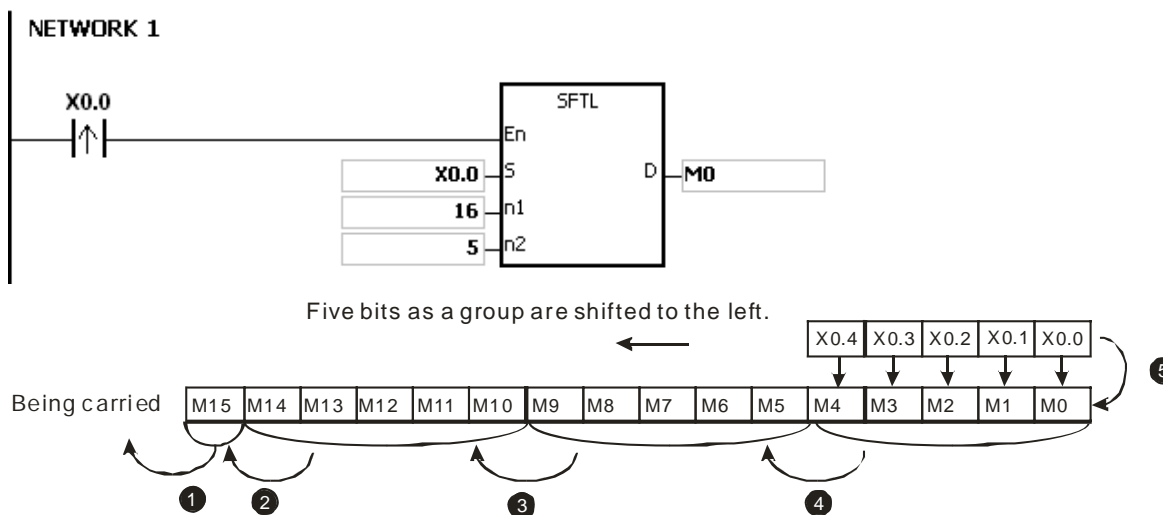
1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (four bits as a group), and these groups are shifted to the left.
2. The shift of the states of the bit devices to the left during a scan is illustrated as follows.
 - ① M15~M12 → Being carried
 - ② M11~M8 → M15~M12
 - ③ M7~M4 → M11~M8
 - ④ M3~M0 → M7~M4
 - ⑤ X0.3~X0.0 → M3~M0



Example 2:

1. When X0.0 is switched from OFF to ON, the states of the sixteen bit devices starting from M0 are divided into groups (five bits as a group), and these groups are shifted to the left.
2. The shift of the states of the bit devices to the left during a scan is illustrated as follows.

- ① M15 → Being carried
- ② M10 → M15
- ③ M9~M5 → M14~M10
- ④ M4~M0 → M9~M5
- ⑤ X0.4~X0.0 → M4~M0



Additional remark:

1. If S+n2-1 or D+n1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If n1 is less than 1, or if n1 is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If n2 is less than 1, or if n2 is larger than n1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

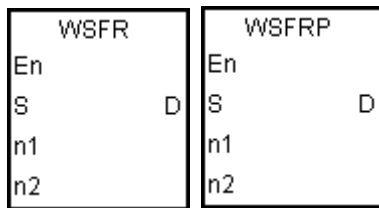
FB/FC	Instruction		Operand	Description
FC	WSFR	P	S, D, n₁, n₂	Shifting the data in the word devices to the right

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●					●							
D		●					●							
n₁, n₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n₁, n₂	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



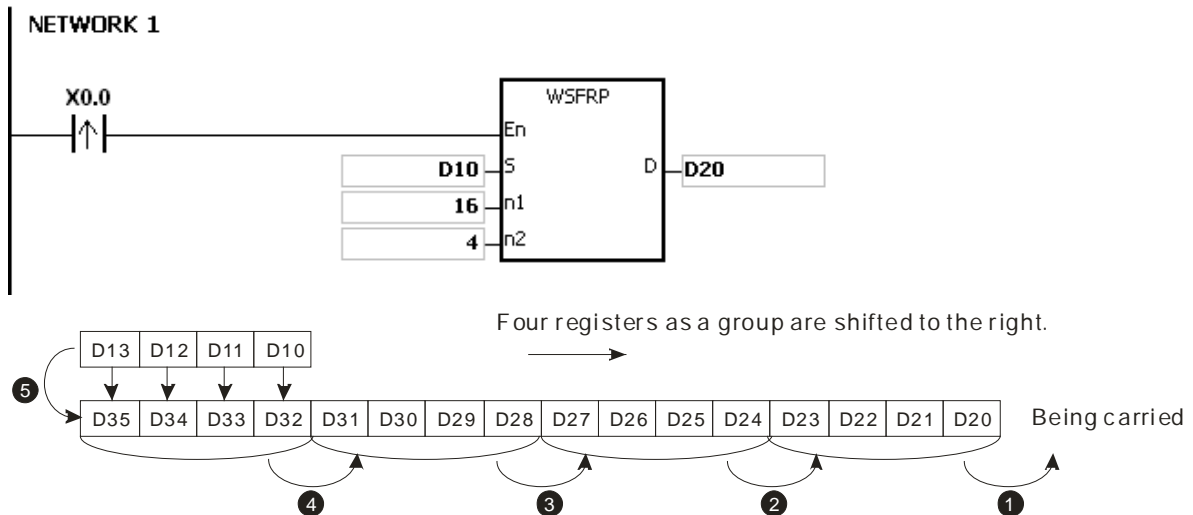
- S** : Initial device in which the value is shifted
- D** : Initial device in which the value is shifted
- n₁** : Length of the data which is shifted
- n₂** : Number of bits forming a group

Explanation:

1. The data in the n₁ word devices starting from D is divided into groups (n₂ words as a group), and these groups are shifted to the right. The data in the n₂ word devices starting from S are shifted to the devices starting from D to fill the vacancy.
2. Generally, the pulse instruction WSFRP is used.
3. The operand n₁ should be within the range between 1 and 512. The operand n₂ should be within the range between 1 and n₁.

Example 1:

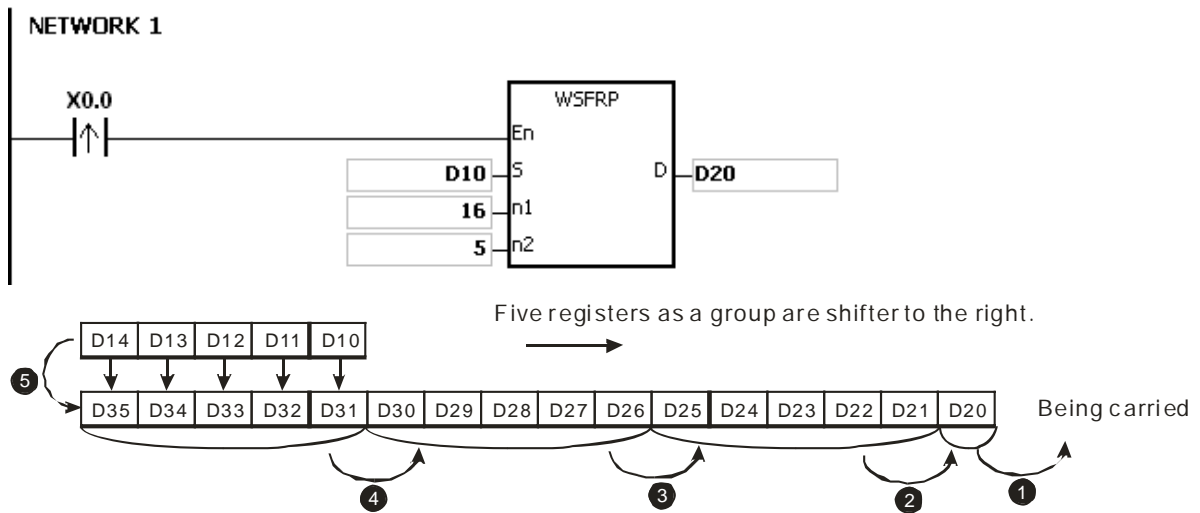
1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 is divided into groups (four words as a group), and these groups are shifted to the right.
2. The shift of the data in the word devices to the right during a scan is illustrated as follows.
 - ① D23~D20 → Being carried
 - ② D27~D24 → D23~D20
 - ③ D31~D28 → D27~D24
 - ④ D35~D32 → D31~D28
 - ⑤ D13~D10 → D35~D32



Example 2:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 is divided into groups (five words as a group), and these groups are shifted to the right.
2. The shift of the data in the word devices to the right during a scan is illustrated as follows.

- ① D20 → Being carried
- ② D25 → D20
- ③ D30~D26 → D25~D21
- ④ D35~D31 → D30~D26
- ⑤ D14~D10 → D35~D31



Additional remark:

1. If S+n2-1 or D+n1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If n1 is less than 1, or if n1 is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If n2 is less than 1, or if n2 is larger than n1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand	Description
FC	WSFL	P	S, D, n₁, n₂	Shifting the data in the word devices to the left

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●					●							
D		●					●							
n₁, n₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n₁, n₂	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



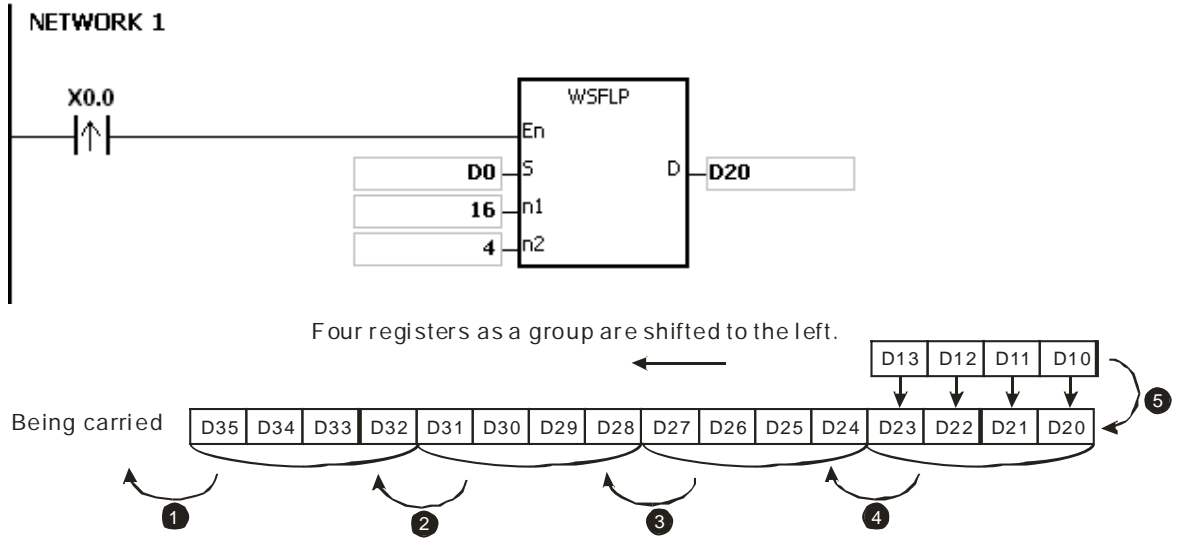
- S** : Initial device in which the value is shifted
- D** : Initial device in which the value is shifted
- n₁** : Length of the data which is shifted
- n₂** : Number of bits forming a group

Explanation:

1. The data in the n₁ word devices starting from D is divided into groups (n₂ words as a group), and these groups are shifted to the left. The data in the n₂ word devices starting from S are shifted to the devices starting from D to fill the vacancy.
2. Generally, the pulse instruction WSFLP is used.
3. The operand n₁ should be within the range between 1 and 512. The operand n₂ should be within the range between 1 and n₁.

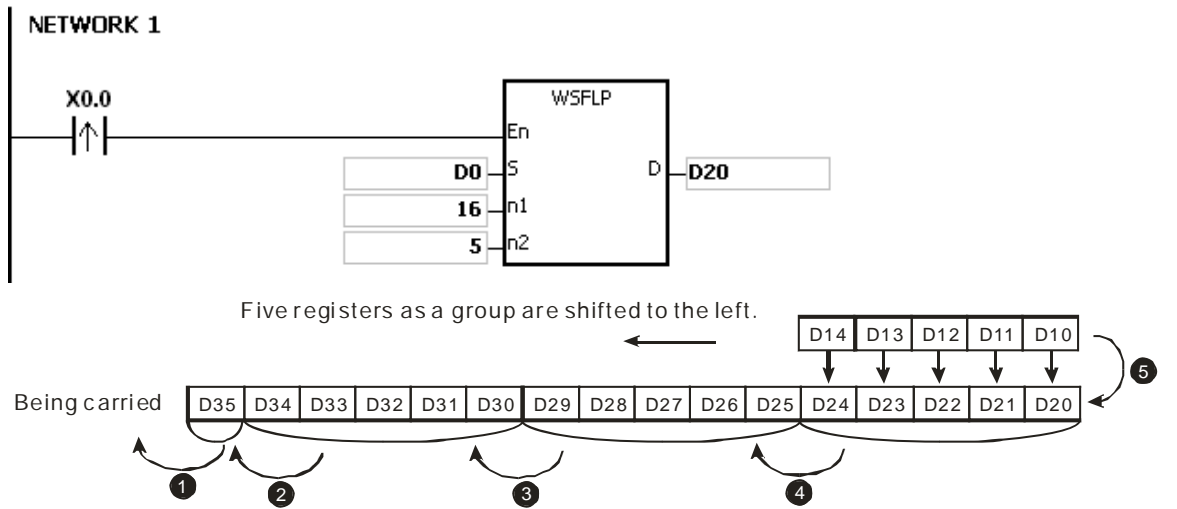
Example 1:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 is divided into groups (four words as a group), and these groups are shifted to the left.
2. The shift of the data in the word devices to the left during a scan is illustrated as follows.
 - ① D35~D32 → Being carried
 - ② D31~D28 → D35~D32
 - ③ D27~D24 → D31~D28
 - ④ D23~D20 → D27~D24
 - ⑤ D13~D10 → D23~D20



Example 2:

1. When X0.0 is switched from OFF to ON, the data in the sixteen word devices starting from D20 is divided into groups (five words as a group), and these groups are shifted to the left.
2. The shift of the data in the word devices to the left during a scan is illustrated as follows.
 - ① D35 → Being carried
 - ② D30 → D35
 - ③ D29~D25 → D34~D30
 - ④ D24~D20 → D29~D25
 - ⑤ D14~D10 → D24~D20



Additional remark:

1. If S+n2-1 or D+n1-1 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If n1 is less than 1, or if n1 is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If n2 is less than 1, or if n2 is larger than n1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

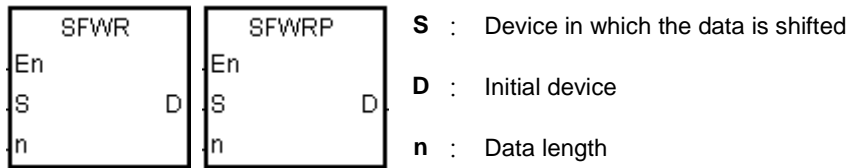
FB/FC	Instruction		Operand	Description
FC	SFWR	P	S, D, n	Shifting the data and writing it into the word device

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●		●				
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

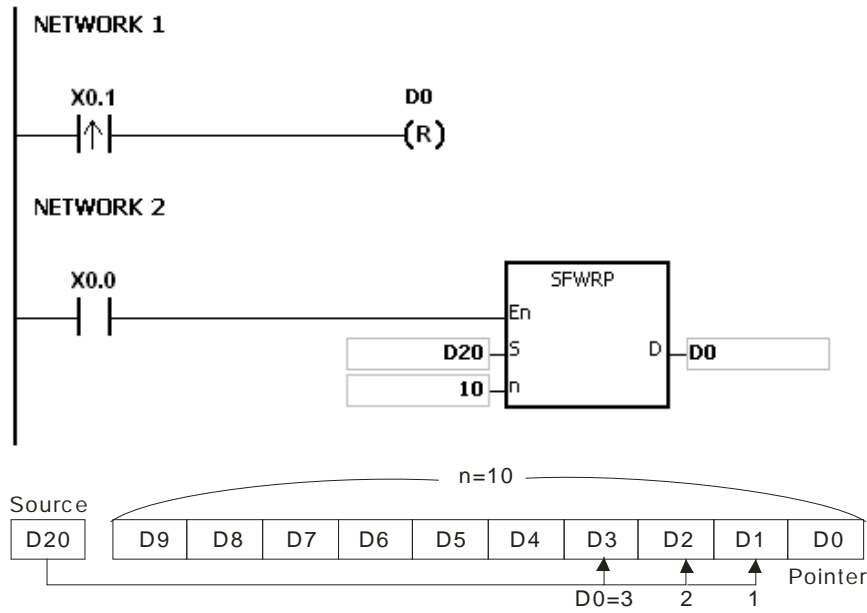


Explanation:

- The data in the n word devices starting from the device specified by D is defined as a first in, first out data type, and the device specified by D is taken as a pointer. When the instruction is executed, the value of the pointer increases by one, and the data in the device specified by S is written into the device specified by the pointer. When the value of the pointer is larger than or equal to n-1, the instruction does not process the writing of the data, and the carry flag SM602 is ON.
- Generally, the pulse instruction SFWRP is used.
- The operand n should be within the range between 2 and 512.

Example:

- The value of the pointer D0 is cleared to 0 first. When X0.0 is switched from OFF to ON, the data in D20 is written into D1, and the value in D0 becomes 1. When X0.0 is switched from OFF to ON again, the data in D20 is written to D2, and the value in D0 becomes 2.
- The data in the word device is shifted and written in the following way.
 - The data in D20 is written into D1.
 - The value in D0 becomes 1.



3

Additional remark:

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 is less than 2, or if n is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. The instruction SFWR can be used with the instruction SFRD to write and read the data.

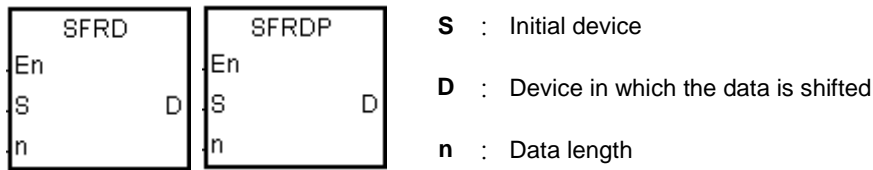
FB/FC	Instruction		Operand				Description					
FC		SFRD	P	S, D, n				Shifting the data and reading it from the word device				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●	○	●	○	○		
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

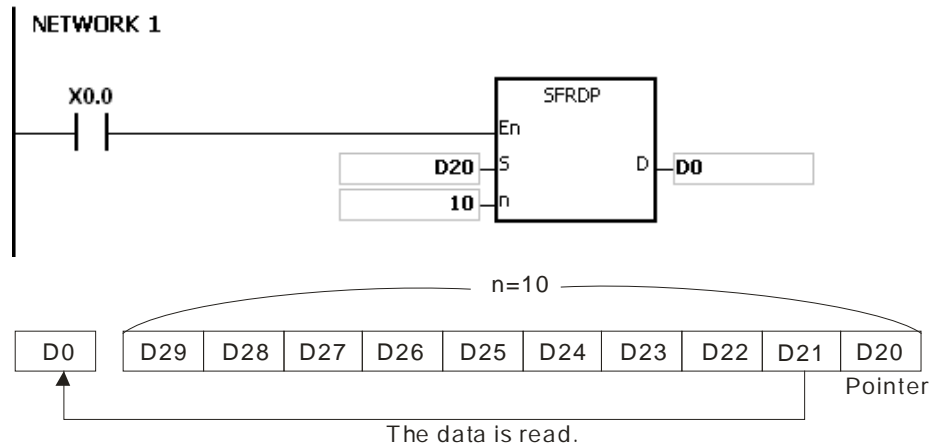


Explanation:

1. The data in the n word devices starting from the device specified by S is defined as a first in, first out data type, and the device specified by S is taken as a pointer. When the instruction is executed, the value in the device specified by S decreases by one, the data in the device specified by S+1 is written into the device specified by D, the data in the devices specified by S+n-1~S+2 is shifted to the right, and the data in the device specified by S+n-1 is unchanged. When the value in the device specified by S is equal to 0, the instruction does not process the reading of the data, and the zero flag SM600 is ON.
2. Generally, the pulse instruction SFRDP is used.
3. The operand n should be within the range between 2 and 512.

Example:

1. When X0.0 is switched from OFF to ON, the data in D21 is written into D0, the data in D29~D22 is shifted to the right, the data in D29 is unchanged, and the value in D20 decreases by one.
2. The data in the word device is shifted and read in the following way.
 - 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 remark:

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 is less than 2, or if n is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. The instruction SFWR can be used with the instruction SFRD to write and read the data.

3

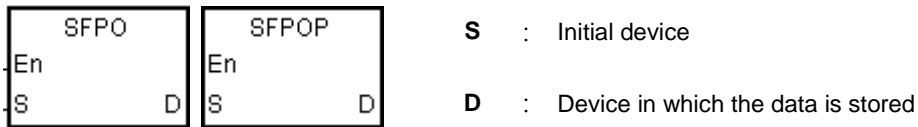
FB/FC	Instruction		Operand				Description						
FC		SFPO	P	S, D				Reading the latest data from the data list					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

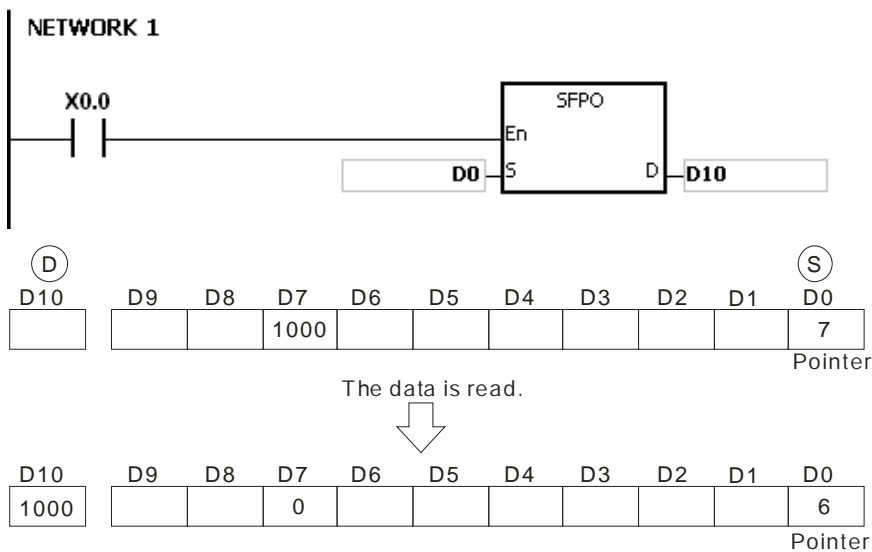


Explanation:

- The device specified by S is taken as a pointer. When the instruction is executed, the data in the device specified by the value of the pointer is written into the device specified by D and cleared to 0, and the value in the device specified by S decreases by one. When the value in the device specified by S is equal to 0, the instruction does not process the reading of the data, and the zero flag SM600 is ON.
- Generally, the pulse instruction SFPOP is used.

Example:

When X0.0 is ON, the data in the device specified by the value in D0 is written into D10. After the data is shifted, the data in the device specified by the value in D0 is cleared to 0, and the value in D0 increases by one.



Additional remark:

- 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.

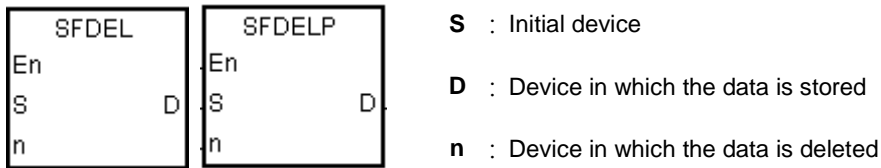
FB/FC	Instruction		Operand	Description
FC	SFDEL	P	S, D, n	Deleting the data from the data list

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●	○	●	○	○		
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

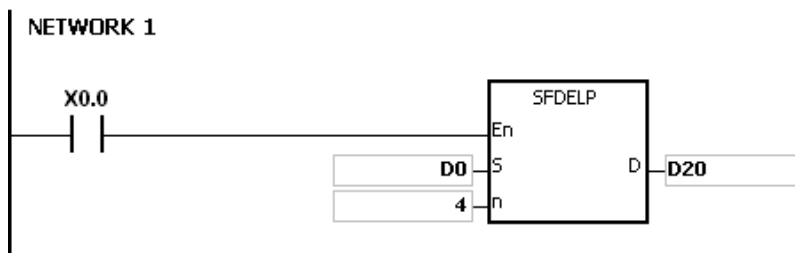


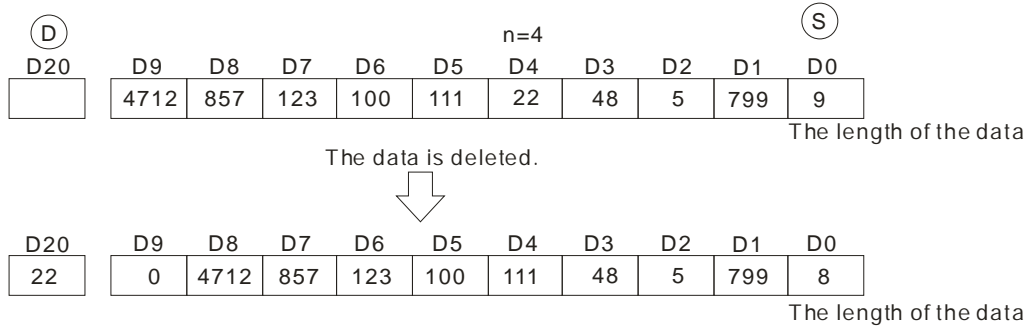
Explanation:

1. The value in the device specified by S indicates the length of the data, and the data is in the devices specified by S+1~S+(The value in S). When the instruction is executed, the data in the device specified by S+n is stored in D and deleted, the data in the devices specified by S+n+1~S+(The value in S) is shifted to the right, the data in the device specified by S+(The value in S) is cleared to 0, and the value in the device specified by S decreases by one. When the value in the device specified by S is equal to 0, the instruction does not process the deleting of the data, and the zero flag SM600 is ON.
2. Generally, the pulse instruction SFDELP is used.
3. The operand n should be within the range between 1 and 32767.

Example:

Suppose the value in D0 is 9, and n is 4. When X0.0 is ON, the data in D4 is stored in D20. After the data in D4 is deleted, the data in D5~D9 is shifted to the right, and the value in D0 decreases by one.





Additional remark:

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 larger than the value in S, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
5. If n is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

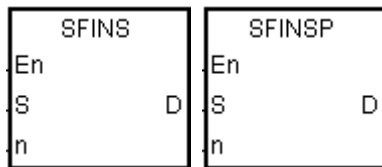
FB/FC	Instruction		Operand	Description
FC	SFINS	P	S, D, n	Inserting the data into the data list

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●	○	●	○	○		
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



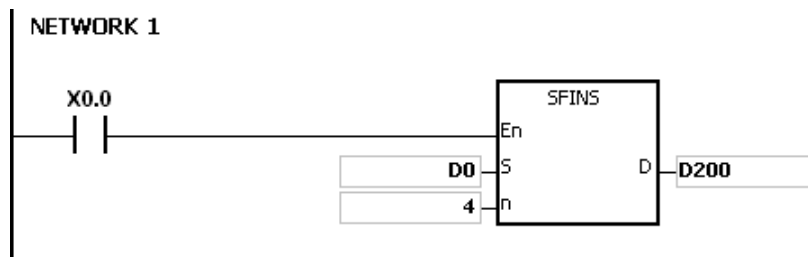
- S : Initial device
- D : Data which is inserted
- n : Device into which the data is inserted

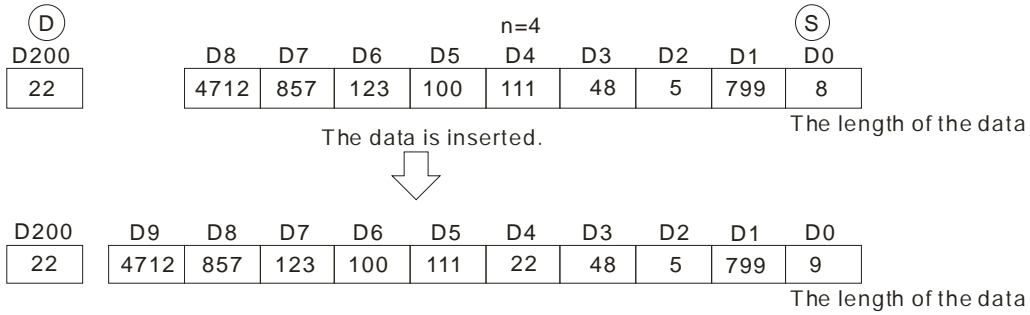
Explanation:

- The value in the device specified by S indicates the length of the data, and the data is in the devices specified by S+1~S+(The value in S). When the instruction is executed, the data in D is inserted into S+n, the original data in the devices specified by S+n~S+(The value in S) is shifted to the left, and the value in the device specified by S increases by one. When the value in the device specified by S is equal to 32767, the instruction does not process the writing of the data, the value in the device specified by S does not increase, and the carry flag SM602 is ON.
- Generally, the pulse instruction SFINSP is used.
- The operand n should be within the range between 1 and 32767.

Example:

Suppose the value in D0 is 8, and n is 4. When X0.0 is ON, the data in D200 is inserted into D4, the original data in D4~D8 is shifted to D5~D9, and the value in D0 increases by one.





Additional remark:

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 larger than the value in S, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
5. If n is less than 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand	Description
FC	MBS	P	S, D, n	Shifting the matrix bits

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●				●				
n	●	●			●	●	●	●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



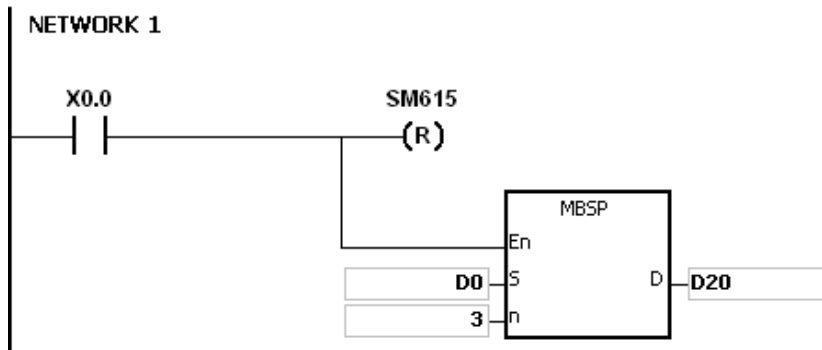
- S : Matrix source
- D : Operation result
- n : Length of the array

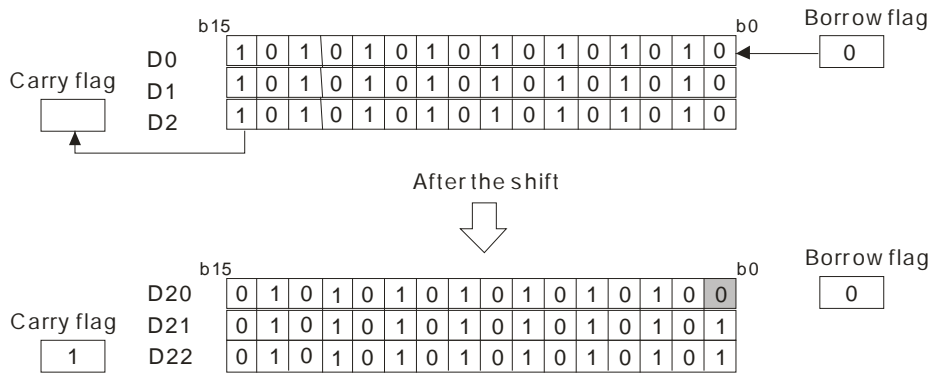
Explanation:

- The values of the n rows of bits in S are shifted to the right or to the left. When SM616 is OFF, the values of the bits are shifted to the left. When SM616 is ON, the values of the bits are shifted to the right. The vacancy resulting from the shift is filled by the state of the borrow flag SM615, the value of the bit shifted last is transmitted to the carry flag SM614, and the operation result is stored in D.
- The operand n should be within the range between 1 and 256.
- Generally, the pulse instruction MBSP is used.

Example 1:

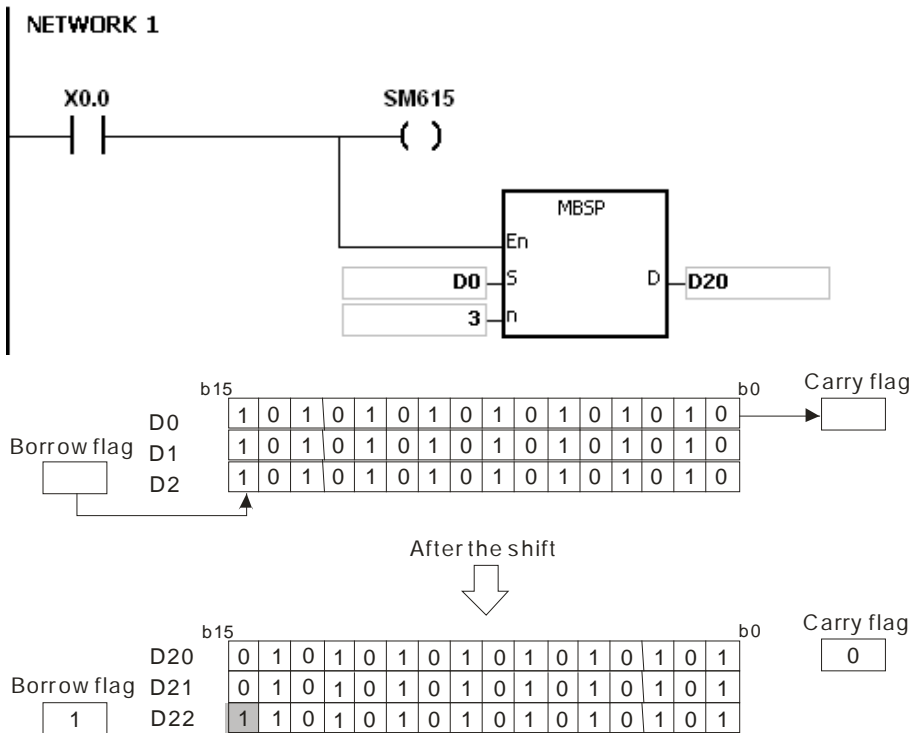
When X0.0 is ON, SM616 is OFF. The values of the bits are shifted to the left. Suppose SM615 is OFF. After the values of the bits in the 16-bit registers D0~D2 are shifted to the left, the operation result is stored in the 16-bit registers D20~D22, and SM614 is ON.





Example 2:

When X0.0 is ON, SM616 is ON. The values of the bits are shifted to the right. Suppose SM615 is ON. After the values of the bits in the 16-bit registers D0~D2 are rotated to the right, the operation result is stored in the 16-bit registers D20~D22, and SM614 is OFF.



Additional remark:

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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. The flags:
 - SM614: It is the carry flag for the matrix rotation/shift/output.
 - SM615: It is the borrow flag for the matrix shift/output.
 - SM616: It is the direction flag for the matrix rotation/shift.

FB/FC	Instruction		Operand	Description
FC	SFR	P	D, n	Shifting the values of the bits in the 16-bit registers by n bits to the right

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

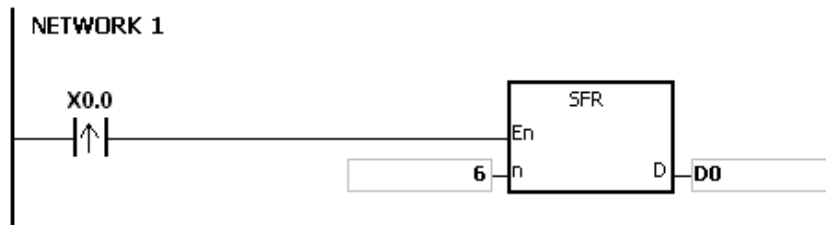


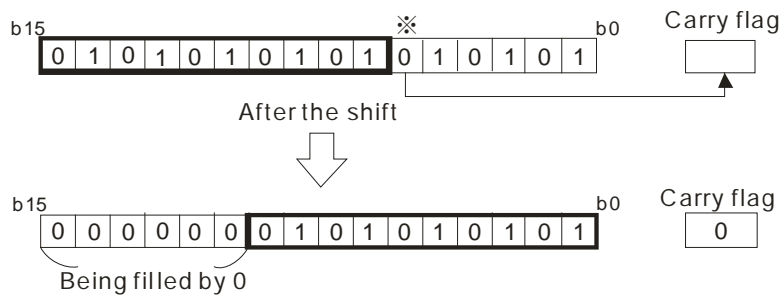
Explanation:

- The values of the bits in D are shifted by n bits to the right. The vacancies (b15~b15-n+1) resulting from the shift is filled by 0, and the value of bn-1 is transmitted to SM602.
- The operand n should be within the range between 1 and 16.
- Generally, the pulse instruction SFRP is used.

Example:

- When X0.0 is ON, the values of b0~b15 in D0 are shifted by 6 bits to the right, and the value of b5 is transmitted to SM602. The values of b10~b15 are cleared to 0 after the shift.
- The shift of the values of the bits to the right during a scan is illustrated as follows.
 - ❶ b5~b0 → Being carried (The value of b5 is transmitted to SM602.)
 - ❷ b15~b6 → b9~b0
 - ❸ 0 → b15~b10



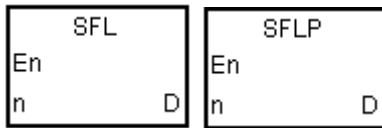


Additional remark:

If **n** is less than 0, or if **n** is larger than 16, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand				Description									
FC		SFL	P	D, n				Shifting the values of the bits in the 16-bit registers by n bits to the left									
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING			
D		●					●										
n		●					●										
Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		
				Pulse instruction				16-bit instruction				32-bit instruction					
				AH Motion CPU				AH Motion CPU				-					

Graphic expression:



D : Device involved in the shift

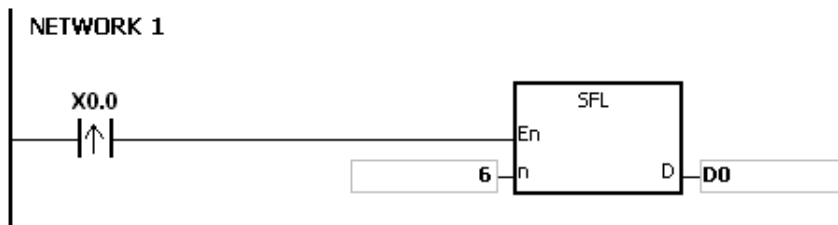
n : Number of bits

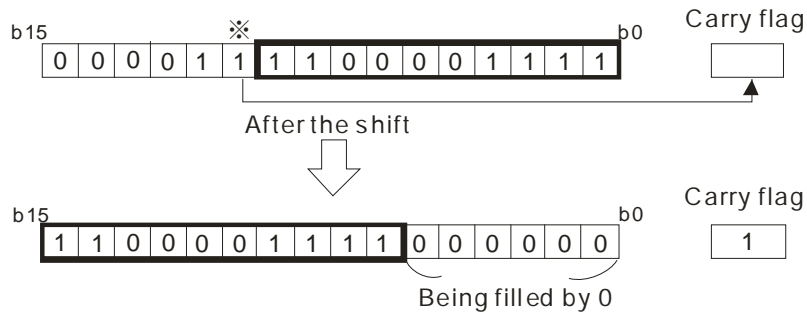
Explanation:

1. The values of the bits in D are shifted by n bits to the left. The vacancies (b0~bn-1) resulting from the shift is filled by 0, and the value of b16-n is transmitted to SM602.
2. The operand n should be within the range between 1 and 16.
3. Generally, the pulse instruction SFLP is used.

Example:

4. When X0.0 is ON, the values of b0~b15 in D0 are shifted by 6 bits to the right, and the value of b10 is transmitted to SM602. The values of b0~b5 are cleared to 0 after the shift.
5. The shift of the values of the bits to the left during a scan is illustrated as follows.
 - ① b15~b10 → Being carried (The value of b10 is transmitted to SM602.)
 - ② b9~b0 → b15~b6
 - ③ 0 → b5~b0





Additional remark:

If **n** is less than 0, or if **n** is larger than 16, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand	Description
FC	BSFR	P	D, n	Shifting the states of the n bit devices by one bit to the right

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●	●	●				●	●	●			●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



D : Initial device involve in the shift

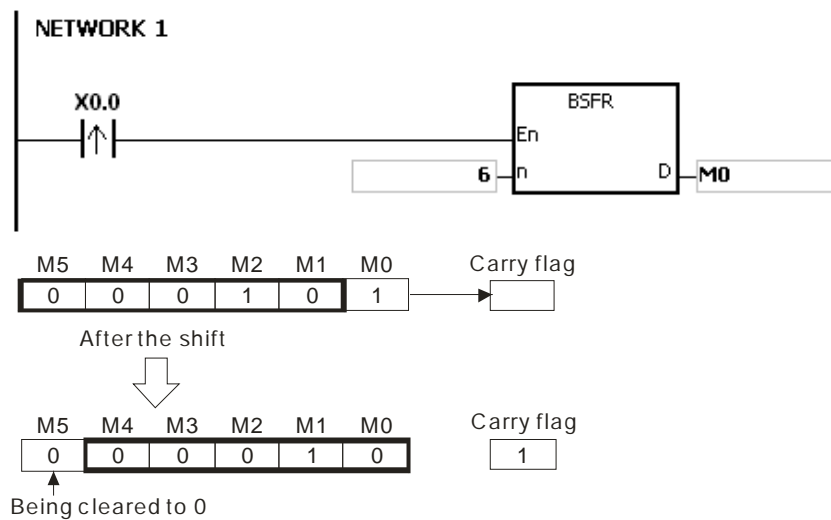
n : Data length

Explanation:

1. The states of the **n** bit devices starting from **D** are shifted by one bit to the right. The state of **D+n-1** is cleared to 0, and the state of **D** is transmitted to the carry flag SM602.
2. Generally, the pulse instruction BSFRP is used.
3. The operand **n** should be within the range between 1 and 1024.

Example:

When X0.0 is ON, the states of M0~M5 are shifted by one bit to the right, the state of M5 is cleared to 0, and the state of M0 is transmitted to the carry flag SM602.



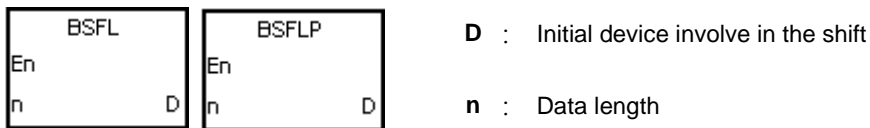
Additional remark:

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 less than 1, or if n is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand		Description												
FC	BSFL	P	D, n		Shifting the states of the n bit devices by one bit to the left												
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING			
D	●																
n		●					●										
Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●	●	●				●	●	●			●				
n	●	●			●	●		●	●		●	○	●	○	○		
					Pulse instruction				16-bit instruction				32-bit instruction				
					AH Motion CPU				AH Motion CPU				-				

Graphic expression:

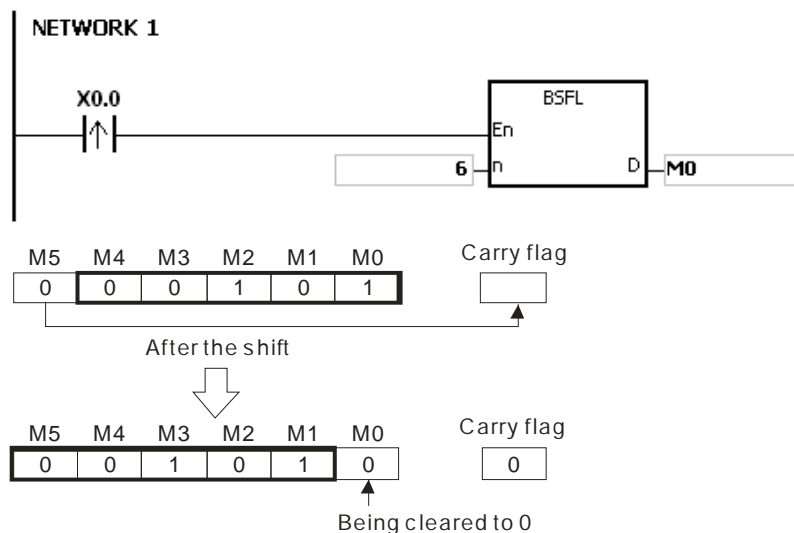


Explanation:

1. The states of the **n** bit devices starting from **D** are shifted by one bit to the left. The state of **D** is cleared to 0, and the state of **D+n-1** is transmitted to the carry flag **SM602**.
2. Generally, the pulse instruction **BSFLP** is used.
3. The operand **n** should be within the range between 1 and 1024.

Example:

When **X0.0** is ON, the states of **M0~M5** are shifted by one bit to the left, the state of **M0** is cleared to 0, and the state of **M5** is transmitted to the carry flag **SM602**.



Additional remark:

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 less than 1, or if n is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

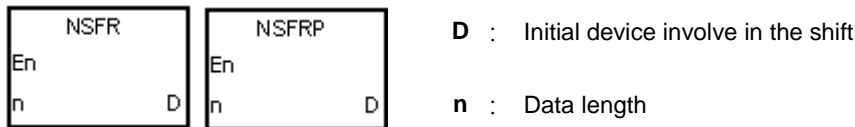
FB/FC	Instruction		Operand	Description
FC	NSFR	P	D, n	Shifting n registers to the right

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

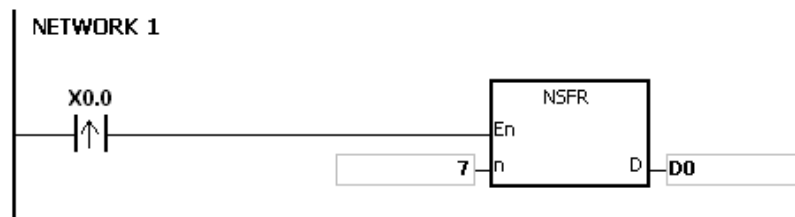


Explanation:

1. The data in the n registers starting from D is shifted to the right, and the data in D+n-1 is cleared to 0.
2. Generally, the pulse instruction NSFRP is used.
3. The operand n should be within the range between 1 and 512.

Example:

When X0.0 is ON, the data in D1~D6 is shifted to the right, and the data in D6 is cleared 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 remark:

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 less than 1, or if n is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction		Operand		Description
FC	NSFL	P	D, n		Shifting n registers to the left

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:



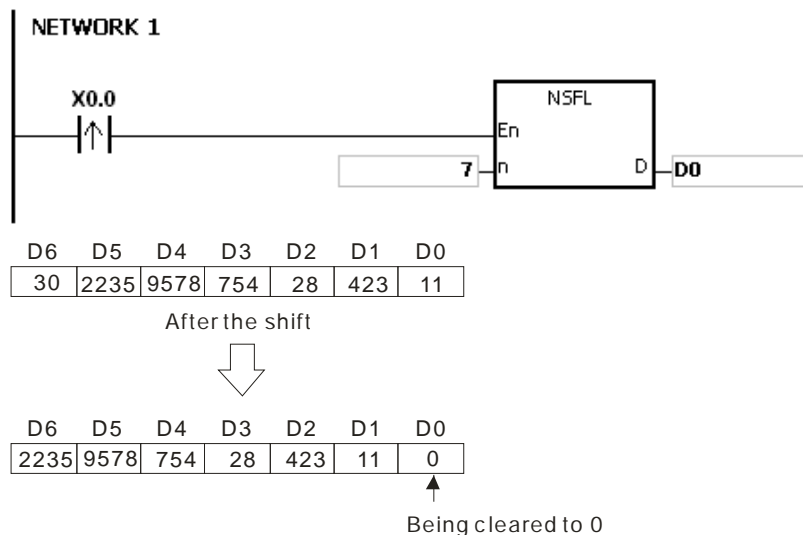
D : Initial device involve in the shift
n : Data length

Explanation:

1. The data in the n registers starting from D is shifted to the left, and the data in D is cleared to 0.
2. Generally, the pulse instruction NSFLP is used.
3. The operand n should be within the range between 1 and 512.

Example:

When X0.0 is ON, the data in D0~D5 is shifted to the left, and the data in D0 is cleared to 0.



Additional remark:

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 less than 1, or if n is larger than 512, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

3.15 Data Processing Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>SER</u>	<u>DSER</u>	✓	Searching the data	9
FC	<u>SUM</u>	<u>DSUM</u>	✓	Number of bits whose states are ON	5
FC	<u>DECO</u>	–	✓	Decoder	7
FC	<u>ENCO</u>	–	✓	Encoder	7
FC	<u>SEGD</u>	–	✓	Seven-segment decoding	5
FC	<u>SORT</u>	<u>DSORT</u>	–	Sorting the data	11
FC	<u>ZRST</u>	–	✓	Resetting the zone	5
FC	<u>BON</u>	<u>DBON</u>	✓	Checking the state of the bit	7
FC	<u>MEAN</u>	<u>DMEAN</u>	✓	Mean	7
FC	<u>CCD</u>	–	✓	Sum check	7
FC	<u>ABS</u>	<u>DABS</u>	✓	Absolute value	3
FC	<u>MINV</u>	–	✓	Inverting the matrix bits	7
FC	<u>MBRD</u>	–	✓	Reading the matrix bit	7
FC	<u>MBWR</u>	–	✓	Writing the matrix bit	7
FC	<u>MBC</u>	–	✓	Counting the bits with the value 0 or 1	7
FC	<u>DIS</u>	–	✓	Disuniting the 16-bit data	7
FC	<u>UNI</u>	–	✓	Uniting the 16-bit data	7
FC	<u>WSUM</u>	<u>DWSUM</u>	✓	Getting the sum	7
FC	<u>BSET</u>	–	✓	Setting the bit in the word device to ON	5
FC	<u>BRST</u>	–	✓	Resetting the bit in the word device	5
FC	<u>BKRST</u>	–	✓	Resetting the specified zone	5
FC	<u>LIMIT</u>	<u>DLIMIT</u>	✓	Confining the value within the bounds	9
FC	<u>BAND</u>	<u>DBAND</u>	✓	Deadband control	9
FC	<u>ZONE</u>	<u>DZONE</u>	✓	Controlling the zone	9

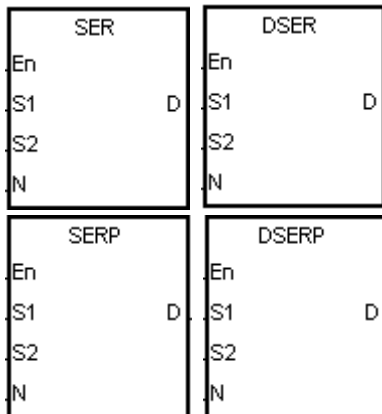
FB/FC	Instruction			Operand	Description
FC	D*	SER	P	S ₁ , S ₂ , D, n	Searching the data

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●	●*				●	●*						
S ₂		●	●*				●	●*						
D		●	●*				●	●*						
n		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●	●	●	●		●		●				
S ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●				●				
n	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Initial device involved in the comparison

S₂ : Compared data

D : Initial device in which the comparison result is stored

n : Data length

Explanation:

- n signed decimal values in the registers starting from the register specified by S1 are compared with the signed decimal value in the register specified by S2, and the comparison results are stored in the registers D~D+4.

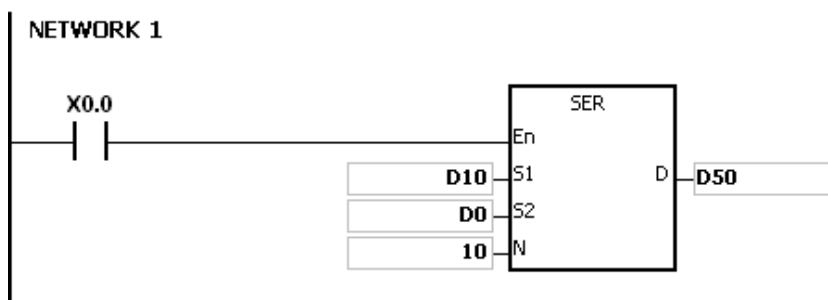
Device	Description
D	Number of equal values
D+1	Data number of the first equal value
D+2	Data number of the last equal value
D+3	Data number of the minimum value
D+4	Data number of the maximum value

- The operand n used in the 16-bit instruction should be within the range between 1 and 256. The operand n used in the 32-bit instruction should be within the range between 1 and 128.

- Only the 32-bit instructions can use the 32-bit counter.

Example:

- When X0.0 is ON, the values in D10~D19 are compared with the value in D0, and the comparison results are stored in D50~D54. When the equal value does not exist, the values in D50~D52 are 0.
- The data number of the minimum value is stored in D53, and the data number of the maximum value is stored in D54. If there is more than one minimum value or maximum value, the data number which is bigger is stored.



S ₁	Value	Compared data	Data number	Result	D	Value	Description
D10	88	S ₂ D0=100	0		D50	4	Number of equal values
D11	100		1	Equal	D51	1	Data number of the first equal value
D12	110		2		D52	8	Data number of the last equal value
D13	150		3		D53	7	Data number of the minimum value
D14	100		4	Equal	D54	9	Data number of the maximum value
D15	300		5				
D16	100		6	Equal			
D17	5		7	Minimum			
D18	100		8	Equal			
D19	500		9	Maximum			

Additional remark:

- If S1+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.
- If the operand n used in the 16-bit instruction 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.
- If the operand n used in the 32-bit instruction is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
- If the operand D used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of WORD/INT.
- If the operand D used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of DWORD/DINT.

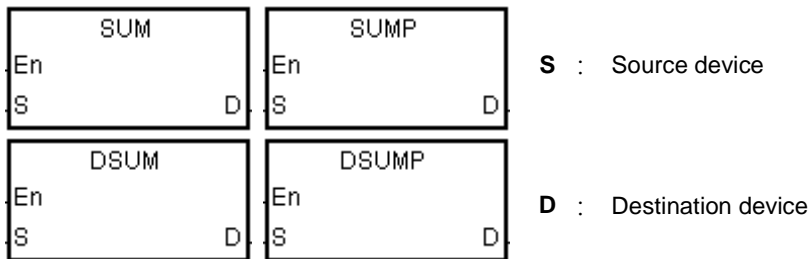
FB/FC	Instruction			Operand	Description
FC	D*	SUM	P	S, D	Number of bits whose states are ON

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3 Graphic expression:

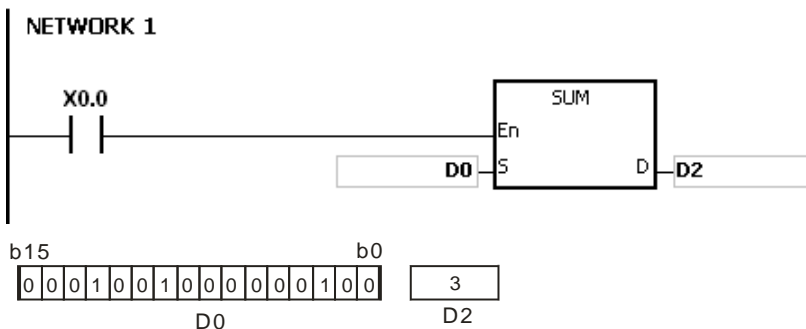


Explanation:

1. The number of bits whose values are 1 in S is stored in D.
2. When the values of 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.

Example:

When X0.0 is ON, the number of bits whose values are 1 in D0 is stored in D2.



Additional remark:

If the device exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

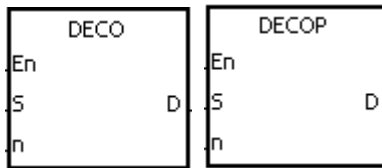
FB/FC	Instruction		Operand	Description
FC	DECO	P	S, D, n	Decoder

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●	●	●	●	●		●	●	●	●	○	●	○	○		
D	●	●	●	●	●	●		●	●	●	●	○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



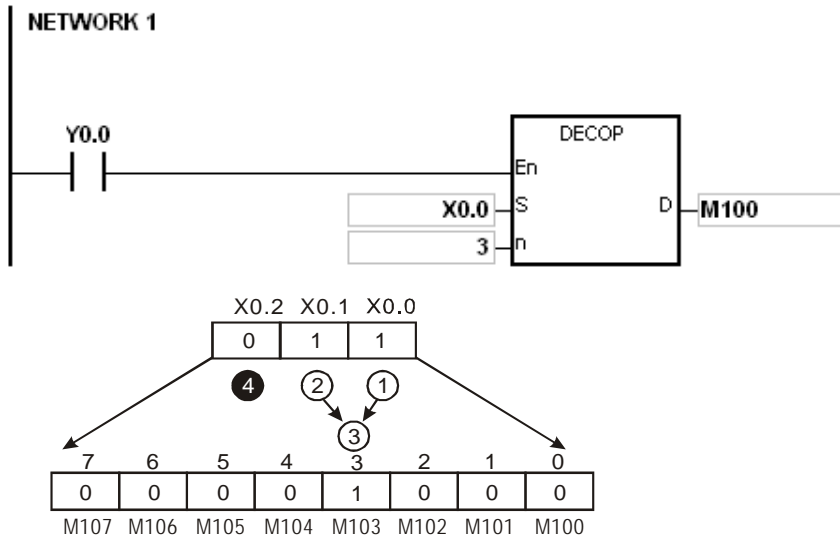
- S** : Source device
- D** : Device in which the decoded values are stored
- n** : Number of bits whose values are decoded

Explanation:

1. The values of the lower n bits in the source device specified by S are decoded as the values of the lower 2n bits in D.
2. When D is a bit device, n is within the range between 1 and 8. When n is 8, the values of the 8 bits is decoded as the values of the 256 bits. (Please note that the devices in which the decoded values are stored can not be used repeatedly.)
3. When D is a word device, n is within the range between 1 and 4. When n is 4, the values of the 4 bits is decoded as the values of the 16 bits.
4. Generally, the pulse instruction DECOP is used.

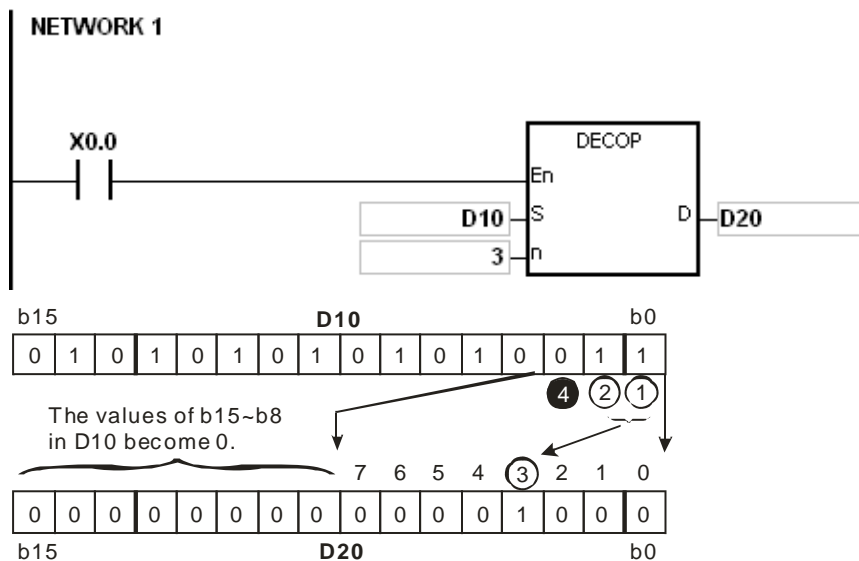
Example 1:

1. When Y0.0 is switched from OFF to ON, the instruction DECO decodes the values of the 3 bits in X0.0~X0.2 as the values of the 8 bits in M100~M107.
2. After the values of the 3 bits in X0.0~X0.2 are added up, the value 3 is gotten. The third bit in M10~M1007, that is, the bit in M103, is set to 1.
3. After the instruction DECO is executed and Y0.0 is switched OFF, the values of the 8 bits in M100~M107 are unchanged.



Example 2:

1. When X0.0 is switched from OFF to ON, the instruction DECO decodes the values of b2~b0 in D10 as the values of b7~b0 in D20, and the values of b15~b8 in D10 become 0.
2. The values of the lower 3 bits in D10 is decoded as the values of the lower 8 bits in D20. The values of the higher 8 bits are 0.
3. After the instruction DECO is executed and X0.0 is switched OFF, the data in D20 is unchanged.



Additional remark:

1. Suppose D is a bit device. If n is less than 1, or if n is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
2. Suppose D is a word device. If n is less than 1, or if n is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. Suppose S is a bit device. 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. Suppose D is a bit device. 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.

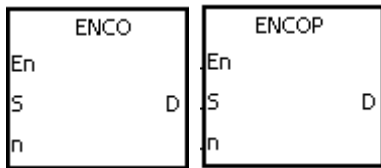
FB/FC	Instruction		Operand	Description
FC	ENCO	P	S, D, n	Encoder

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●	●	●	●	●		●	●	●	●	○	●				
D	●	●			●	●		●	●			○	●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S : Source device

D : Device in which the encoded values are stored

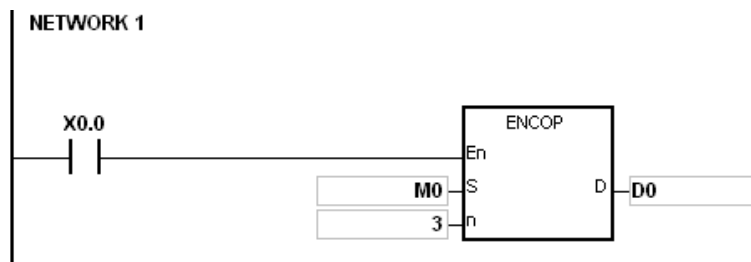
n : Number of bits whose values are encoded

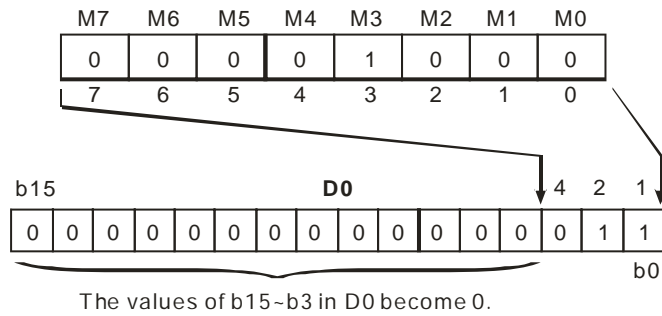
Explanation:

1. The values of the lower 2n bits in the source device specified by S are encoded as the values of the lower n bits in D.
2. If there are many bits whose values are 1 in the source device specified by S, the first bit with the value 1 from the left is processed.
3. When S is a bit device, n is within the range between 1 and 8. When n is 8, the values of the 256 bits is encoded as the values of the 8 bits.
4. When S is a word device, n is within the range between 1 and 4. When n is 4, the values of the 16 bits is encoded as the values of the 4 bits.
5. Generally, the pulse instruction ENCOP is used.

Example 1:

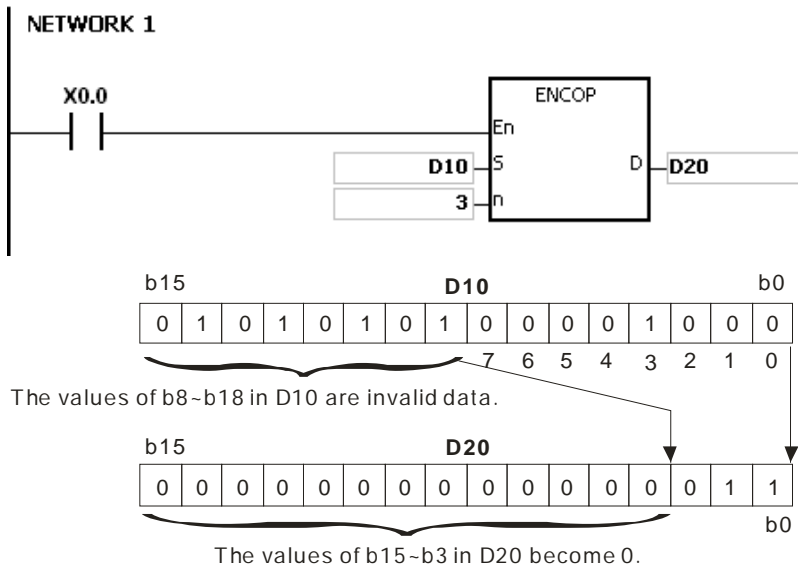
1. When X0.0 is switched from OFF to ON, the instruction ENCO encodes the values of the 8 bits in M0~M7 as the values of the lower 3 bits in D0, and the values of b15~b3 in D0 become 0.
2. After the instruction ENCO is executed and X0.0 is switched OFF, the data in D is unchanged.





Example 2:

1. When X0.0 is switched from OFF to ON, the instruction ENCO encodes the values of b0~b7 in D10 as the values of b2~b0 in D20, and the values of b15~b3 in D20 become 0. (The values of b8~b18 in D10 are invalid data.)
2. After the instruction ENCO is executed and X0.0 is switched OFF, the data in D is unchanged.



Additional remark:

1. If there is no bit whose value is 1 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. Suppose S is a bit device. If n is less than 1, or if n is larger than 8, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. Suppose S is a word device. If n is less than 1, or if n is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
4. Suppose S is a bit device. 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. Suppose D is a bit device. 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.

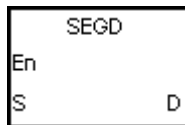
FB/FC	Instruction		Operand	Description
FC	SEGD	P	S, D	Seven-segment decoding

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S : Source device

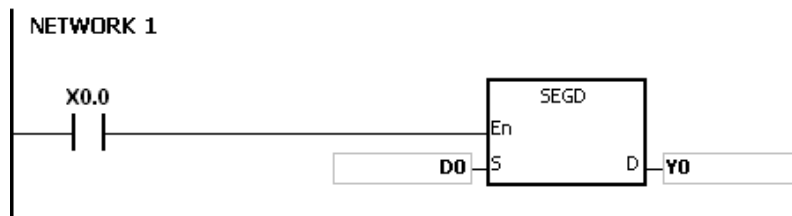
D : Device in which the seven-segment data is stored

Explanation:


The values of the lower 4 bits (b0~b3) in the source device specified by **S** are decoded as the seven-segment data stored in **D**.

Example:

When X0.0 is ON, the values of b0~b3 in D0 are decoded as the seven-segment data stored in Y0.0~Y0.15. If the data in the source device exceeds four bits, the values of the lower 4 bits are decoded.



The relation between the seven-segment data and the bit pattern of source data is presented in the following table.

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	0
1	0001		OFF	ON	ON	OFF	OFF	OFF	OFF	1
2	0010		ON	ON	OFF	ON	ON	OFF	ON	2
3	0011		ON	ON	ON	ON	OFF	OFF	ON	3
4	0100		OFF	ON	ON	OFF	OFF	ON	ON	4
5	0101		ON	OFF	ON	ON	OFF	ON	ON	5
6	0110		ON	OFF	ON	ON	ON	ON	ON	6
7	0111		ON	ON	ON	OFF	OFF	ON	OFF	7
8	1000		ON	ON	ON	ON	ON	ON	ON	8
9	1001		ON	ON	ON	ON	OFF	ON	ON	9
A	1010		ON	ON	ON	OFF	ON	ON	ON	A
B	1011		OFF	OFF	ON	ON	ON	ON	ON	b
C	1100		ON	OFF	OFF	ON	ON	ON	OFF	c
D	1101		OFF	ON	ON	ON	ON	OFF	ON	d
E	1110		ON	OFF	OFF	ON	ON	ON	ON	E
F	1111		ON	OFF	OFF	OFF	ON	ON	ON	F

3

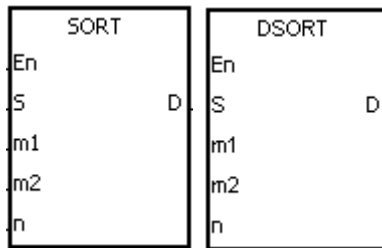
FB/FC	Instruction			Operand				Description				
FC	D*	SORT		S, m ₁ , m ₂ , D, n				Sorting the data				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
m ₁ , m ₂		●	●*				●	●*						
D		●	●*				●	●*						
n		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				
m ₁ , m ₂	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●				●				
n	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



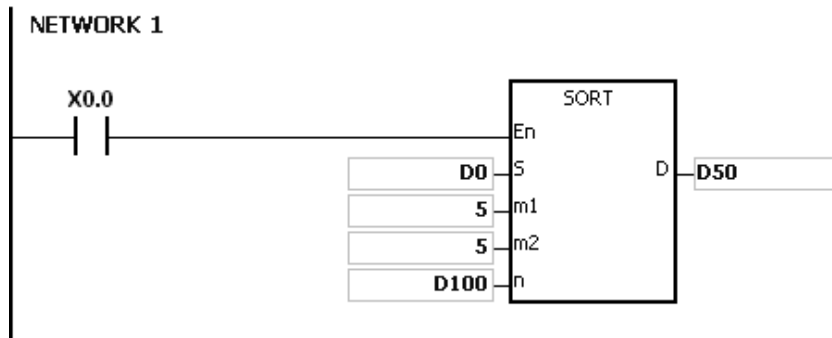
- S** : Initial device in which the original data is stored
- m₁** : Number of rows of data
- m₂** : Number of columns of data
- D** : Initial device in which the sorted data is stored
- n** : Reference value involved in the sorting of the data

Explanation:

1. The data which is sorted is stored in the m₁xm₂ 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 m₁ should be within the range between 1 and 32. The operand m₂ should be within the range between 1 and 6. The operand n should be within the range between 1 and m₂.
3. It is better that the rightmost number of the device number of the register specified by S is 0
4. If the value in n is altered during the execution of the instruction, the data is sorted according to the new value.
5. When SM604 is OFF, the data is sorted in ascending order. When SM604 is ON, the data is sorted in descending order.
6. To prevent the data from being sorted repeatedly, it is suggested that you use the pulse instruction.
7. Only the 32-bit instruction can use the 32-bit counter.

Example:

Suppose SM604 is OFF. When X0.0 is switched from OFF to ON, the data is sorted in ascending order. When the sorting of the data is complete, SM603 is ON.



1. The data which will be sorted is shown below.

		← m ₂ columns of data →				
		Column				
		1	2	3	4	5
Row	Column	Student number	Chinese	English	Math	Physics
↑ m ₁ rows of data ↓	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

2. When the value in D100 is 3, the data is sorted as follows.

		← m ₂ columns of data →				
		Column				
		1	2	3	4	5
Row	Column	Student number	Chinese	English	Math	Physics
↑ m ₁ rows of data ↓	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

3. When the value in D100 is 5, the data is sorted as follows.

		← m ₂ columns of data →				
		Column				
		1	2	3	4	5
Column \ Row	Row	Student number	Chinese	English	Math	Physics
↑ m ₁ rows of data ↓	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

Additional remark:

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 m₁, m₂, or n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	ZRST	P		D ₁ , D ₂	Resetting the zone

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
D ₁	●	●					●							
D ₂	●	●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D ₁	●	●	●	●	●	●	●	●	●	●		○	●				
D ₂	●	●	●	●	●	●	●	●	●	●		○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3 Graphic expression:



D₁ : Initial device which is reset

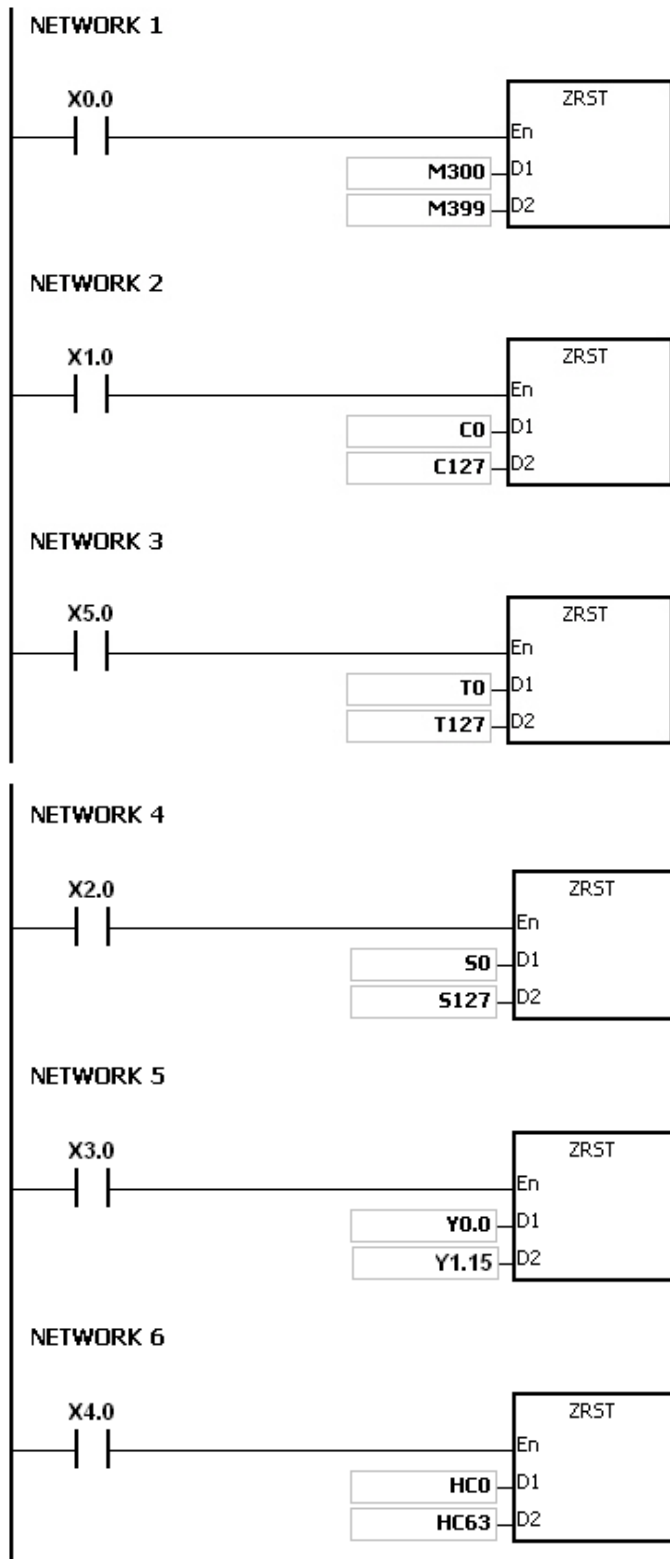
D₂ : Final device which is reset

Explanation:

1. When the instruction is executed, the values in D1~D2 are cleared.
2. When the device number of D1 is larger than the device number of D2, only D2 is reset.

Example:

1. When X0.0 is ON, the auxiliary relays M300~M399 are reset to OFF.
2. When X1.0 is ON, the 16-bit counters C0~C127 are reset. (The values of C0~C127 are cleared to 0, and the contact and the coil are reset to OFF.)
3. When X5.0 is ON, the timers T0~T127 are reset. (The values of T0~T127 are cleared to 0. and the contact and the coil are reset to OFF.)
4. When X2.0 is ON, the stepping relays S0~S127 are reset to OFF.
5. When X3.0 is ON, the output relays Y0.0~Y1.15 are reset to OFF.
6. When X4.0 is ON, the 32-bit counters HC0~HC63 are reset. (The values of HC0~HC63 are cleared to 0, and the contact and the coil are reset to OFF.)

**Additional remark:**

1. If D1 and D2 are different types of devices, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.
2. If D1 and D2 contain different formats of data, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2007.

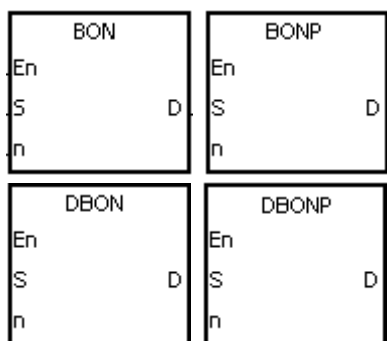
FB/FC	Instruction			Operand	Description
FC	D*	BON	P	S, D, n	Checking the state of the bit

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●	●	●	●	●		●	●	●			●				
n	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S : Source device

D : Device in which the check result is stored

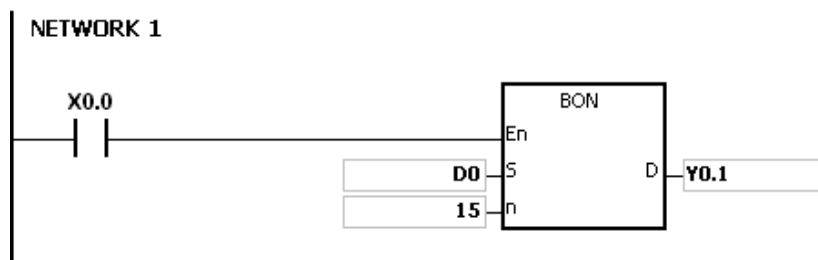
n : Bit whose state is judged

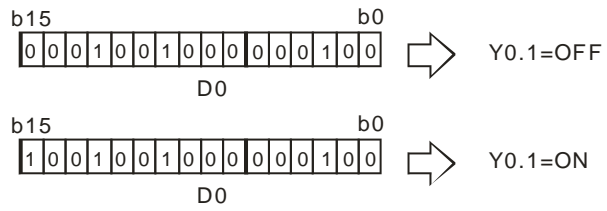
Explanation:

1. The state of the nth bit in S is checked, and the result is stored in D.
2. The operand n used in the 16-bit instruction should be within the range between 0 and 15, and the operand n used in the 32-bit instruction should be within the range between 0 and 31.
3. Only the 32-bit instructions can use the 32-bit counter.

Example:

1. When X0.0 is ON, Y0.1 is ON if the value of the 15th bit in D0 is 1. When X0.0 is ON, Y0.1 is OFF if the value of the 15th bit in D0 is 0.
2. When X0.0 is switched OFF, the state of Y0.1 remains the same as that before X0.0's being OFF.



**Additional remark:**

If **n** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

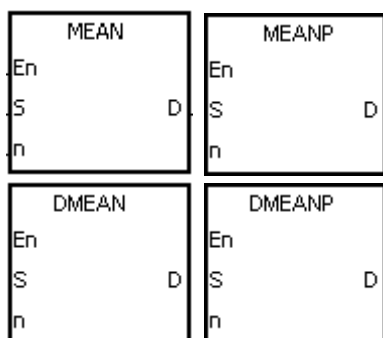
FB/FC	Instruction			Operand	Description
FC	D*	MEAN	P	S, D, n	Mean

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●		●	○	●				
n	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S : Initial device

D : Device in which the mean is stored

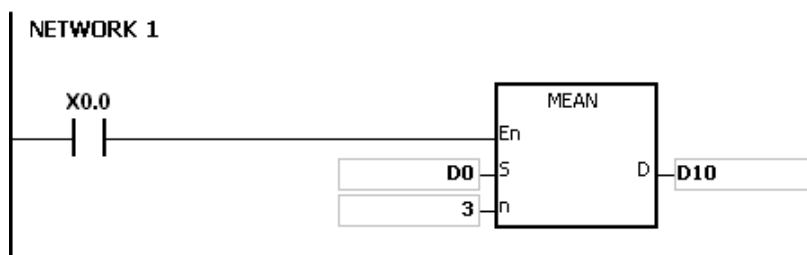
n : Number of devices

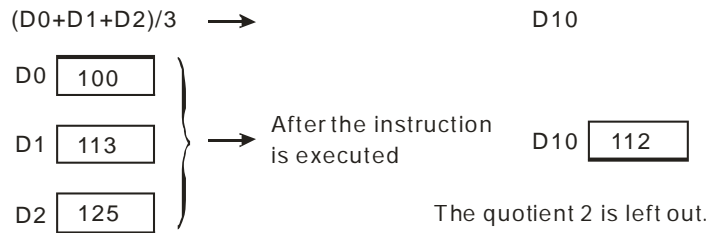
Explanation:

1. After the values in the n devices starting from the device specified by S are added up, the mean of the sum is stored in D.
2. If a remainder appears in the calculation, it is left out.
3. The operand n used in the 16-bit instruction should be within the range between 1 and 256, and the operand n used in the 32-bit instruction should be within the range between 1 and 128.
4. Only the 32-bit instructions can use the 32-bit counter.

Example:

When X0.0 is ON, the values in the three registers starting from D0 are added up. After the values are added up, the sum is divided by 3. The quotient is stored in D10, and the remainder is left out.



**Additional remark:**

1. If the operand n used in the 16-bit instruction 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 operand n used in the 32-bit instruction is less than 1 or larger than 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.

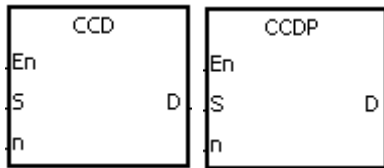
FB/FC	Instruction			Operand	Description
FC	CCD	P		S, D, n	Sum check

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



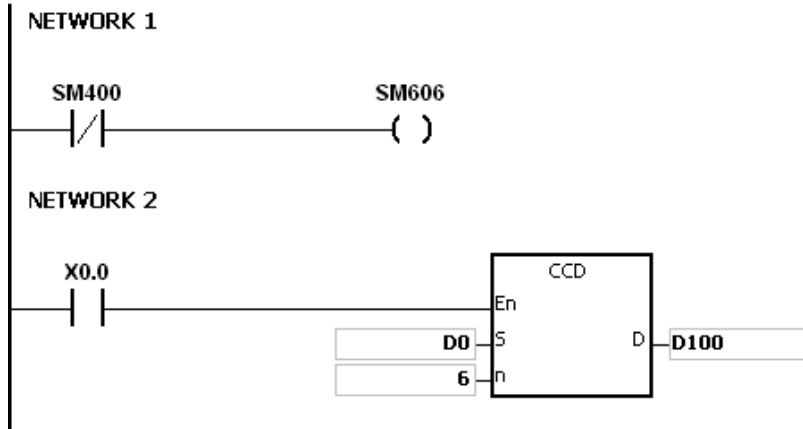
- S : Initial device
- D : Device in which the sum is stored
- n : Number of pieces of data

Explanation:

- In communication, the sum check is used to compare checksums on the same data on different occasions or on different representations of the data in order to verify the data integrity.
- The 16-bit conversion mode: When SM606 is OFF, the working mode of the instruction is the 16-bit conversion mode. The n pieces of data in the registers starting from the register specified by S (eight bits as a group) are added up. The sum is stored in the register specified by D, and the values of the parity bits are stored in D+1.
- The 8-bit conversion mode: When SM606 is ON, the working mode of the instruction is the 8-bit conversion mode. The n pieces of data in the registers starting from the register specified by S (Eight bits forms a group, and only low eight bits are valid.) are added up. The sum is stored in the register specified by D, and the values of the parity bits are stored in D+1.
- The operand n should be within the range between 1 and 256.

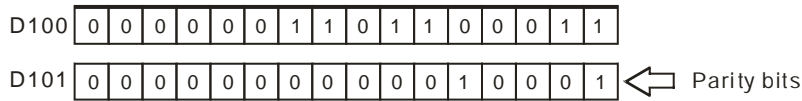
Example 1:

- When SM606 is OFF, the working mode of the instruction is the 16-bit conversion mode.
- When X0.0 is ON, the six pieces of data in D0~D2 (eight bits as a group) are added up. The sum is stored in D100, and the values of the parity bits are stored in D101.



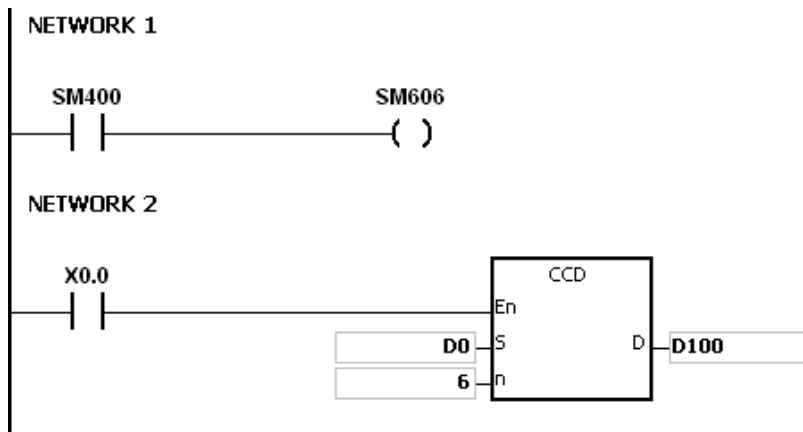
S	Data
D0 Low	100 = 0 1 1 0 0 1 0 0
D0 High	111 = 0 1 1 0 1 1 1 ① ←
D1 Low	120 = 0 1 1 1 1 0 0 0
D1 High	202 = 1 1 0 0 1 0 1 0
D2 Low	123 = 0 1 1 1 1 0 1 ① ←
D2 High	211 = 1 1 0 1 0 0 1 ① ←
D100	867
D101	0 0 0 1 0 0 0 ①

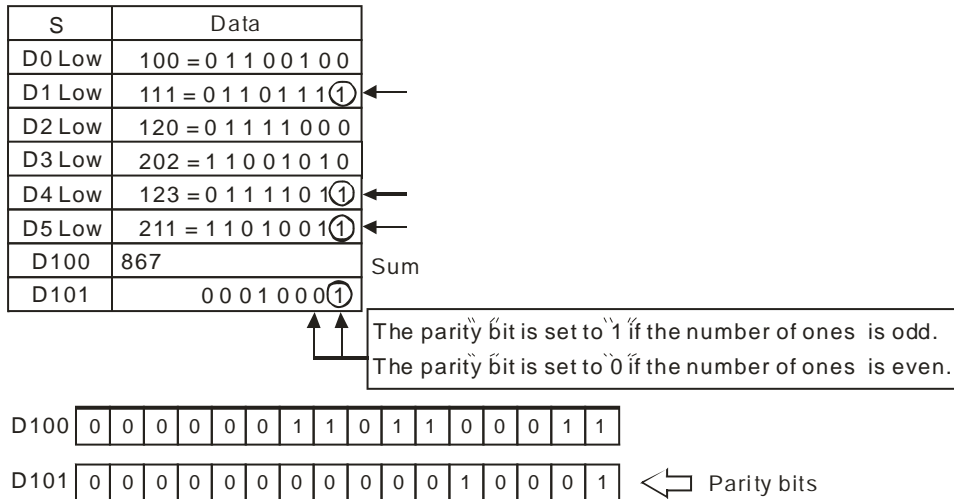
The parity bit is set to 1 if the number of ones is odd.
The parity bit is set to 0 if the number of ones is even.



Example 2:

1. When SM606 is ON, the working mode of the instruction is the 8-bit conversion mode.
2. When X0.0 is ON, the six pieces of data in D0~D5 (eight bits as a group) are added up. The sum is stored in D100, and the values of the parity bits are stored in D101.





Additional remark:

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 less than 1, or if n is larger than 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 will be ARRAY [2] of WORD/INT.

FB/FC	Instruction			Operand	Description
FC	D*	ABS	P	D	Absolute value

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●	●	●	●		●	○					

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



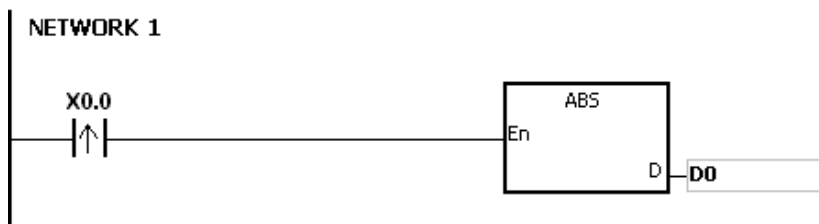
D : Device involved in the getting of the absolute value

Explanation:

1. When the instruction ABS is executed, the absolute value of the value in the device specified by D is gotten.
2. Generally, the pulse instruction ABSP is used.
3. Only the 32-bit instructions can use the 32-bit counter.

Example:

Suppose the value in D0 before the execution of the instruction is -1234. When X0.0 is switched from OFF to ON, the absolute value of -1234 in D0 is gotten. That is, the value in D0 becomes 1234 after the instruction is executed.



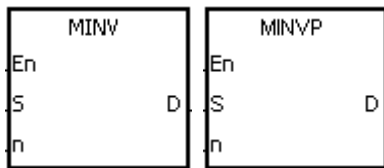
FB/FC	Instruction			Operand			Description		
FC	MINV	P		S, D, n			Inverting the matrix bits		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●				●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



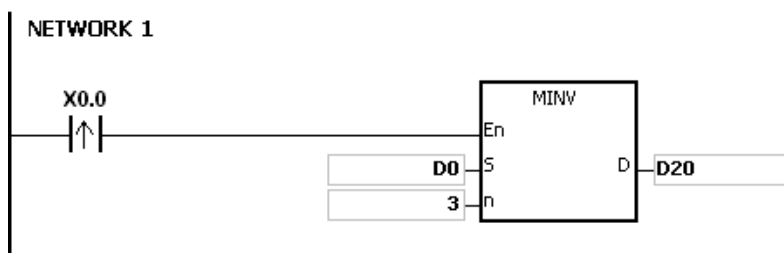
- S** : =Matrix source
- D** : Operation result
- n** : Length of the array

Explanation:

- The bits in the n devices starting from the device specified by S are inverted, and the inversion result is stored in D.
- The operand n should be within the range between 1 and 256.

Example:

When X0.0 is ON, the bits in the three 16-bit registers D0~D2 are inverted, and the inversion result is stored in the 16-bit registers D20~D22.



	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

↓ After the instruction is executed

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D20	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D21	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D22	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Additional remark:

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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	MBRD	P		S, D, n	Reading the matrix bit

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



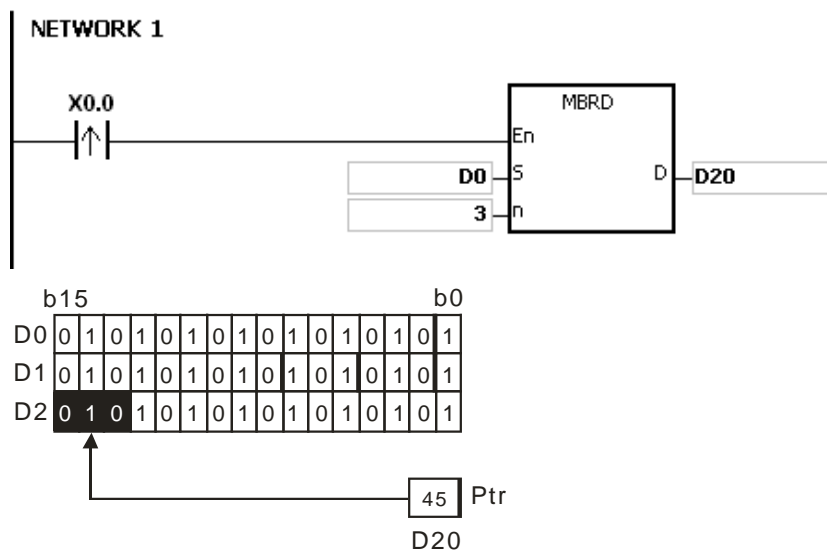
- S** : Matrix source
- D** : Pointer
- n** : Length of the array

Explanation:

- When the instruction is executed, the state of SM613 is checked. If SM613 is ON, the value of the pointer D is cleared to 0. The value of the bit specified by the value of the pointer D is read into SM614. After the value of the bit is read, the state of SM612 is checked. If SM612 is ON, the value of the pointer D increases by one.
- When the value of the last bit is read, SM608 is ON, and the bit number is recorded in the pointer D.
- The operand n should be within the range between 1 and 256.
- The value of the pointer is specified by users. The values range from 0 to 16n-1, and correspond to the range from b0 to b16n-1. If the value of the pointer exceeds the range, SM611 is set to 1, and the instruction is not executed.

Example:

- Suppose SM613 is OFF and SM612 is ON when X0.0 is switched from OFF to ON.
- Suppose the current value in D20 is 45. When X0.0 is switched from OFF to ON three times, you can get the following execution results.
 - ① The value in D20 is 46, SM614 is OFF, and SM608 is OFF.
 - ② The value in D20 is 47, SM614 is ON, and SM608 is OFF.
 - ③ The value in D20 is 47, SM614 is OFF, and SM608 is ON.

**Additional remark:**

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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. The flags:
 - SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.
 - SM611: It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.
 - SM612: It is the matrix pointer increasing flag. The current value of the pointer increases by one.
 - SM613: It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.
 - SM614: It is the carry flag for the matrix rotation/shift/output.

FB/FC	Instruction			Operand			Description		
FC	MBWR	P		S, D, n			Writing the matrix bit		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●				●				
D	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



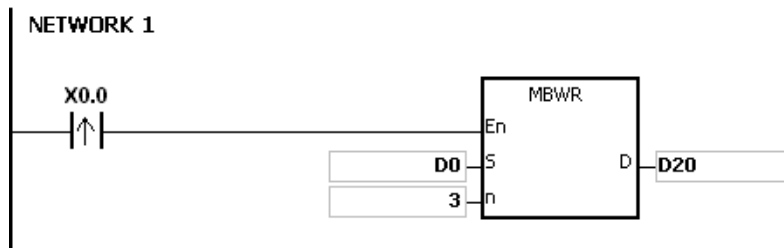
- S** : Matrix source
- D** : Pointer
- n** : Length of the array

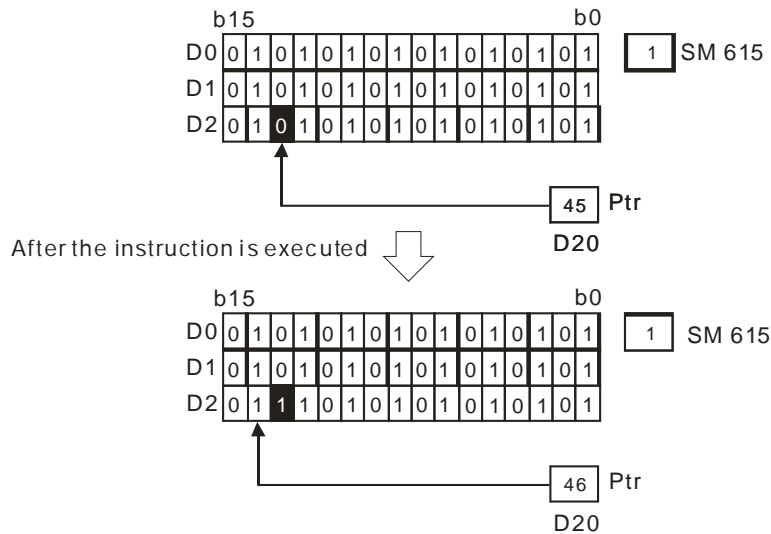
Explanation:

- When the instruction is executed, the state of SM613 is checked. If SM613 is ON, the value of the pointer D is cleared to 0. The state of SM615 is written into the bit specified by the value of the pointer D. After the state of SM615 is written into the bit, the state of SM612 is checked. If SM612 is ON, the value in the pointer D increases by one.
- When the state of SM615 is written into the last bit, SM608 is ON, and the bit number is recorded in the pointer D. If value of the pointer D exceeds the range, SM611 is ON.
- The operand n should be within the range between 1 and 256.
- The value of the pointer is specified by users. The values range from 0 to 16n-1, and correspond to the range from b0 to b16n-1. If the value of the pointer exceeds the range, SM611 is set to 1, and the instruction is not executed.

Example:

- Suppose SM613 is OFF and SM612 is ON when X0.0 is switched from OFF to ON.
- Suppose the current value in D20 is 45. When X0.0 is switched from OFF to ON one time, you can get the execution result shown below. When the value in D20 is 45, SM615 is OFF, and SM608 is OFF.



**Additional remark:**

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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. The flags:
 - SM608: The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.
 - SM611: It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.
 - SM612: It is the matrix pointer increasing flag. The current value of the pointer increases by one.
 - SM613: It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.
 - SM615: It is the borrow flag for the matrix shift/output.

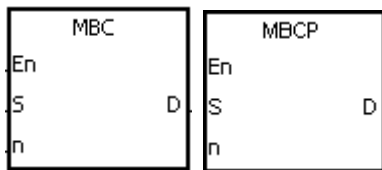
FB/FC	Instruction			Operand			Description									
FC		MBC	P	S, D, n			Counting the bits with the value 0 or 1									

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



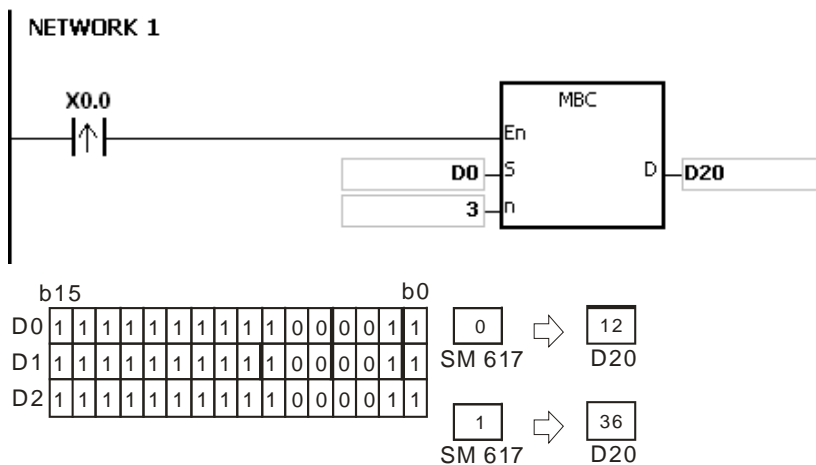
- S** : Matrix source
- D** : Operation result
- n** : Length of the array

Explanation:

- The instruction is used to count the bits with the value 1 or 0 in the n devices starting from the device specified by S. The operation result is stored in D.
- When SM617 is ON, the bits with the value 1 is counted. When SM617 is OFF, the bits with the value 0 is counted. When the operation result is 0, SM618 is ON.
- The operand n should be within the range between 1 and 256.

Example:

Suppose SM617 is ON. When X0.0 is ON, the bits with the value 1 are counted, and the results are stored in D20.
 Suppose SM617 is OFF. When X0.0 is ON, the bits with the value 0 are counted, and the results are stored in D20.



Additional remark:

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 less than 1, or if n is larger than 256, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. The flags:
 - SM617: The bits with the value 0 or 1 are counted.
 - SM618: It is ON when the matrix counting result is 0.

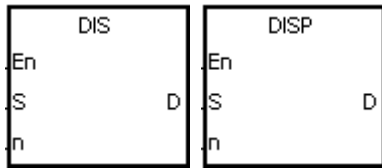
FB/FC	Instruction			Operand	Description
FC	DIS	P		S, D, n	Disuniting the 16-bit data

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●				●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

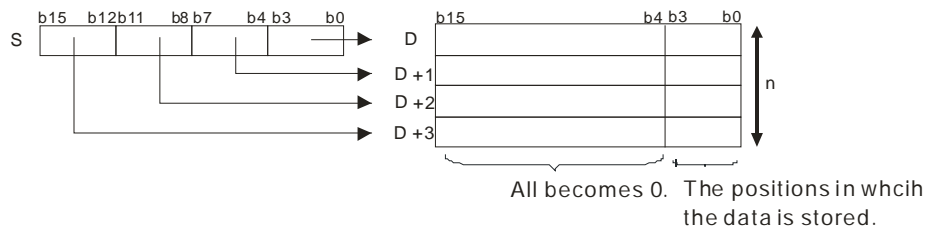
Graphic expression:



- S** : Data source
- n** : Number of devices
- D** : Operation result

Explanation:

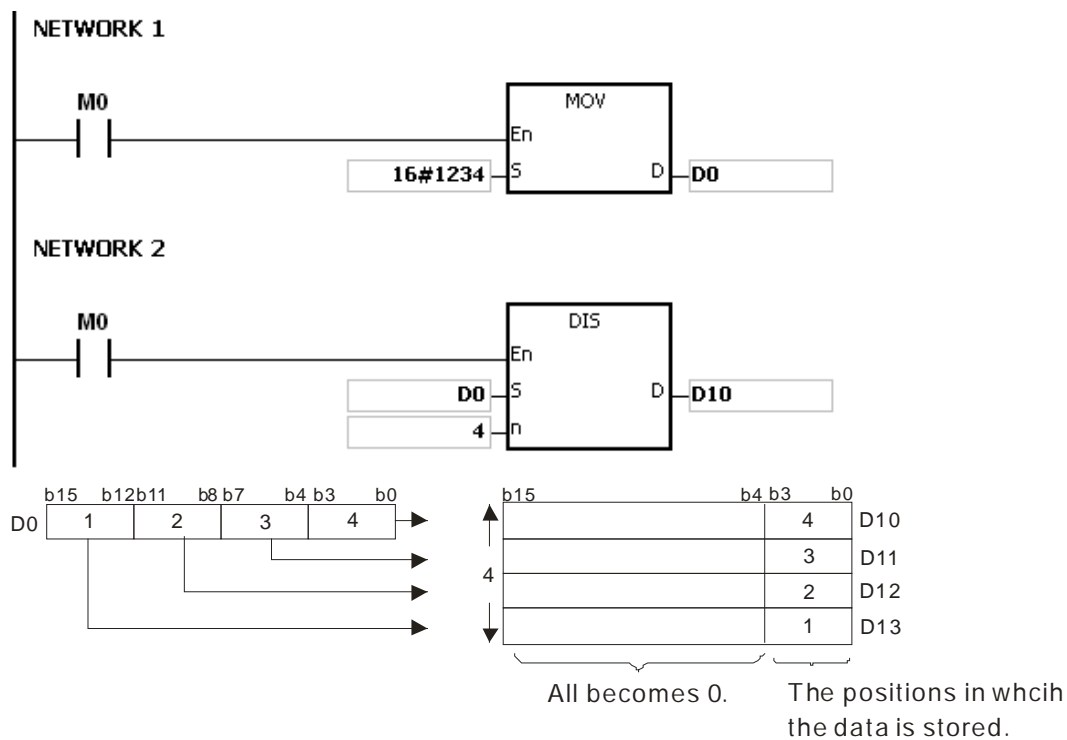
- The 16-bit value in the register specified by **S** is divided into four groups (four bits as a group), and these groups are stored in the low four bits in every register (The registers range from **D** to **D+(n-1)**).



- The operand **n** should be within the range between 1 and 4.

Example:

Suppose the value in D0 is 16#1234. When M0 is enabled, the instruction DIS is executed. The value in D0 is divided into four groups (four bits as a group), and these groups are stored in the low four bits in every register (The registers range from D10 to D13.).



Additional remark:

1. If $D \sim D+(n-1)$ 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 less than 1, or if n is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

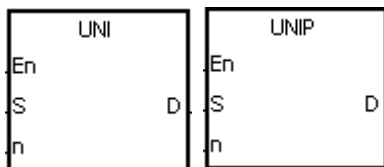
FB/FC	Instruction			Operand	Description
FC	UNI	P		S, D, n	Uniting the 16-bit data

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●		●		●				
n	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

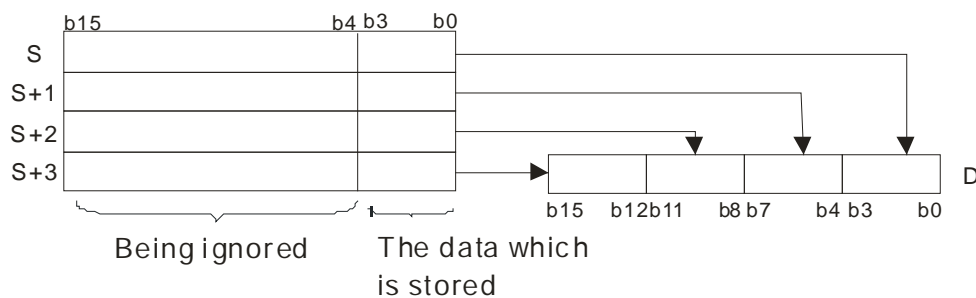
Graphic expression:



- S : Data source
- n : Data length
- D : Operation result

Explanation:

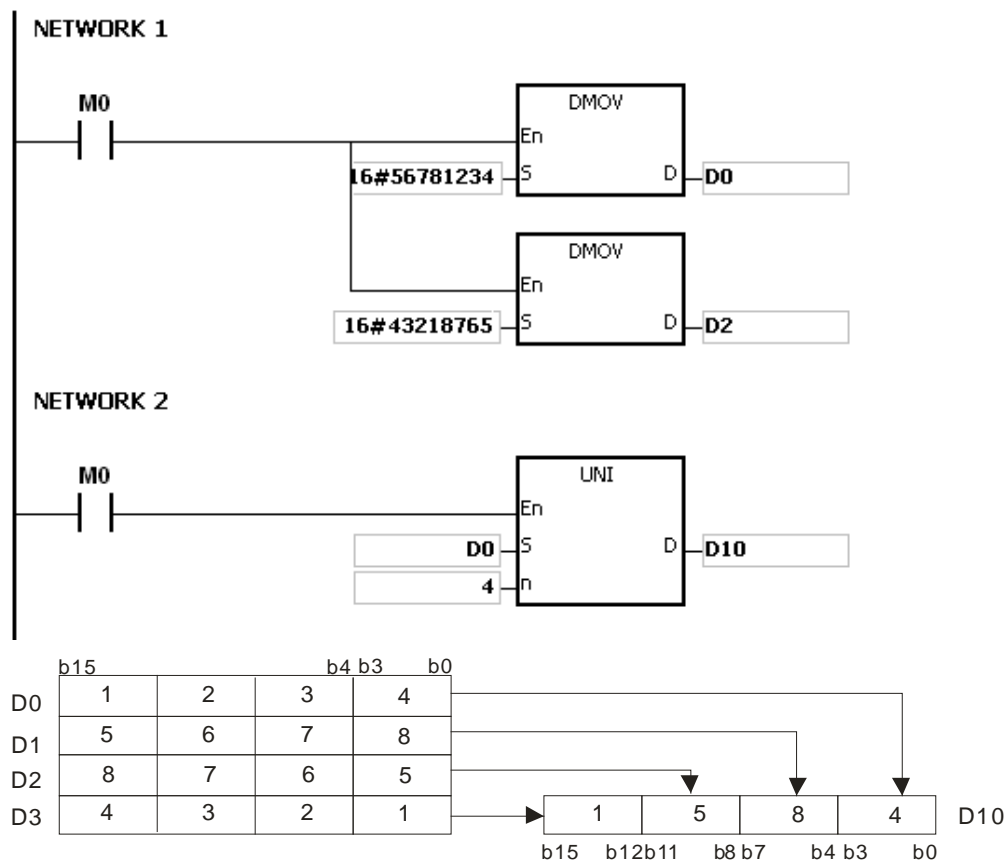
- The 16-bit values in the registers specified by S~S+(n-1) are divided into groups (four bits as a group), and every group which is composed of b0~b3 is stored in the register specified by D.



- The operand n should be within the range between 1 and 4.

Example:

Suppose the values in D0~D3 are 16#1234, 16#5678, 16#8765, and 16#4321 respectively. When M0 is enabled, the instruction UNI is executed. The values in D0~D3 are divided into groups (four bits as a group), and every group which is composed of b0~b3 is stored in D10.



3

Additional remark:

1. If S~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 is less than 1, or if n is larger than 4, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

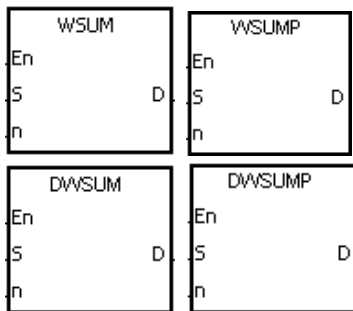
FB/FC	Instruction			Operand	Description
FC	D*	WSUM	P	S, D, n	Getting the sum

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●		●				
D	●	●			●	●	●	●	●		●		●				
n	●	●			●	●	●	●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

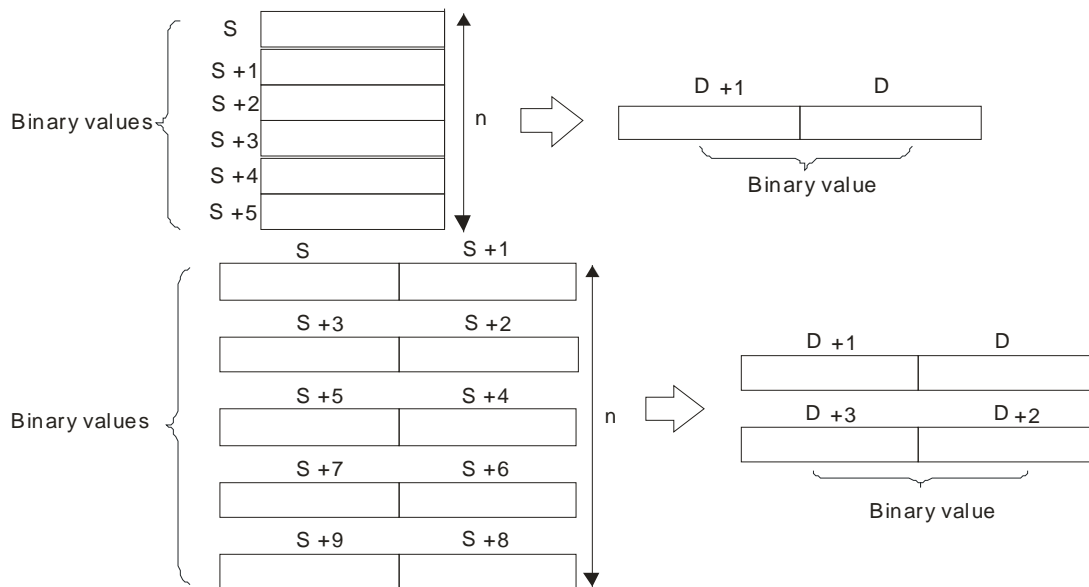
Graphic expression:



S : Data source
 n : Data length
 D : Operation result

Explanation:

- The signed decimal values in **S~S+n-1** are added up, and the sum is stored in the register specified by **D**.



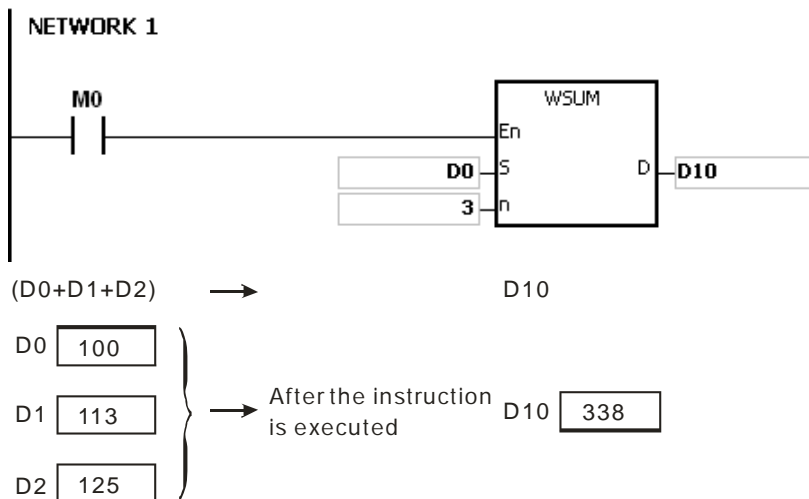
- The operand **n** used in the 16-bit instruction should be within the range between 1 and 256, and the operand

n used in the 32-bit instruction should be within the range between 1 and 128.

- Only the 32-bit instructions can use the 32-bit counter.

Example:

When the instruction WSUM is executed, the values in D0~D2 are added up, and the sum is stored in D10.



Additional remark:

- If n used in the 16-bit instruction 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.
- If n used in the 32-bit instruction is less than 1 or larger than 128, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
- 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.

FB/FC	Instruction			Operand	Description
FC	BSET	P		S, D, n	Setting the bit in the word device to ON

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:

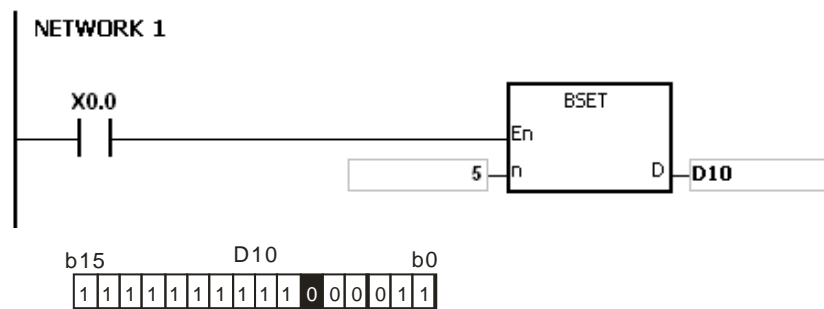


Explanation:

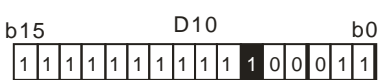
1. The instruction is used to set the nth bit in the register specified by D to 1.
2. When the instruction BSET is driven, the specified bit is set to ON. No matter the instruction BSET is still driven or not, the bit keeps ON. You can use the instruction BRST to set the bit OFF.
3. The operand n should be within the range between 0 and 15.

Example:

When X0.0 is ON, the fifth bit in D10 is set to 1.



After the instruction is executed



Additional remark:

If n is less than 0, or if n is larger than 15, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

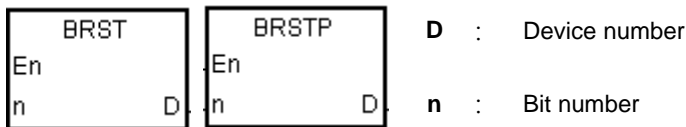
FB/FC	Instruction		Operand		Description
FC	BRST	P	D, n		Resetting the bit in the word device

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●		●	○	●				
n	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

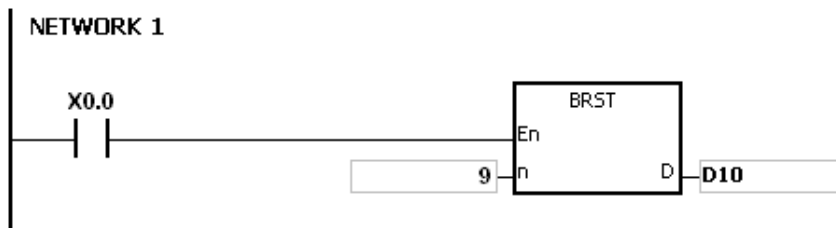


Explanation:

1. The instruction is used to set the nth bit in the register specified by D to 0.
2. When the instruction BRST is driven, the specified bit is set to OFF.
3. The operand n should be within the range between 0 and 15.

Example:

When X0.0 is ON, the ninth bit in D10 is set to 0.



After the instruction is executed



Additional remark:

If n is less than 0, or if n is larger than 15, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	BKRST	P		D, n	Resetting the specified zone

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●			○	●				
n	●	●	●	●	●	●	●	●	●	●	●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:



D : Device number

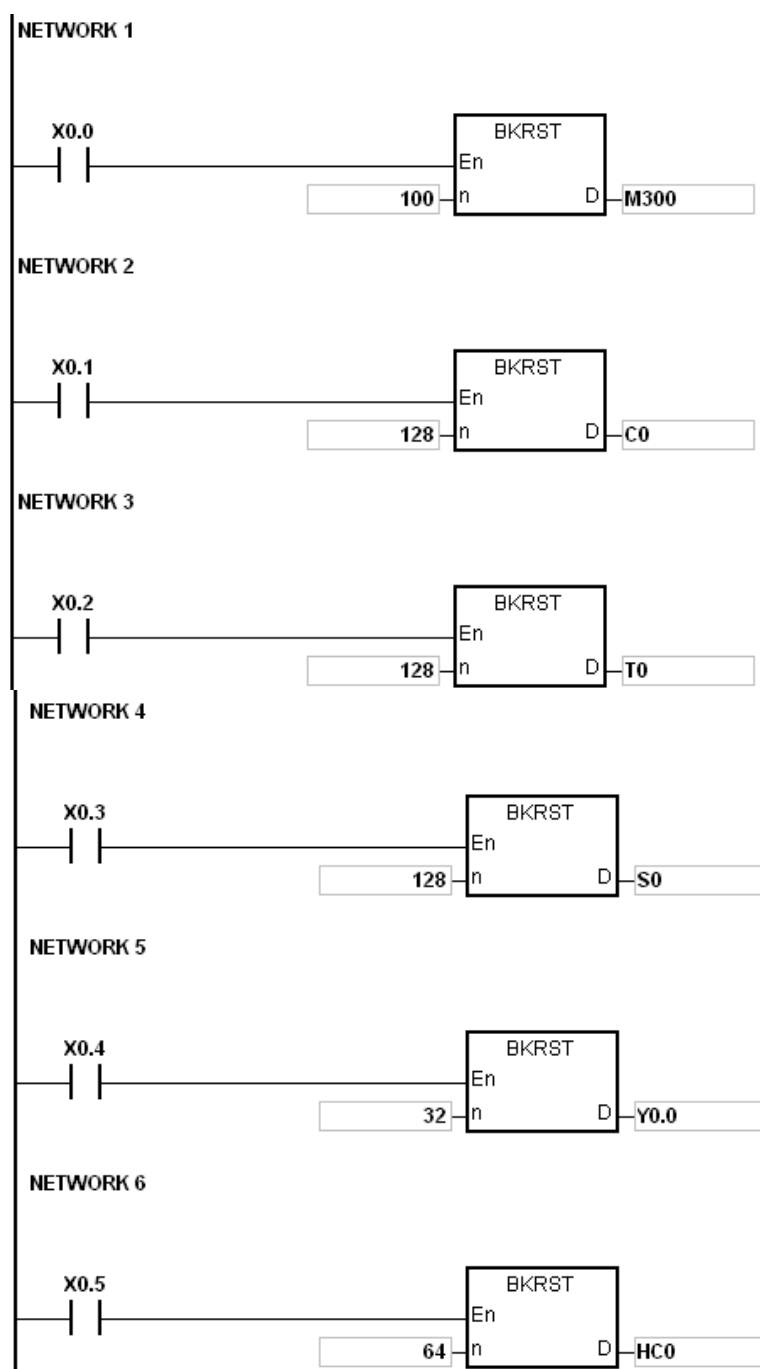
n : Length

Explanation:

1. The instruction is used to clear the values in D~D+(n-1).
2. The operand n should be within the range between 1 and 1024.

Example:

1. When X0.0 is ON, the auxiliary relays M300~M399 are reset to OFF.
2. When X0.1 is ON, the counters C0~C127 are reset. (The values of C0~C127 are cleared to 0, and the contact and the coil are reset to OFF.)
3. When X0.2 is ON, the timers T0~T127 are reset. (The values of T0~T127 are cleared to 0. and the contact and the coil are reset to OFF.)
4. When X0.3 is ON, the stepping relays S0~S127 are reset to OFF.
5. When X0.4 is ON, the output relays Y0.0~Y1.15 are reset to OFF.
6. When X0.5 is ON, the counters HC0~HC63 are reset. (The values of HC0~HC63 are cleared to 0, and the contact and the coil are reset to OFF.)

**Additional remark:**

1. If $D \sim D+(n-1)$ 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 less than 0, or if n is larger than 1024, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.

FB/FC	Instruction			Operand	Description
FC	D*	LIMIT	P	S ₁ , S ₂ , S ₃ , D	Confining the value within the bounds

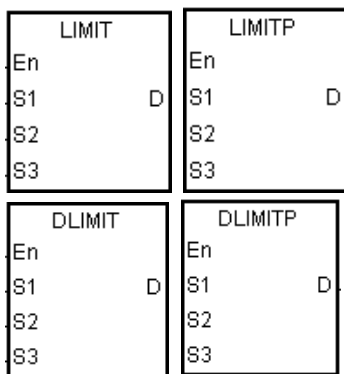
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂ , S ₃		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂ , S ₃	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:



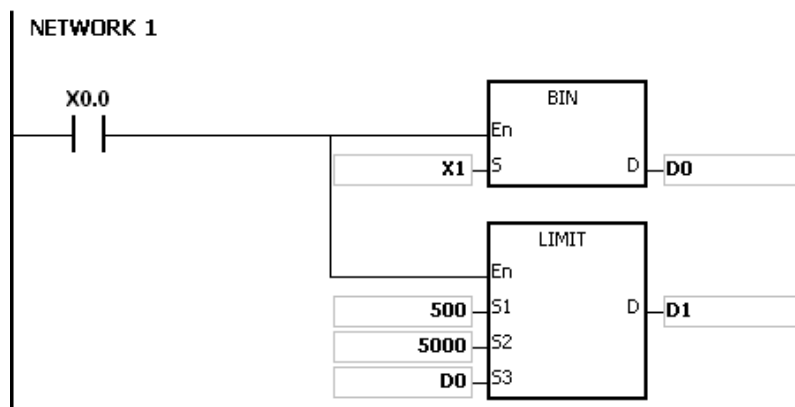
- S₁ : Minimum output value
- S₂ : Maximum output value
- S₃ : Input value
- D : Output value

Explanation:

1. The input value in S₃ is compared with the minimum output value in S₁ and the maximum output value in S₂, and the comparison result is stored in D.
 - If the minimum output value in S₁ is larger than the input value in S₃, the output value stored in D is equal to the minimum output value in S₁.
 - If the maximum output value in S₂ is less than the input value in S₃, the output value stored in D is equal to the maximum output value in S₂.
 - If the input value in S₃ is within the range between the minimum output value in S₁ and the maximum output value in S₂, the output value stored in D is equal to the input value in S₃.
 - If the minimum output value in S₁ is larger than the maximum output value in S₂, the instruction is not executed.
2. Only the 32-bit instructions can use the 32-bit counter.

Example:

When X0.0 is ON, the state of X1 is converted into the binary value, and the conversion result is stored in D0. Besides, the value stored in D0 is compared with 500 and 5000, and the comparison result is stored 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	$500 \leq D0 \leq 5000$	600

Additional remark:

If the minimum output value in **S₁** is larger than the maximum output value in **S₂**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction			Operand	Description
FC	D*	BAND	P	S₁, S₂, S₃, D	Deadband control

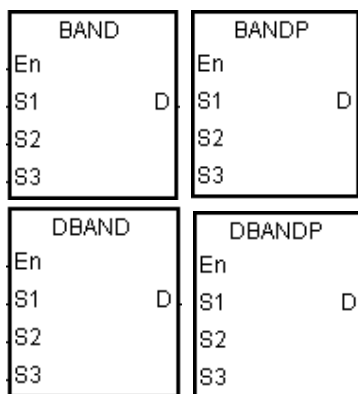
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂, S₃		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂, S₃	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:



S₁ : Minimum value of the deadband

S₂ : Maximum value of the deadband

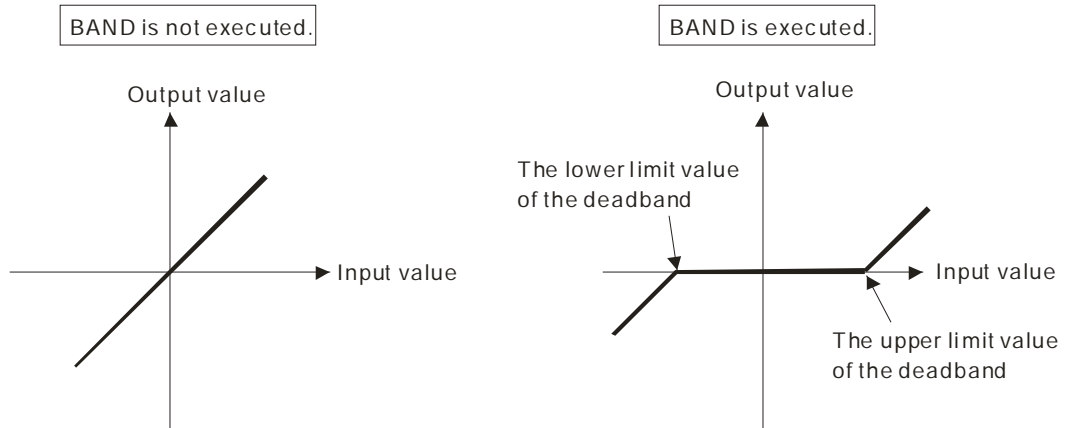
S₃ : Input value

D : Output value

Explanation:

- The minimum value of the deadband in **S₁** or the maximum value of the deadband in **S₂** is subtracted from the input value in **S₃**, and the difference is stored in **D**.
 - If the minimum value of the deadband in **S₁** is larger than the input value in **S₃**, the minimum value of the deadband in **S₁** is subtracted from the input value in **S₃**, and the difference is stored in **D**.
 - If the maximum value of the deadband in **S₂** is less than the input value in **S₃**, the maximum value of the deadband in **S₂** is subtracted from the input value in **S₃**, and the difference is stored in **D**.
 - If the input value in **S₃** is within the range between the minimum of the deadband in **S₁** and the maximum value of the deadband in **S₂**, the output value stored in **D** is 0.
 - If the minimum value of the deadband in **S₁** is larger than the maximum value of the deadband in **S₂**, the instruction is not executed.
- Only the 32-bit instructions can use the 32-bit counter.

3. The figures:



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 should be within the range described below.

- If the instruction BAND is executed, 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 is within the range between -32768 and 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 output value in D is calculated as follows.

$$\text{Output value in } D = -32768 - 10 = 16\#8000 - 16\#000A = 16\#7FF6 = 32758$$

- If the instruction DBAND is executed, 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 is within the range between -2147483648 and 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 output value in $(D+1, D)$ is calculated as follows.

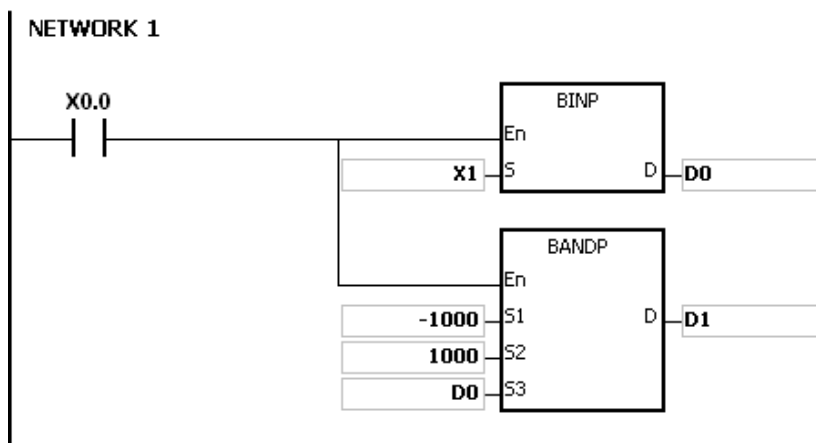
$$\text{Output value in } (D+1, D)$$

$$= -2147483648 - 1000 = 16\#80000000 - 16\#000003E8 = 16\#7FFFC18$$

$$= 2147482648$$

Example 1:

When X0.0 is ON, -1000 or 1000 is subtracted from the binary-coded decimal value in X1, and the difference is stored in D1.

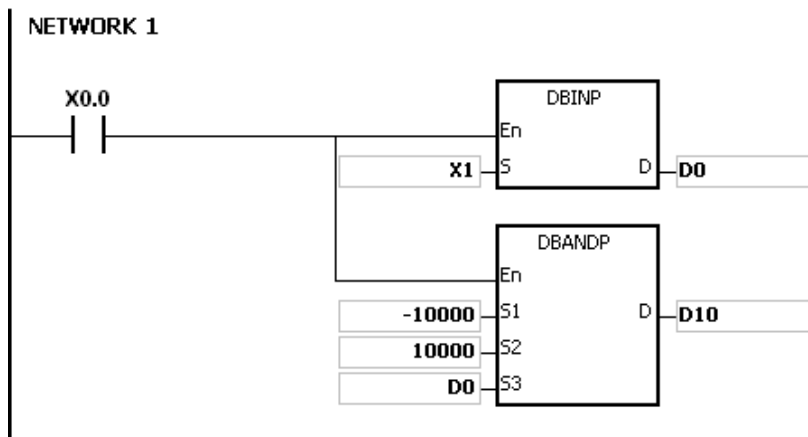


The execution results:

Minimum value of the deadband	Maximum value of the deadband	Input value in D0	Function	Output value in D1
-1000	1000	-1200	$D0 < -1000 \Rightarrow D1 = D0 - (-1000)$	-200
		1200	$D0 > 1000 \Rightarrow D1 = D0 - 1000$	200
		500	$-1000 \leq D0 \leq 1000 \Rightarrow D0 = 0$	0

Example 2:

When X0.0 is ON, -10000 or 10000 is subtracted from the binary-coded decimal value in (X2, X1), and the difference is stored in (D11, D10).



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	$(D1, D0) < -10000 \Rightarrow (D11, D10) = (D1, D0) - (-10000)$	-2000
		12000	$(D1, D0) > 10000 \Rightarrow (D11, D10) = (D1, D0) - 10000$	2000
		5000	$-10000 \leq (D1, D0) \leq 10000 \Rightarrow (D1, D0) = 0$	0

Additional remark:

If the minimum value of the deadband in S₁ is larger than the maximum value of the deadband in S₂, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction			Operand	Description
FC	D*	ZONE	P	S ₁ , S ₂ , S ₃ , D	Controlling the zone

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂ , S ₃		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂ , S ₃	●	●			●	●	●	●	●			○	●	○	○		
D	●	●			●	●	●	●	●			○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



S₁ : Negative deviation

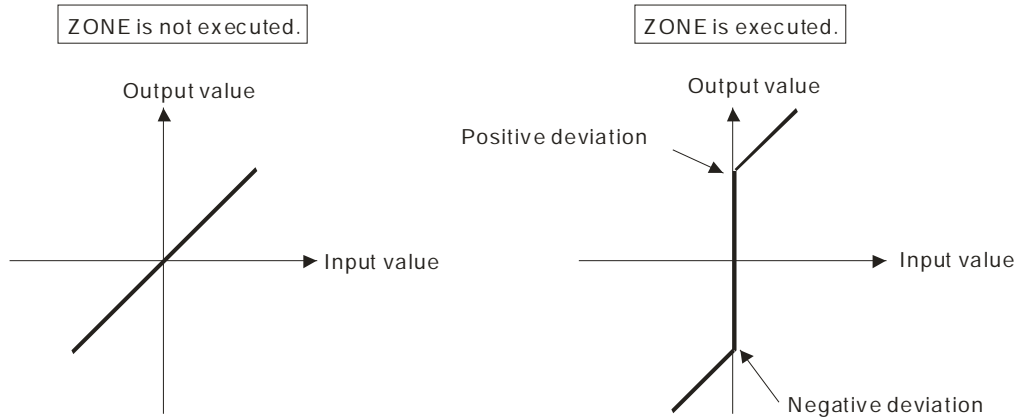
S₂ : Positive deviation

S₃ : Input value

D : Output value

Explanation:

- The negative deviation in S₁ or the positive deviation in S₂ is added to the input value in S₃, and the sum is stored in D.
 - If the input value in S₃ is less than 0, the negative deviation in S₁ is added to the input value in S₃, and the sum is stored in D.
 - If the input value in S₃ is larger than 0, the positive deviation in S₂ is added to the input value in S₃, and the sum is stored in D.
 - If the input value in S₃ is equal to 0, the output value stored in D is 0.
- The figures:



3. Only the 32-bit instructions can use the 32-bit counter.
4. The negative deviation in **S₁**, the positive deviation in **S₂**, the input value in **S₃**, and the output value in **D** should be within the range described below.

- If the instruction ZONE is executed, the negative deviation in **S₁**, the positive deviation in **S₂**, the input value in **S₃**, and the output value in **D** is within the range between -32768 and 32767. Suppose the negative deviation in **S₁** is -100 and the input value in **S₃** is -32768. The output value in **D** is calculated as follows.

$$\text{Output value in D} = (-32768) + (-100) = 16\#8000 + 16\#FF9C = 16\#7F9C = 32668$$

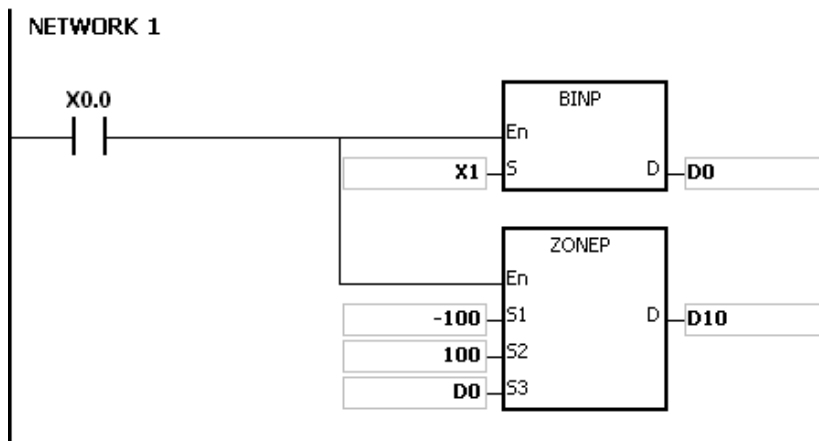
- If the instruction DZONE is executed, the negative deviation in **S₁**, the positive deviation in **S₂**, the input value in **S₃**, and the output value in **D** is within the range between -2147483648 and 2147483647. Suppose the negative deviation in (**S₁+1, S₁**) is -1000 and the input value in (**S₃+1, S₃**) is -2147483648. The output value in (**D+1, D**) is calculated as follows.

$$\text{Output value in (D+1, D)}$$

$$= -2147483648 + (-1000) = 16\#80000000 + 16\#FFFFFFC8 = 16\#7FFFFFFC8 = 2147482648$$

Example 1:

When X0.0 is ON, -100 or 100 is added to the binary-coded decimal value in X1, and the sum is stored in D10.

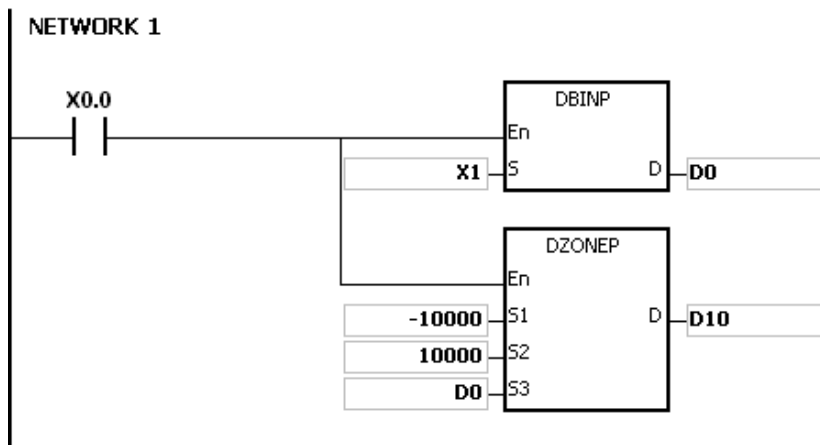


The execution results:

Negative deviation	Positive deviation	Input value in D0	Function	Output value in D10
-100	100	-10	$D0 < 0 \Rightarrow D10 = (-10) + (-100)$	-110
		0	$D0 = 0 \Rightarrow D10 = 0$	0
		50	$D0 > 0 \Rightarrow D10 = 50 + 100$	150

Example 2:

When X0.0 is ON, -10000 or 10000 is added to the binary-coded decimal value in (X2, X1), and the sum is stored in (D11, D10).



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

3.16 Structure Creation Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>FOR</u>	–	–	Start of the nested loop	3
FC	<u>NEXT</u>	–	–	End of the nested loop	1
FC	<u>BREAK</u>	–	✓	Terminating the FOR-NEXT loop	5

FB/FC	Instruction			Operand				Description					
FC		FOR		S				Start of the nested loop					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

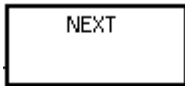
Graphic expression:



S : Number of times the loop is executed repeatedly

FB/FC	Instruction		Operand	Description
FC		NEXT	-	End of the nested loop
			Pulse instruction	16-bit instruction
			-	AH Motion CPU
				32-bit instruction
				-

Graphic expression:

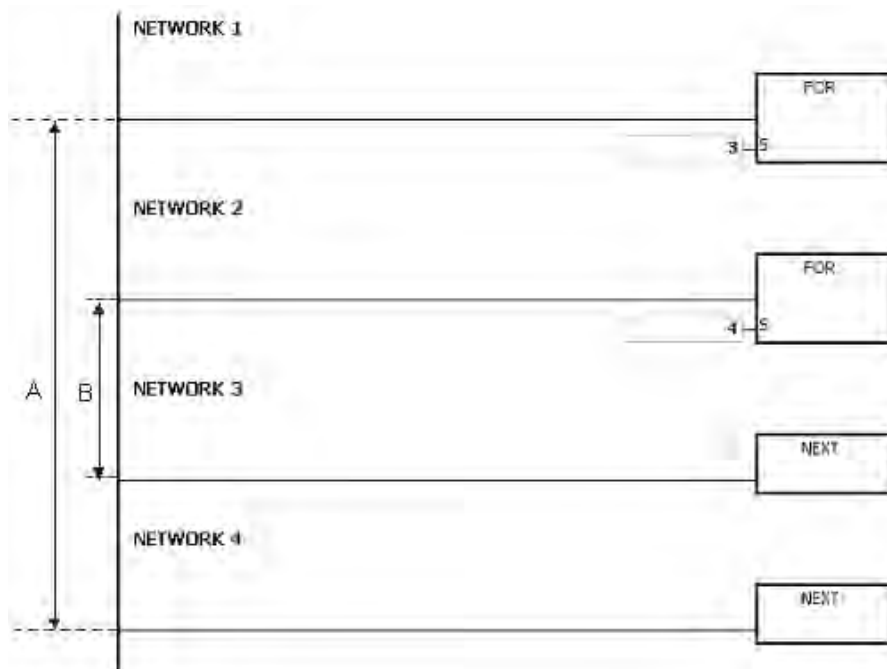


Explanation:

- The program between FOR and NEXT is executed N times. After the program between FOR and NEXT is executed N times, the program follows NEXT is executed. The instruction FOR specifies the number of times the program between FOR and NEXT is executed.
 - N should be within the range between 1 and 32,767. If N is less than 1, it is count as 1.
- If the program between FOR and NEXT is not executed, it can be skipped by the use of the instruction CJ.
- The following conditions result in errors.
 - The instruction NEXT is prior to the instruction FOR.
 - The instruction FOR exists, but the instruction NEXT does not exist.
 - The instruction NEXT follows the instruction FEND or END.
 - The number of times the instruction FOR is used is different from the number of times the instruction NEXT is used.
- FOR/NEXT supports the nested program structure. There are at most 32 levels of nested program structures. If the loop is executed many times, it takes more time for the program in the PLC to be scanned, and the watchdog timer error will occur. Users can use the instruction WDT to resolve the problem.

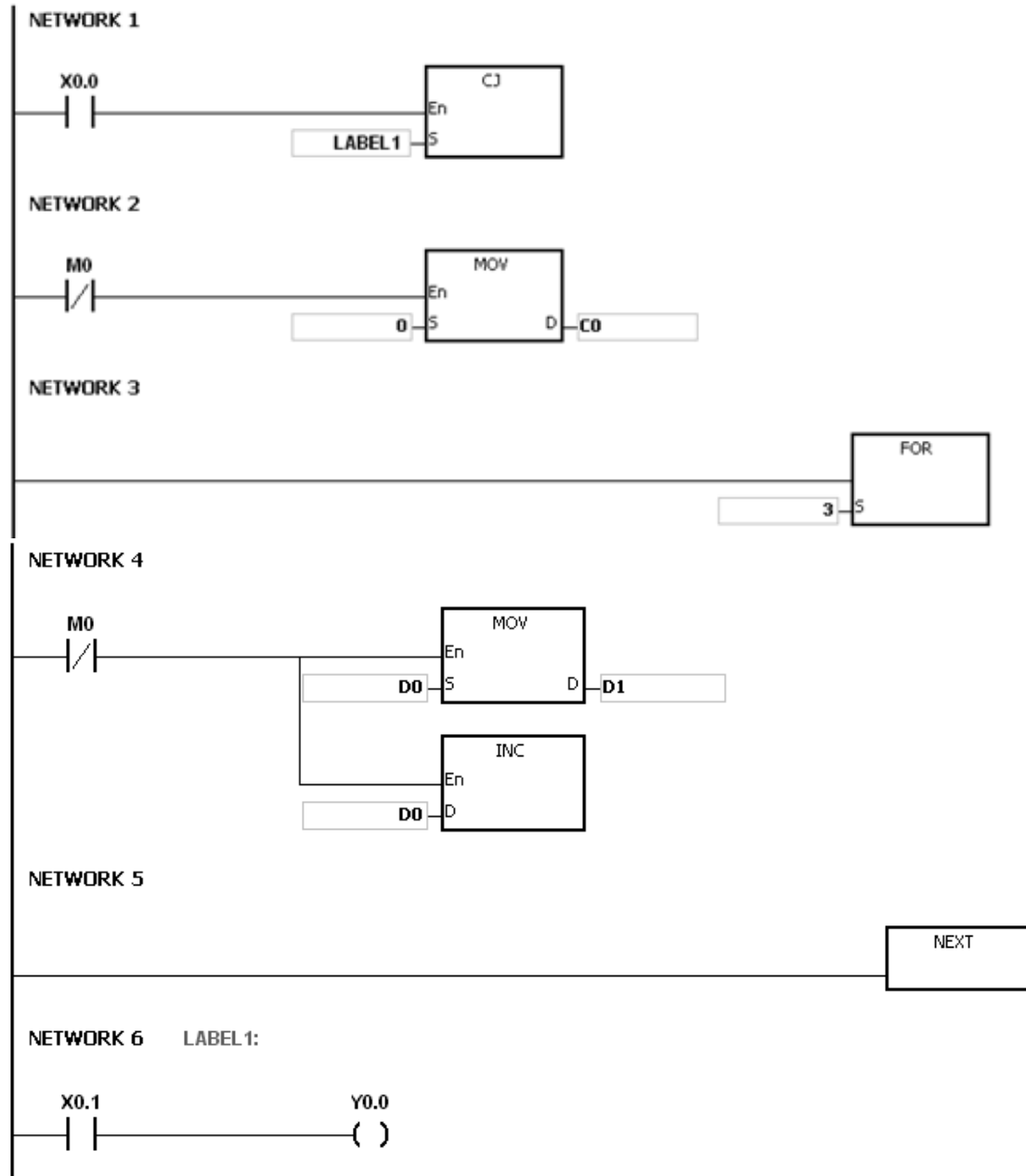
Example 1:

After program A is executed three times, the program follows the instruction NEXT is executed. Program B is executed four times every time program is executed. Therefore, program B is executed twelve times in total.



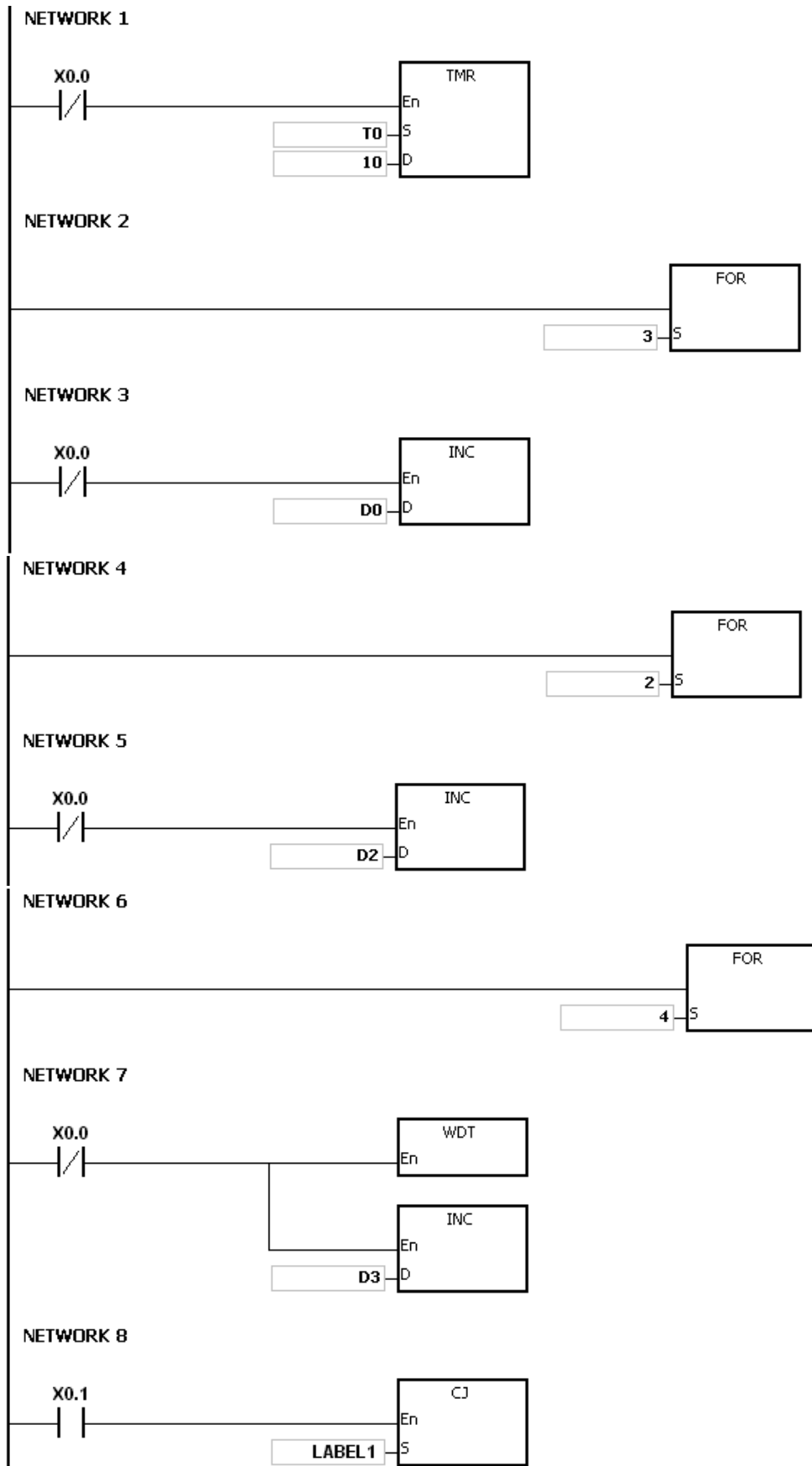
Example 2:

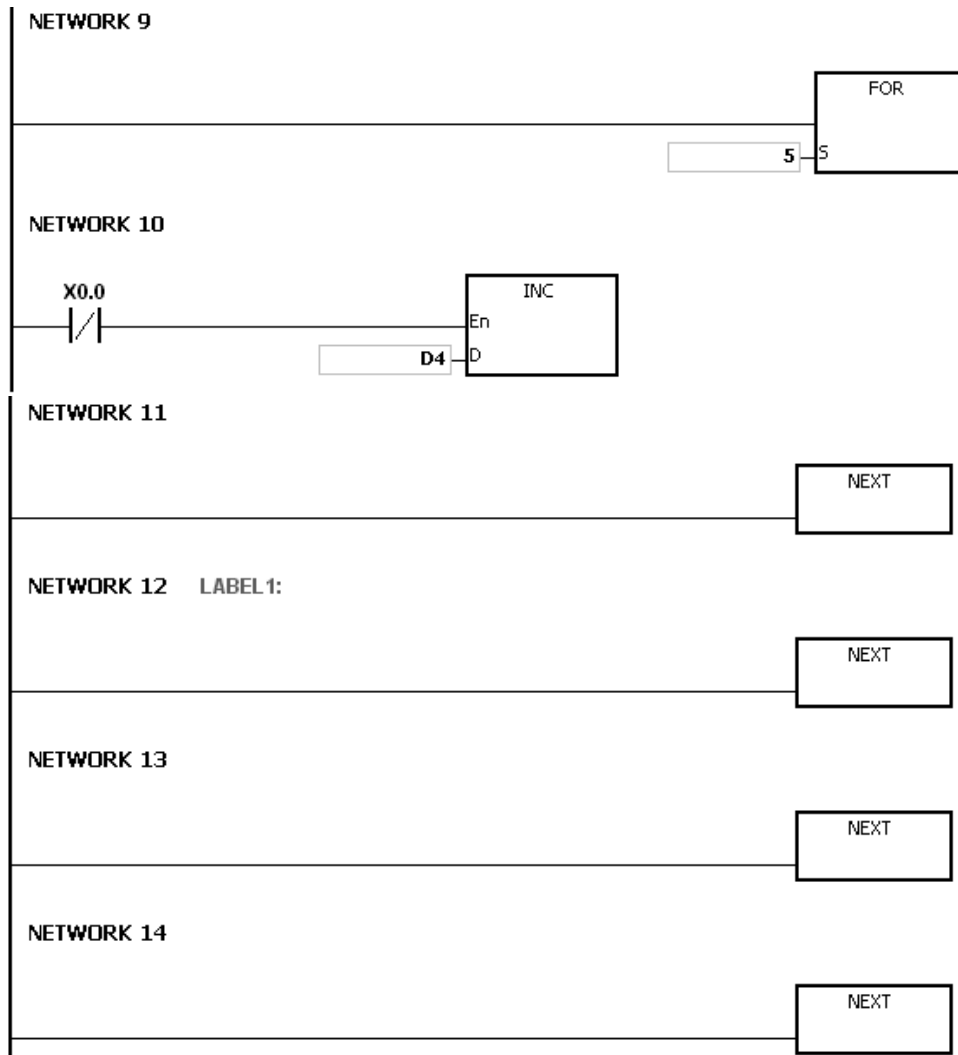
When X0.0 is OFF, the program between FOR and NEXT is executed. When X0.0 is ON, the instruction CJ is executed. The execution of the program jumps to LABEL 1; i.e. NETWORK 6, and NETWORK 4~NETWORK 5 are not executed.



Example 3:

If the program between FOR and NEXT is not executed, it can be skipped by the use of the instruction CJ. When X0.1 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~NETWORK 11 are not executed.





Additional remark:

Please refer to ISPSOft User Manual for more information related to the usage of the label.

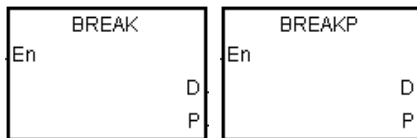
FB/FC	Instruction		Operand		Description
FC	BREAK	P	D, P		Terminating the FOR-NEXT loop

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	●	●			●	●		●	●		●	○	●				
P																	

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



D : Device in which the remaining number of times the loop can be executed is stored

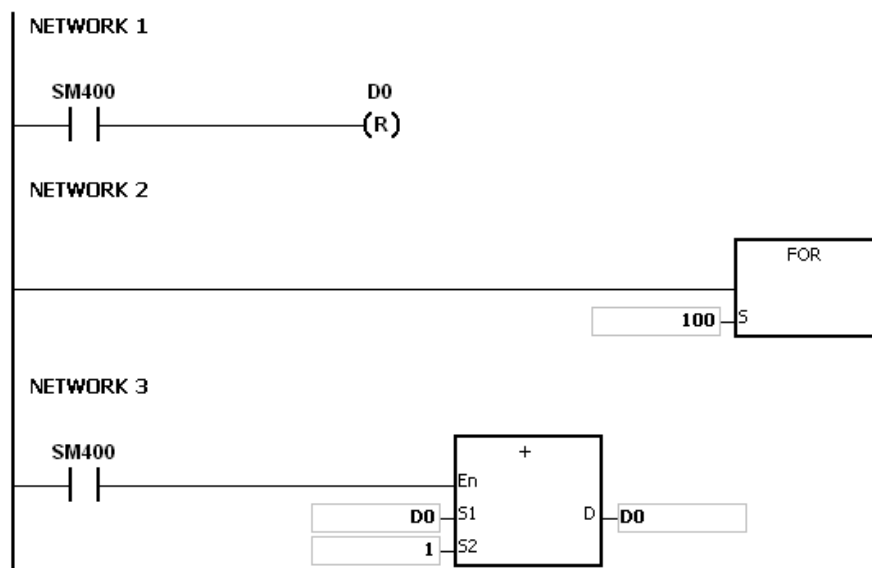
P : Pointer

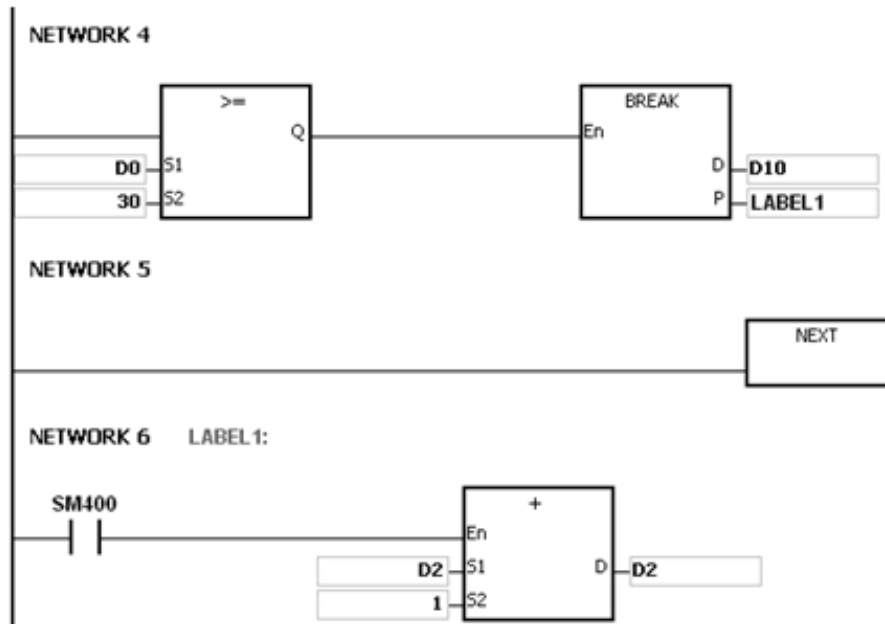
Explanation:

1. The instruction BREAK is used to terminate the FOR/NEXT loop. The remaining number of times the FOR/NEXT loop can be repeated is stored in **D**, and the execution of the program jumps to the part of program specified by the pointer
2. When the instruction BREAK 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.

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 remark:**

1. If the part of the program specified by the pointer in the instruction BREAK does not exist, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2004.
2. If the instruction BREAK is outside the FOR/NEXT loop, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2017.
3. Please refer to ISPSOft User Manual for more information related to the usage of the label.

3.17 Module Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>FROM</u>	<u>DFROM</u>	✓	Reading the data from the control register in the special module	13
FC	<u>TO</u>	<u>DTO</u>	✓	Writing the data into the control register in the special module	13

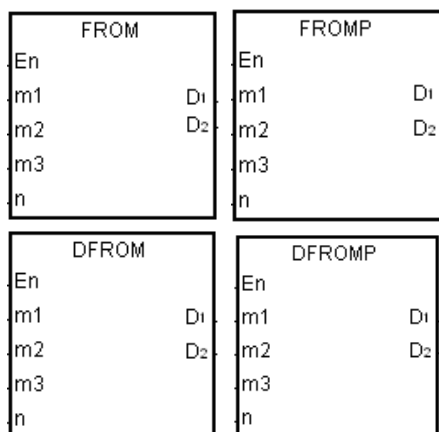
FB/FC	Instruction			Operand	Description
FC	D*	FROM	P	m ₁ , m ₂ , m ₃ , D ₁ , D ₂ , n	Reading the data from the control register in the special module

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m ₁ , m ₂ , m ₃		●	●*				●	●*						
D ₁ , D ₂		●	●*				●	●*						
n		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
m ₁ , m ₂ , m ₃	●	●			●	●	●	●	●				●	○	○		
D ₁ , D ₂	●	●			●	●	●	●	●				●				
n	●	●			●	●	●	●	●				●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:



- m₁ : Rack code
- m₂ : Slot code
- m₃ : Control register number
- D₁ : Device in which the data is stored
- D₂ : Device in which the error code is stored
- n : Data length

Explanation:

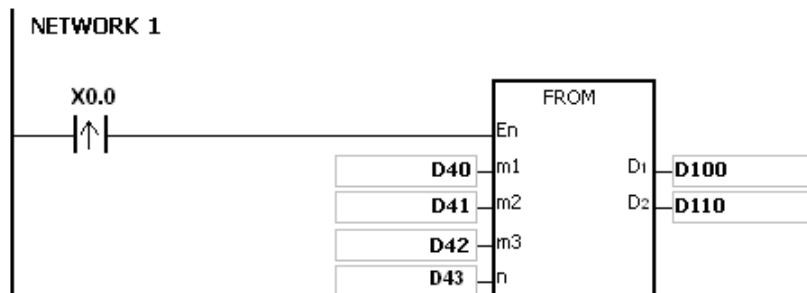
1. Users can use this instruction to read the data from the control register in the special module to the AH500 series PLC.
2. The operand m₁ should be within the range between 1 and 8. 1 represents a main rack, and 2~8 represent extension racks.
3. The operand m₂ should be within the range between 0 and 11. If the rack code is 1, the slot code should be within the range between 0 and 11. If the rack code is within the range between 2 and 8, the slot code should be within the range between 0 and 7.
4. The operand m₃ specifies the control register number.
5. When the instruction FROM is executed, D₂ is set to 0. When an error occurs, D₂ is not set to 0. Please refer to the additional remark below for more information about the error codes.
6. The operand n used in the 16-bit instruction should be within the range between 1 and 256, and the operand n used in the 32-bit instruction should be within the range between 1 and 128.
7. Only the 32-bit instructions can use the 32-bit counter.
8. Please refer to the regulation of the operands in the instruction TO for more information about the numbering

of the special modules.

- Special modules include analog I/O modules, network I/O modules, and position I/O modules.

Example:

Suppose the first special module at the right side of the CPU module is AH10SCM-A5. When X0.0 is switched from OFF to ON, the instruction FROM is executed. The mode of the data exchange through COM1 on AH10SCM-5A stored in CR#7 is read into D100. Owing to the fact that no error occurs, the code stored in D110 is 16#0000.



The use of the parameters:

- The module is placed on the main rack. Therefore, the rack code stored in D40 is 16#0001.
- The module is inserted in the first slot. Therefore, the slot code stored in D41 is 16#0000.
- The mode of the data exchange through COM1 is stored in CR#7. Therefore, the control register number stored in D42 is 16#0007.
- Owing to the fact that the mode of the data exchange through COM1 occupies one register, the value in D43 is 1.
- The data which is read from CR#7 is stored in D100.

Additional remark:

- If the values in m_1 and m_2 exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If $D_1 \sim D_{1+n-1}$ exceed the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- 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.
- Due to the fact that the use of the instruction FROM decreases the execution efficiency of the CPU module and that of the I/O module, users should use it less often.
- The descriptions of the error codes:

Error code	Description
16#2003	Please refer to point 1 and point 2 in the additional remark.
16#200B	Please refer to point 3 in the additional remark.
16#1400	An error occurs when the data is accessed through the auxiliary processor.
16#1401	An error occurs when the data in the I/O module is accessed.
16#1402	The arrangement of the I/O modules is not consistent with the module table.
16#1407	A communication error occurs when the data is accessed through the auxiliary processor.

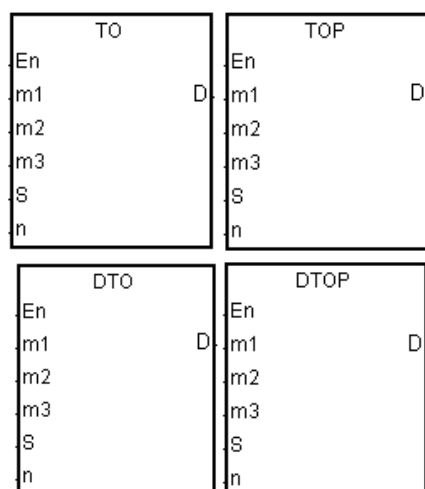
FB/FC	Instruction			Operand	Description
FC	D*	TO	P	m_1, m_2, m_3, S, D, n	Writing the data into the control register in the special module

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
m_1, m_2, m_3		●	●*				●	●*						
S		●	●*				●	●*						
D		●	●*				●	●*						
n		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
m_1, m_2, m_3	●	●			●	●	●	●	●				●	○	○		
S	●	●			●	●	●	●	●				●	○	○		
D	●	●			●	●	●	●	●				●				
n	●	●			●	●	●	●	●				●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

 m_1 : Rack code m_2 : Slot code m_3 : Control register number**S** : Data which is written into the control register**D** : Device in which the error code is stored**n** : Data length

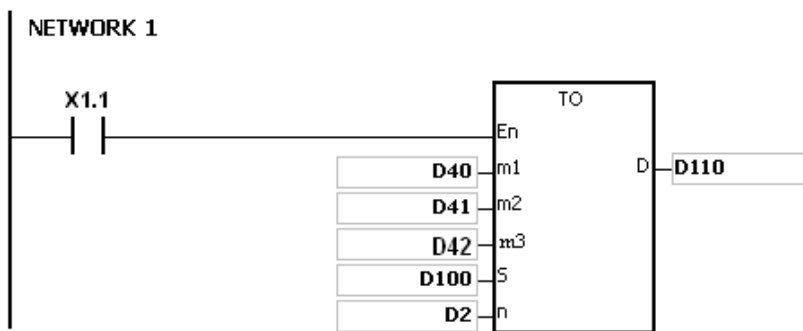
Explanation:

- Users can use this instruction to write the data in the AH500 series PLC into the control register in the special module.
- The operand m_1 should be within the range between 1 and 8. 1 represents a main rack, and 2~8 represent extension racks.
- The operand m_2 should be within the range between 0 and 11. If the rack code is 1, the slot code should be within the range between 0 and 11. If the rack code is within the range between 2 and 8, the slot code should be within the range between 0 and 7.
- The operand m_3 specifies the control register number.
- When the instruction TO is executed, **D** is set to 0. When an error occurs, **D** is not set to 0. Please refer to the additional remark below for more information about the error codes.

6. The operand **n** used in the 16-bit instruction should be within the range between 1 and 256, and the operand **n** used in the 32-bit instruction should be within the range between 1 and 128.
7. Only the 32-bit instructions can use the 32-bit counter.
8. Special modules include analog I/O modules, network I/O modules, and position I/O modules.
9. When **S** is a decimal value or a hexadecimal value, **n** decimal values or **n** hexadecimal values are transmitted to the I/O module. Suppose **S** is 16#0001 and **n** is 3. Three 16#0001s are transmitted to the I/O module.

Example:

Suppose the first special module at the right side of the CPU module is AH10SCM-A5. When X1.1 is switched from OFF to ON, the instruction TO is executed. The mode of the data exchange through COM1 on AH10SCM-5A stored in CR#7 changes from being disabled to being enabled. Owing to the fact that no error occurs, the code stored in D110 is 16#0000.

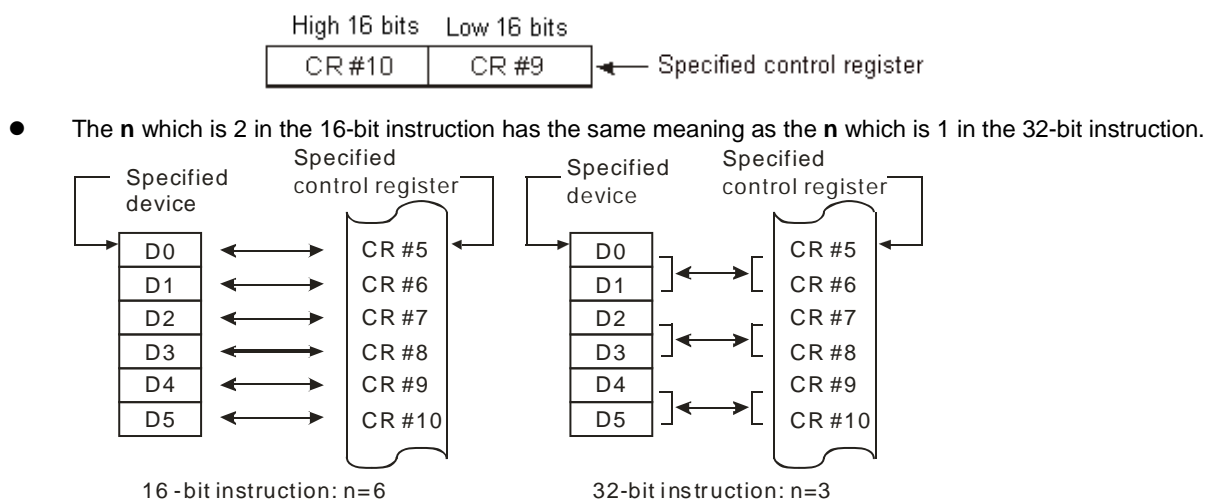


The use of the parameters:

- The module is placed on the main rack. Therefore, the rack code stored in D40 is 16#0001.
- The module is inserted in the first slot. Therefore, the slot code stored in D41 is 16#0000.
- The mode of the data exchange through COM1 is stored in CR#7. Therefore, the control register number stored in D42 is 16#0007.
- Because the mode of the data exchange through COM1 occupies one register, the value in D2 is 1.
- The data which is written into CR#7 is stored in D100. Therefore, the value in D100 is 16#0002.

The regulation of the operands in the instruction:

- The operand m1 specifies the rack code. It should be within the range between 1 and 8. 1 represents a main rack, and 2~8 represent extension racks.
- The operand m2 specifies the slot code. It should be within the range between 0 and 11. If the rack code is 1, the slot code should be within the range between 0 and 11. If the rack code is within the range between 2 and 8, the slot code should be within the range between 0 and 7.
- The operand m3 specifies the control register number. The 16-bit memories built in the special modules are called the control registers. The control register numbers are decimal numbers #0-#N, and the number of control registers varies with the module. The operating conditions of the special module and the setting values are stored in the control registers.
- At most 68 special modules can be placed on the rack, and they do not occupy inputs/outputs.
- If the instruction FROM/TO is used, one control register is taken as a unit for the reading/writing of the data. If the instruction DFROM/DTO is used, two control registers are taken as a unit for the reading/writing of the data.

**Additional remark:**

1. If the values in m_1 and m_2 exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If $S \sim S+n-1$ 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 use of the instruction TO decreases the execution efficiency of the CPU module and that of the I/O module, users should use it less often.
5. The descriptions of the error codes:

Error code	Description
16#2003	Please refer to point 1 and point 2 in the additional remark.
16#200B	Please refer to point 3 in the additional remark.
16#1400	An error occurs when the data is accessed through the auxiliary processor.
16#1401	An error occurs when the data in the I/O module is accessed.
16#1402	The arrangement of the I/O modules is not consistent with the module table.
16#1407	A communication error occurs when the data is accessed through the auxiliary processor.

3.18 Floating Point Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	–	<u>FSIN</u>	✓	Sine of the floating-point number	5-6
FC	–	<u>FCOS</u>	✓	Cosine of the floating-point number	5-6
FC	–	<u>FTAN</u>	✓	Tangent of the floating-point number	5-6
FC	–	<u>FASIN</u>	✓	Arcsine of the floating-point number	5-6
FC	–	<u>FACOS</u>	✓	Arccosine of the floating-point number	5-6
FC	–	<u>FATAN</u>	✓	Arctangent of the floating-point number	5-6
FC	–	<u>FSINH</u>	✓	Hyperbolic sine of the floating-point number	5-6
FC	–	<u>FCOSH</u>	✓	Hyperbolic cosine of the floating-point number	5-6
FC	–	<u>FTANH</u>	✓	Hyperbolic tangent of the floating-point number	5-6
FC	–	<u>FRAD</u>	✓	Converting the degree to the radian	5-6
FC	–	<u>FDEG</u>	✓	Converting the radian to the degree	5-6
FC	<u>SQR</u>	<u>DSQR</u>	✓	Square root of the binary number	5
FC	–	<u>FSQR</u>	✓	Square root of the floating-point number	5-6
FC	–	<u>FEXP</u>	✓	Exponent of the floating-point number	5-6
FC	–	<u>FLOG</u>	✓	Logarithm of the floating-point number	7-9
FC	–	<u>FLN</u>	✓	Natural logarithm of the binary floating-point number	5-6
FC	–	<u>FPOW</u>	✓	Power of the floating-point number	7-9
FC	<u>RAND</u>	–	✓	Random number	7
FC	<u>BSQR</u>	<u>DBSQR</u>	✓	Square root of the binary-coded decimal number	5
FC	<u>BSIN</u>	–	✓	Sine of the binary-coded decimal number	5
FC	<u>BCOS</u>	–	✓	Cosine of the binary-coded decimal number	5
FC	<u>BTAN</u>	–	✓	Tangent of the binary-coded decimal number	5
FC	<u>BASIN</u>	–	✓	Arcsine of the binary-coded decimal number	5
FC	<u>BACOS</u>	–	✓	Arccosine of the binary-coded decimal number	5
FC	<u>BATAN</u>	–	✓	Arctangent e of the binary-coded decimal number	5

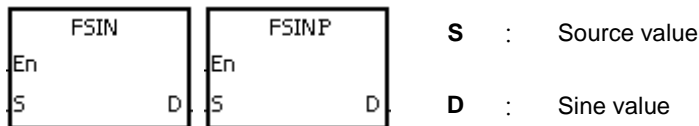
FB/FC	Instruction			Operand			Description		
FC	D*	FSIN	P	S, D			Sine of the floating-point number		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

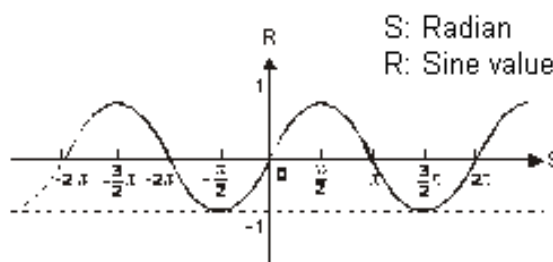
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



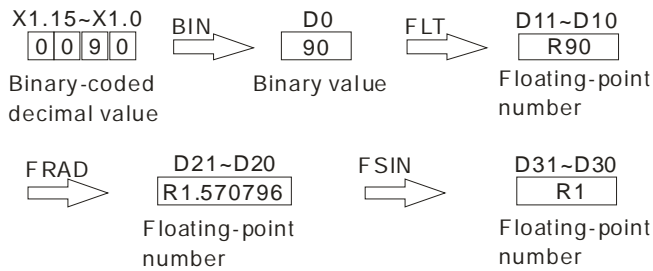
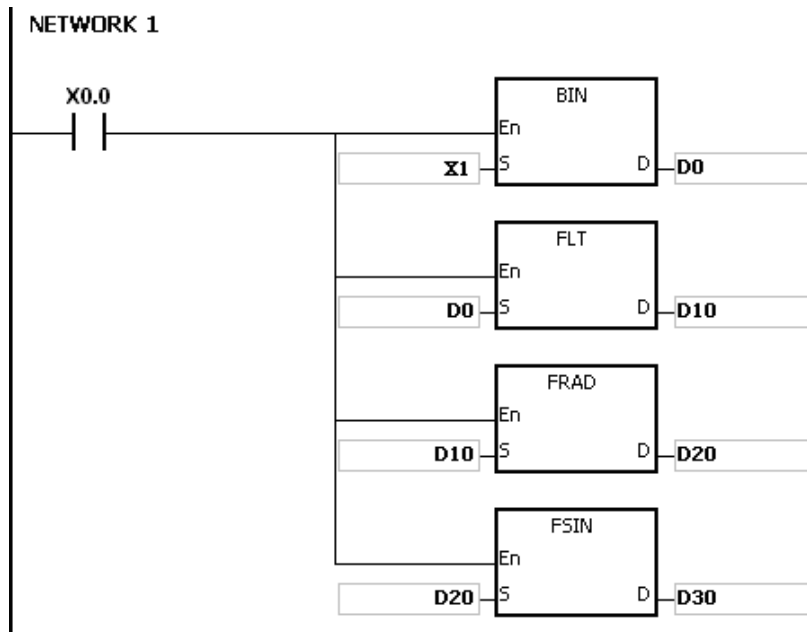
Explanation:

- Whether the source value specified by **S** is a radian or a degree depends on the state of SM695.
- If SM695 is OFF, the source value specified by **S** is a radian. $\text{Radian} = \text{Degree} \times \pi / 180$.
- If SM695 is ON, the source value specified by **S** is a degree. $\text{Degree} = \text{Radian} \times 180 / \pi$. ($0^\circ \leq \text{Degree} \leq 360^\circ$)
- If the conversion result is 0, SM600 is ON.
- The sine of the source value specified by **S** is stored in the register specified by **D**.
- The relation between radians and sine values are shown below.



Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The sine of the radian in (D21, D20) is stored in (D31, D30), and the sine value is the floating-point number.



Additional remark:

1. If the value in **S** exceeds the range of values which 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.
2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

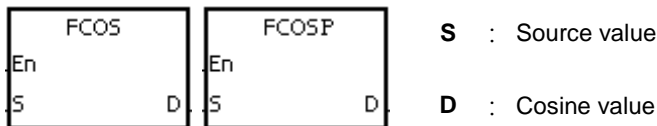
FB/FC	Instruction			Operand				Description				
FC	D*	FCOS	P	S, D				Cosine of the floating-point number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

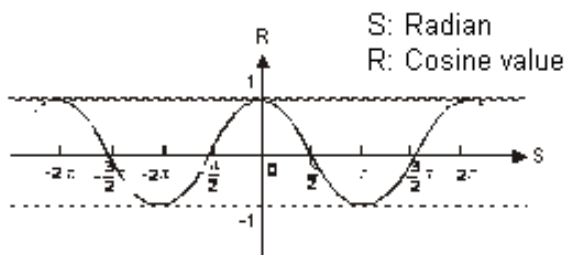
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



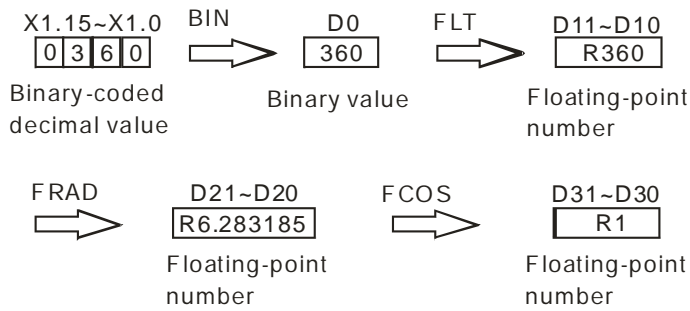
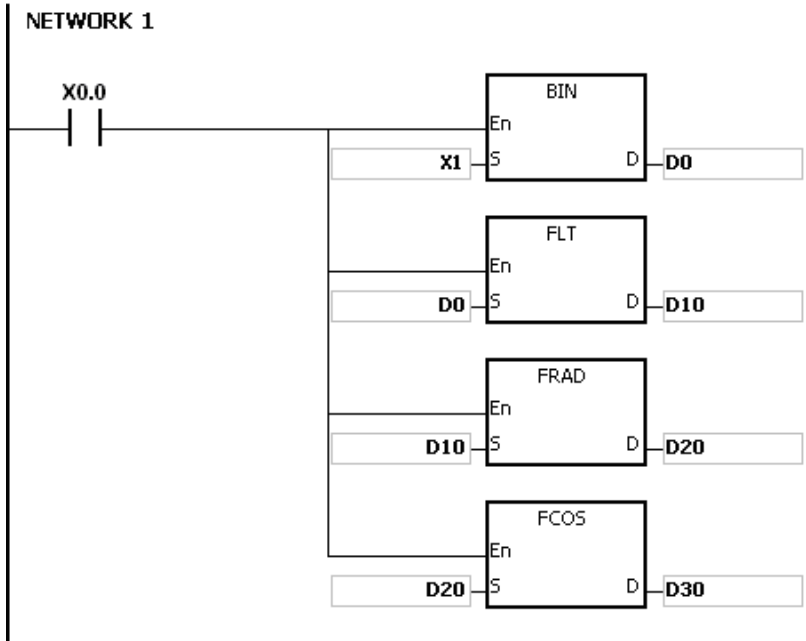
Explanation:

- Whether the source value specified by S is a radian or a degree depends on the state of SM695.
- If SM695 is OFF, the source value specified by S is a radian. $\text{Radian} = \text{Degree} \times \pi / 180$.
- If SM695 is ON, the source value specified by S is a degree.
 $\text{Degree} = \text{Radian} \times 180 / \pi$. ($0^\circ \leq \text{Degree} \leq 360^\circ$)
- If the conversion result is 0, SM600 is ON.
- The cosine of the source value specified by S is stored in the register specified by D.
- The relation between radians and cosine values are shown below.



Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The cosine of the radian in (D21, D20) is stored in (D31, D30), and the cosine value is the floating-point number.



Additional remark:

1. If the value in **S** exceeds the range of values which 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.
2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

FB/FC	Instruction			Operand						Description					
FC	D*	FTAN	P	S, D						Tangent of the floating-point number					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

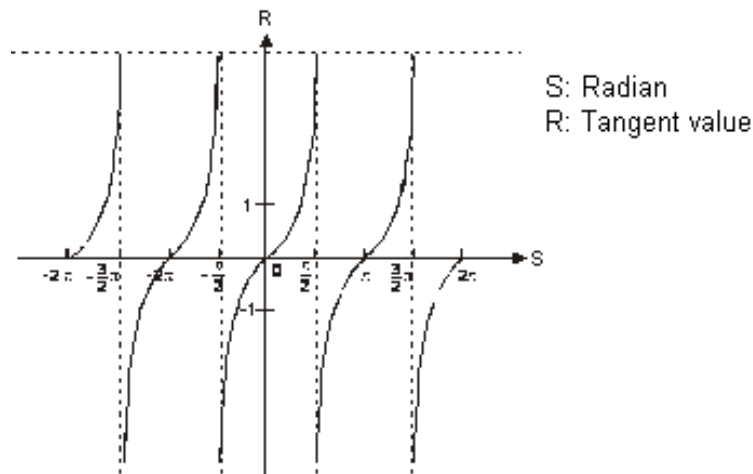
Graphic expression:



S : Source value
D : Tangent value

Explanation:

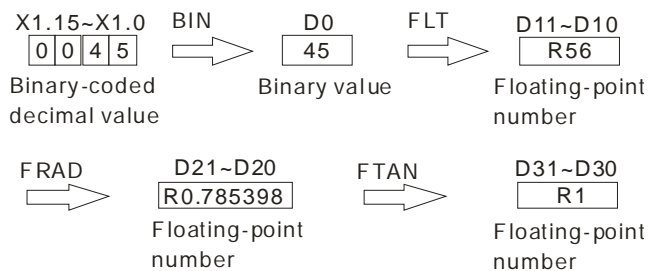
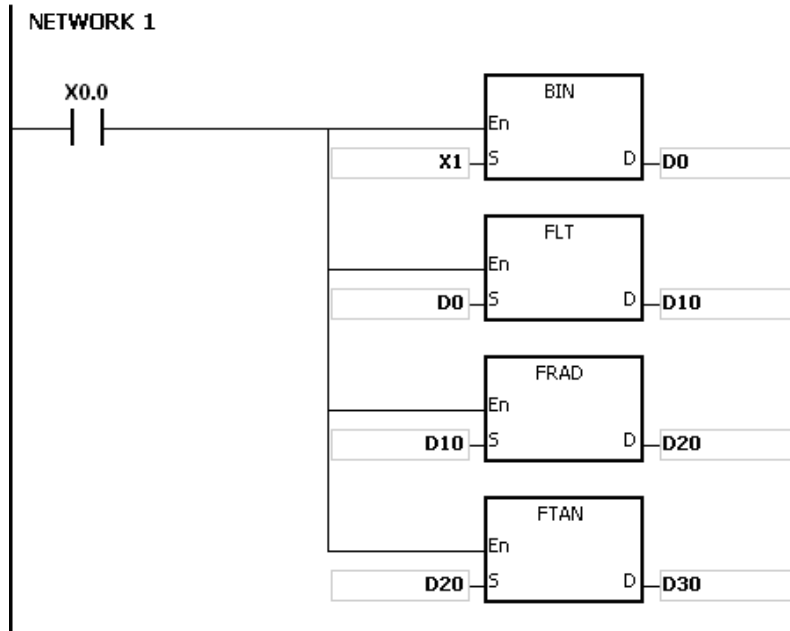
- Whether the source value specified by **S** is a radian or a degree depends on the state of SM695.
- If SM695 is OFF, the source value specified by **S** is a radian. $\text{Radian} = \text{Degree} \times \pi / 180$.
- If SM695 is ON, the source value specified by **S** is a degree.
 $\text{Degree} = \text{Radian} \times 180 / \pi$. ($0^\circ \leq \text{Degree} \leq 360^\circ$)
- If the conversion result is 0, SM600 is ON.
- The tangent of the source value specified by **S** is stored in the register specified by **D**.
- The relation between radians and tangent values are shown below.



Example:

When X0.0 is ON, the binary-coded decimal value in X1.15~X1.0 is converted into the binary value, and the

conversion result is stored in D0. The binary value in D0 is converted into the floating-point number, and the conversion result is stored in (D11, D10). The floating-point number in (D11, D10) is converted into the radian, and the conversion result is stored in (D21, D20). The tangent of the radian in (D21, D20) is stored in (D31, D30), and the tangent value is the floating-point number.



Additional remark:

1. If the value in **S** exceeds the range of values which 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.
2. If SM695 is ON, and the value in S is less than 0 or larger than 360, the instruction is not executed, SM0 is ON, and the error code is 16#2003.

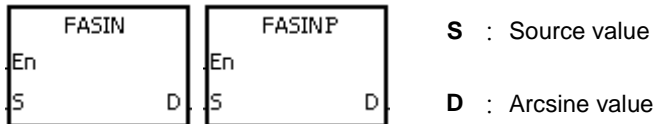
FB/FC	Instruction			Operand	Description
FC	D*	FASIN	P	S, D	Arcsine of the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

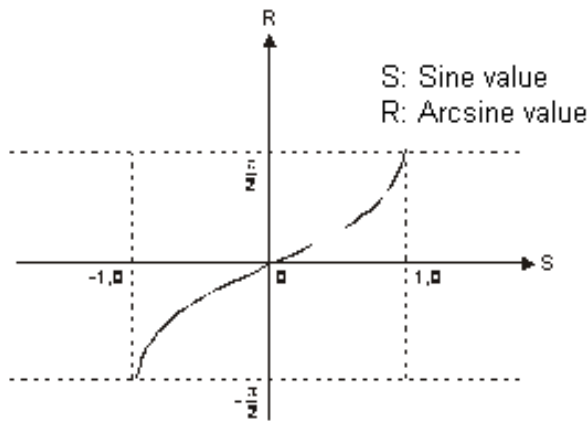
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

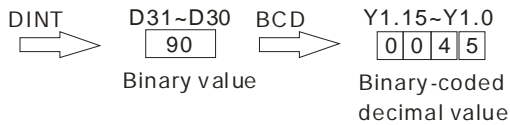
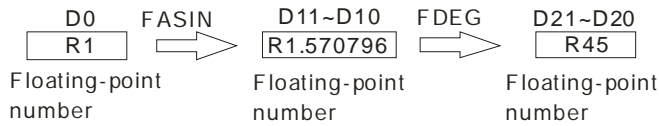
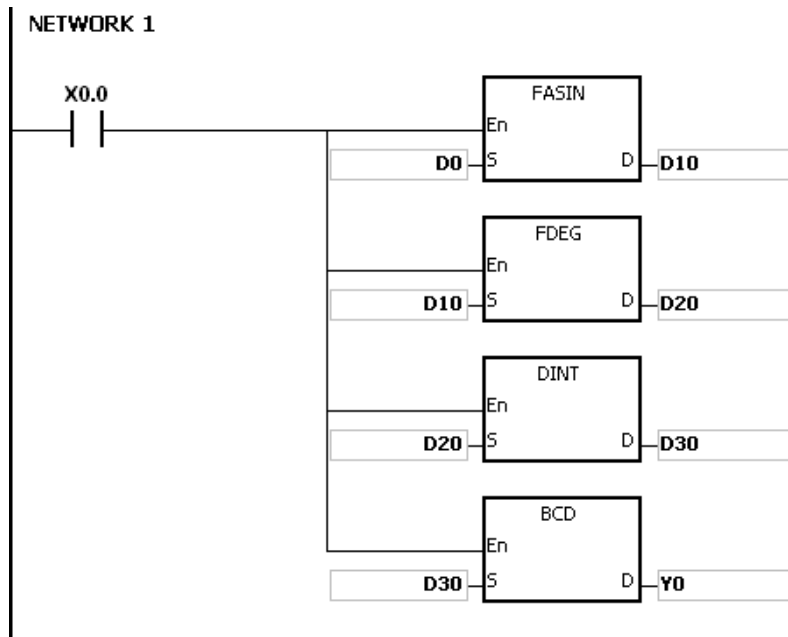
1. Arcsine value = \sin^{-1}
The relation between sine values and arcsine values are shown below.



2. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arcsine of the floating-point number in (D1, D0) is stored in (D11, D10). The arcsine value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in (D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.



Additional remark:

1. The floating-point number specified by the operand **S** should be within the range between -1.0 and $+1.0$. If the floating-point number is not within 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 which 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.

FB/FC	Instruction			Operand				Description				
FC	D*	FACOS	P	S, D				Arccosine of the floating-point number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

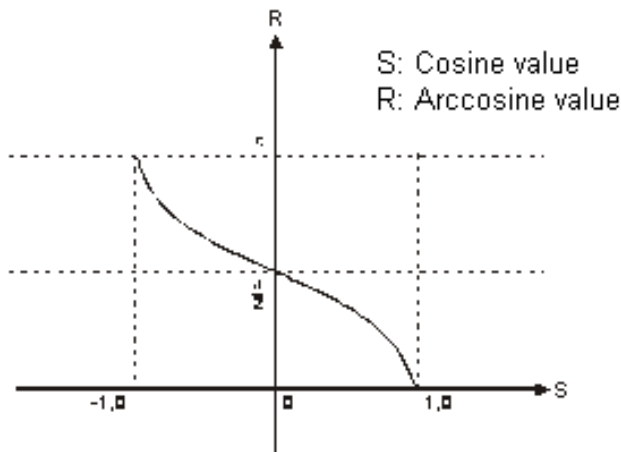


S : Source value
D : Arccosine value

Explanation:

1. Arccosine value = \cos^{-1}

The relation between cosine values and arccosine values are shown below.

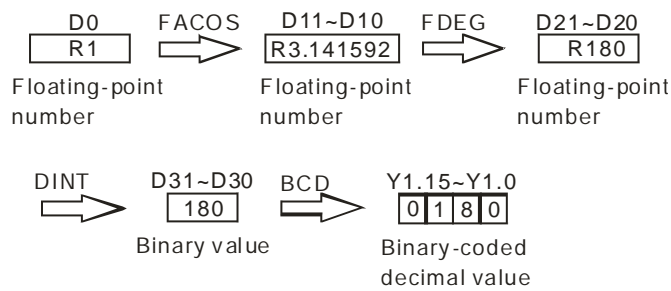
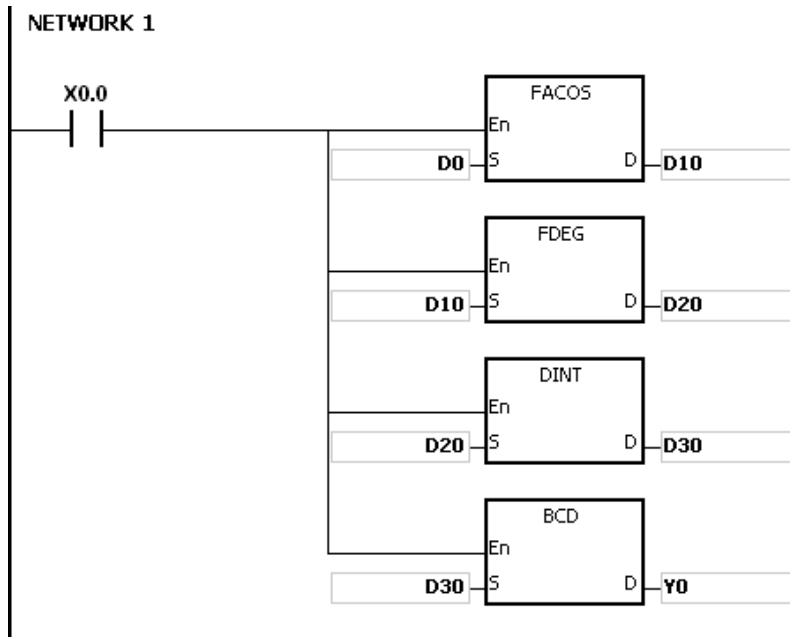


- If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.
- If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.
- If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arccosine of the floating-point number in (D1, D0) is stored in (D11, D10). The arccosine value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in

(D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.



Additional remark:

1. The floating-point number specified by the operand **S** should be within the range between -1.0 and $+1.0$. If the floating-point number is not within 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 which 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.

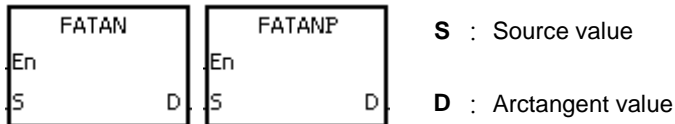
FB/FC	Instruction			Operand	Description
FC	D*	FATAN	P	S, D	Arctangent of the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

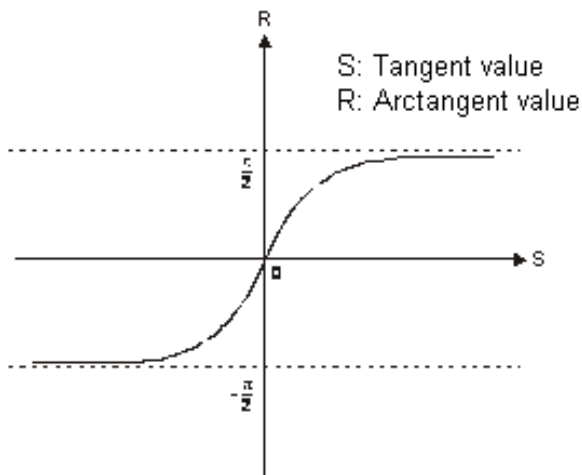
Graphic expression:



Explanation:

1. Arctangent value = \tan^{-1}

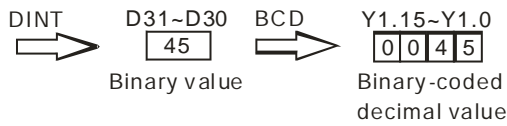
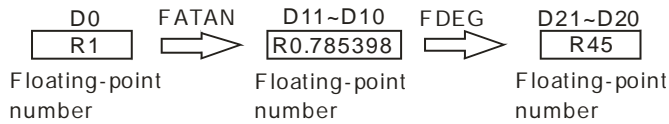
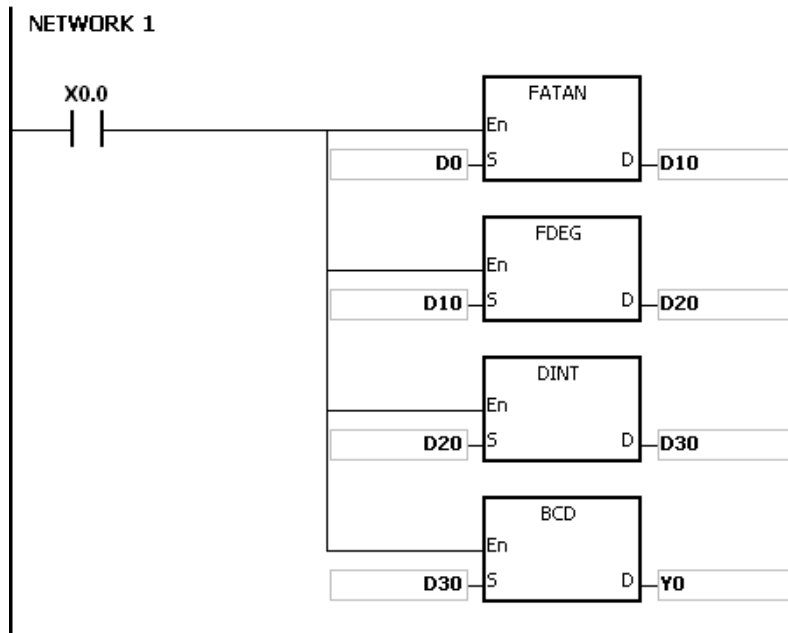
The relation between tangent values and arctangent values are shown below.



2. If the conversion result is 0, SM600 is ON.

Example:

When X0.0 is ON, the arctangent of the floating-point number in (D1, D0) is stored in (D11, D10). The arctangent value in (D11, D10) is converted into the degree, and the conversion result is stored in (D21, D20). The degree in (D21, D20) is converted into the integer, and the conversion result is stored in (D31, D30). The integer in (D31, D30) is converted into the binary-coded decimal value, and the conversion result is stored in Y0.15~Y0.0.



Additional remark:

If the value in **S** exceeds the range of values which 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.

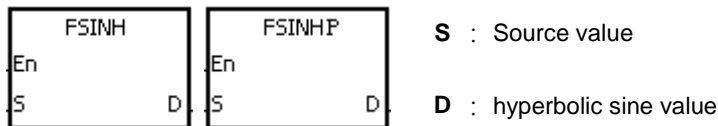
FB/FC	Instruction			Operand				Description				
FC	D*	FSINH	P	S, D				Hyperbolic sine of the floating-point number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

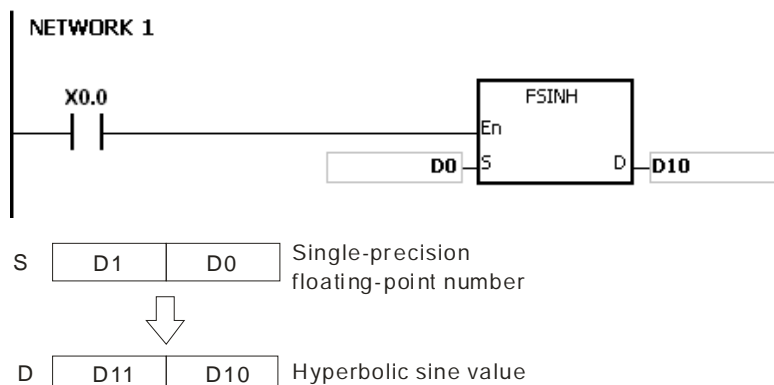


Explanation:

- Hyperbolic sine value = $(e^s - e^{-s})/2$.
- If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F800000, and SM602 is ON.
- If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#7F800000, and SM601 is ON.
- If the conversion result is 0, SM600 is ON.

Example:

- When X0.0 is ON, the hyperbolic sine of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic sine value in (D11, D10) is the floating-point number.



- If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, SM602 is ON.
- If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Additional result:

If the value in **S** exceeds the range of values which 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.

FB/FC	Instruction			Operand				Description				
FC	D*	FCOSH	P	S, D				Hyperbolic cosine of the floating-point number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

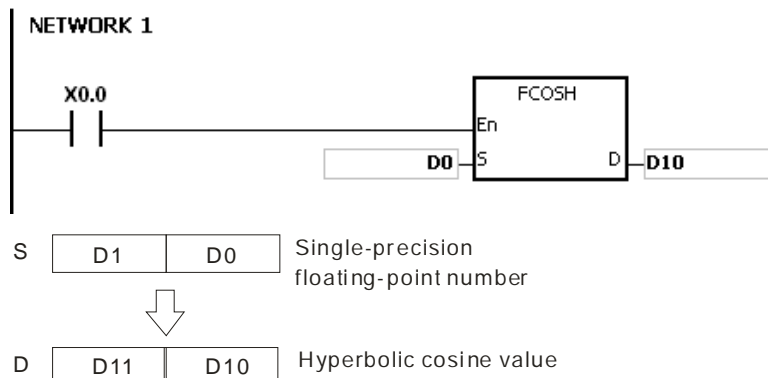


Explanation:

1. Hyperbolic cosine value=(e^s+e^{-s})/2.
2. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, 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 which can be represented by the minimum floating-point number, the value in D is 16#FF800000, and SM601 is ON.
4. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the hyperbolic cosine of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic cosine value in (D11, D10) is the floating-point number.



2. If the absolute value of the conversion result is larger than the value which 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 which can be represented by the minimum floating-point number, SM601 is ON.

4. If the conversion result is 0, SM600 is ON.

Additional remark:

If the value in **S** exceeds the range of values which 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.

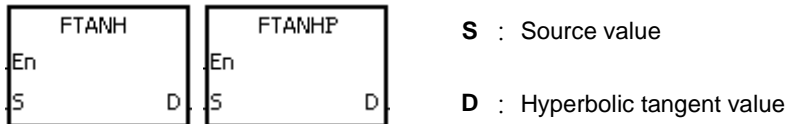
FB/FC	Instruction			Operand	Description
FC	D*	FTANH	P	S, D	Hyperbolic tangent of the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

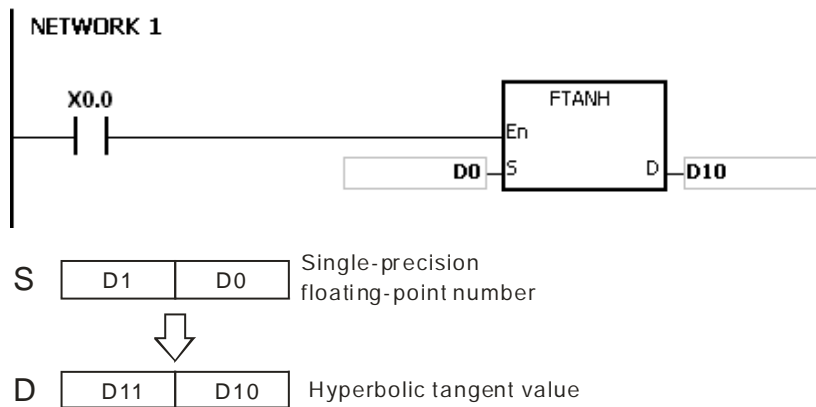


Explanation:

- Hyperbolic tangent value = $(e^s - e^{-s}) / (e^s + e^{-s})$.
- If the conversion result is 0, SM600 is ON.

Example:

- When X0.0 is ON, the hyperbolic tangent of the floating-point number in (D1, D0) is stored in (D11, D10). The hyperbolic tangent value in (D11, D10) is the floating-point number.



- If the conversion result is 0, SM600 is ON.

Additional remark:

If the value in **S** exceeds the range of values which 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.

FB/FC	Instruction			Operand				Description						
FC		FRAD	P	S, D				Converting the degree to the radian						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



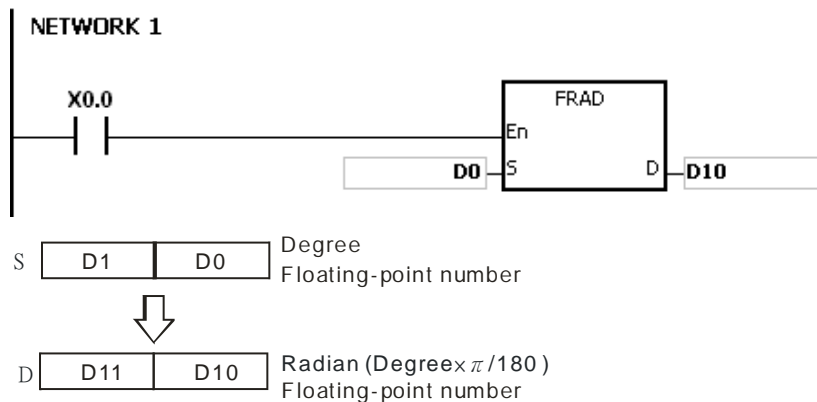
S : Data source (degree)
D : Conversion result (radian)

Explanation:

1. The equation below is used to convert degrees into radians.
2. $\text{Radian} = \text{Degree} \times (\pi/180)$.
3. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the degree in (D1, D0) is converted into the radian, and the conversion result is stored in (D11, D10). The radian in (D11, D10) is the floating-point number.



Additional remark:

If the value in **S** exceeds the range of values which 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.

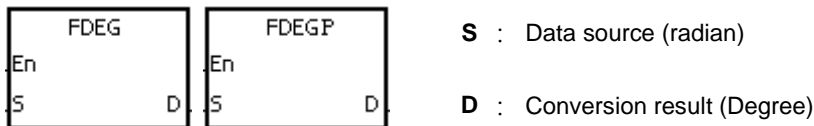
FB/FC	Instruction		Operand		Description
FC	FDEG	P	S, D		Converting the radian to the degree

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

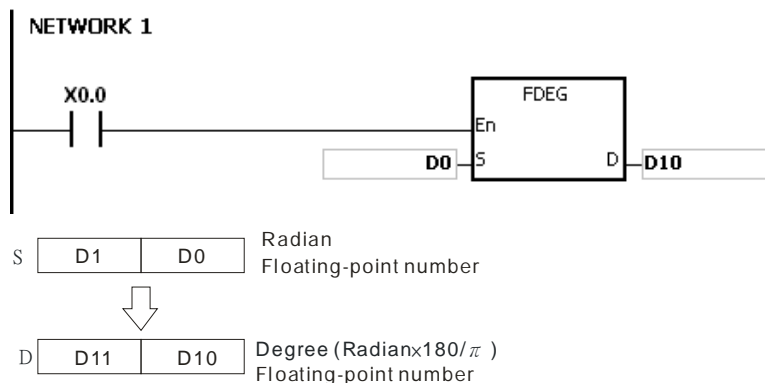


Explanation:

1. The equation below is used to convert radians into degrees.
2. Degree = Radian $\times(180/\pi)$.
3. If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in D is 16#7F7F F FFF.
4. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in D is 16#FF7 F F FFF.
5. If the conversion result is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the radian in (D1, D0) is converted into the degree, and the conversion result is stored in (D11, D10). The degree in (D11, D10) is the floating-point number.



Additional remark:

If the value in **S** exceeds the range of values which 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.

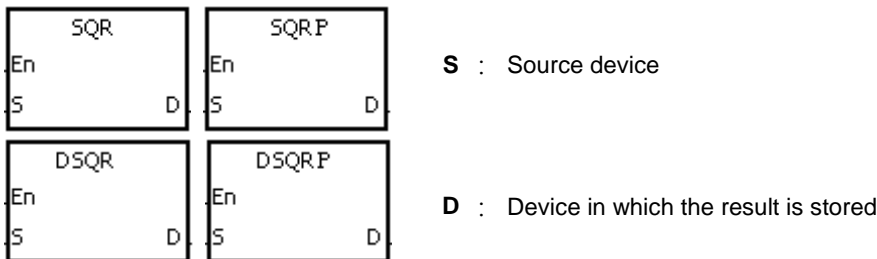
FB/FC	Instruction			Operand	Description
FC	D*	SQR	P	S, D	Square root of the binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●		●	○	○		
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3 Graphic expression:

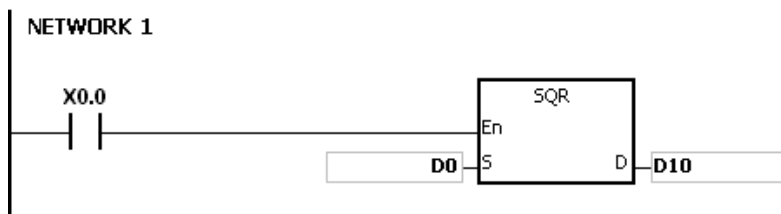


Explanation:

1. The square root of the value in the device specified by S is calculated, and the result is stored in the device specified by D.
2. The operation result stored in D is an integer. If the 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.

Example:

When X0.0 is ON, the square root of the value in D0 is calculated, and the result is stored in D10.



Additional remark:

The value in the device specified by S only can be a positive value. If the value in the device specified by S is a negative value, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

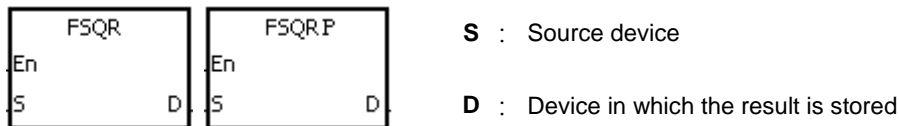
FB/FC	Instruction			Operand	Description
FC	D*	FSQR	P	S, D	Square root of the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

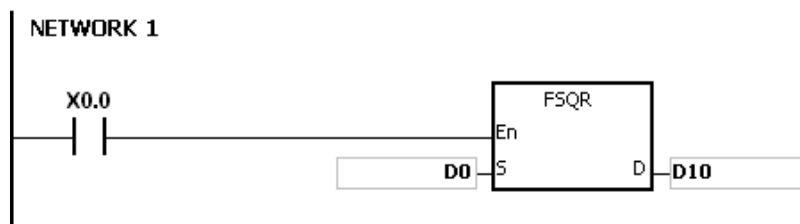


Explanation:

1. The square root of the floating-point number in the register specified by **S** is calculated, and the result is stored in the register specified by **D**.
2. If the operation result stored in **D** is 0, SM600 is ON.

Example 1:

When X0.0 is ON, the square root of the floating-point number in (D1, D0) is calculated, and the result is stored in (D11, D10).



Additional remark:

The value in the device specified by **S** only can be a positive value. If the value in the device specified by **S** is a negative value, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

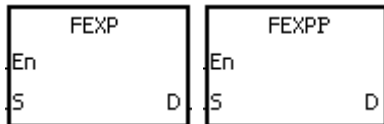
FB/FC	Instruction			Operand	Description
FC	D*	FEXP	P	S, D	An exponent of the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S : Source device

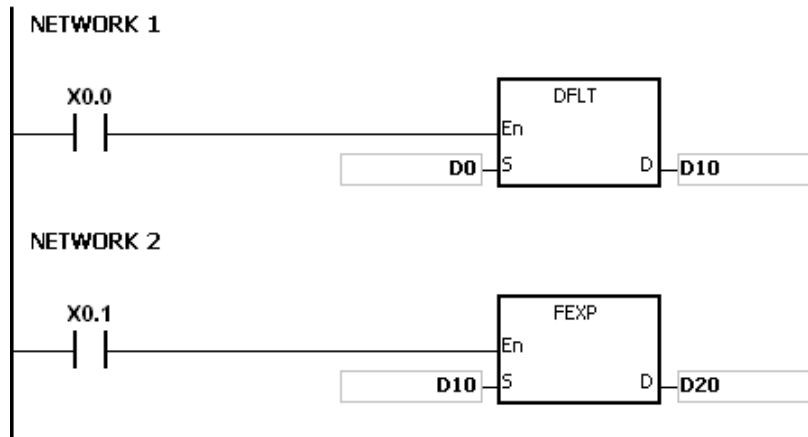
D : Device in which the operation result is stored

Explanation:

- Exponentiation involves two numbers, the base e which represents 2.71828, and the exponent in the device specified by **S**.
- $EXP[D+1, D]=[S+1, S]$.
- The number in the device specified by **S** can be a positive number or a negative number. The register specified by **D** should be a 32-bit register, and the number in the device specified by **S** should be a floating-point number.
- The value in the register specified by **D** is e^S . (e is 2.71828, and **S** represents the source data.)
- If the absolute value of the conversion result is larger than the value which can be represented by the maximum floating-point number, the value in the register specified by **D** is 16#7F800000, and SM602 is ON.
- If the operation result stored in **D** is 0, SM600 is ON.

Example:

- When X0.0 is ON, the value in (D1, D0) is converted into the floating-point number, and the conversion result is stored in (D11, D10).
- When X0.1 is ON, the exponentiation with the value in (D11, D10) as the exponent is performed. The result is a floating-point number, and is stored in (D21, D20).



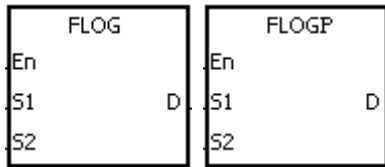
FB/FC	Instruction			Operand	Description
FC	D*	FLOG	P	S ₁ , S ₂ , D	Logarithm of the floating-point number

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂										●	●*			
D			●	●*				●	●*					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁ , S ₂	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : Device in which the base is stored

S₂ : Source device

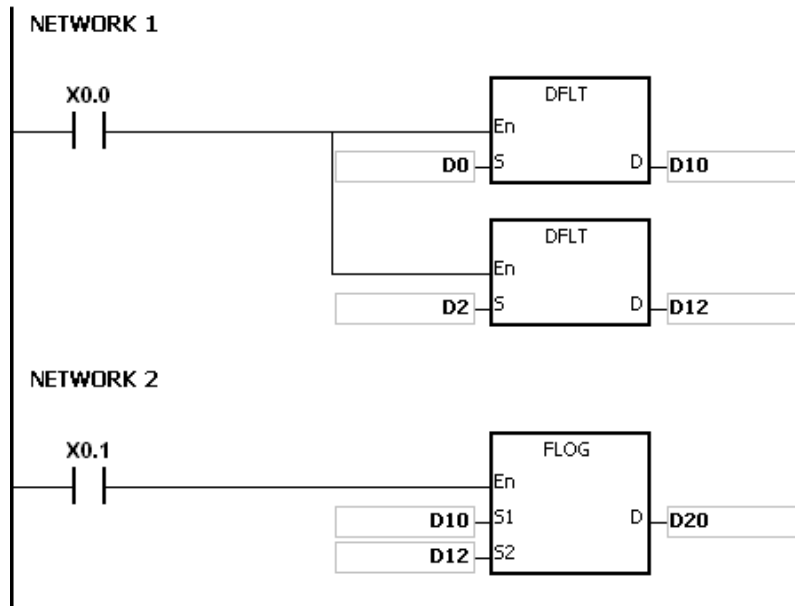
D : Device in which the operation result is stored

Explanation:

- The logarithm of the value in S₂ with respect to the value in S₁ is calculated, and the operation result is stored in D.
- The values in S₁ and S₂ only can be positive values. The register specified by D should be a 32-bit register, and the values in S₁ and S₂ should be floating-point numbers.
- $S_1^D = S_2 \rightarrow D = \text{Log}_{S_1} S_2$.
- Example: Suppose the values in S₁ and S₂ are 5 and 125 respectively. Find $\log_5 125$.
- $S_1^D = S_2 \rightarrow 5^D = 125 \rightarrow D = \log_5 125 = 3$.
- If the operation result stored in D is 0, SM600 is ON.

Example:

- When X0.0 is ON, the values in (D1, D0) and (D3, D2) are converted into the floating-point numbers, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.
- When X0.1 is ON, the logarithm of the floating-point number in (D13, D12) with respect to the floating-point number in (D11, D10) is calculated, and the operation result is stored in (D21, D20).

**Additional remark:**

If the value in **S₁** is less than or equal to 1, or if the value in **S₂** is less or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction			Operand	Description
FC	D*	FLN	P	S, D	Natural logarithm of the binary floating-point number

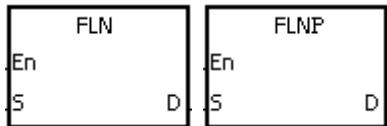
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S										●	●*			
D			●	●*				●	●*					

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●						○
D	●	●			●	●	●	●	●		●						

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

3

Graphic expression:



S : Source device

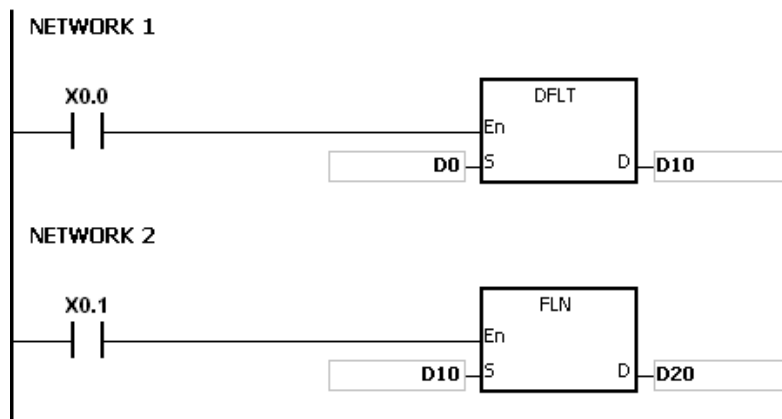
D : Device in which the operation result is stored

Explanation:

1. The natural logarithm of the operand **S** is calculated.
2. $LN[S+1, S]=[S+1, D]$
3. The value in **S** only can be a positive value. The register specified by **D** should be a 32-bit register, and the value in **S** should be a floating-point number.
4. $e^D=S \rightarrow$ The value in **D**=ln**S**. (**S** represents the source data.)
5. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the value in (D1, D0) is converted into the floating-point number, and the conversion result is stored in (D11, D10).
2. When X0.1 is ON, the natural logarithm of the floating-point number in (D11, D10) is calculated, and the operation result is stored in (D21, D20).



Additional remark:

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.

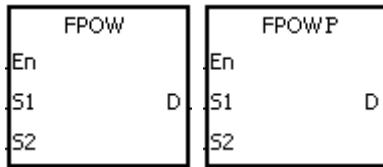
FB/FC	Instruction			Operand	Description
FC	FPOW	P		S₁, S₂, D	A power of the floating-point number

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂										●				
D			●					●						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁, S₂	●	●			●	●	●	●	●		●		●				○
D	●	●			●	●	●	●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



- S₁** : Device in which the base is stored
- S₂** : Device in which the power is stored
- D** : Device in which the operation result is stored

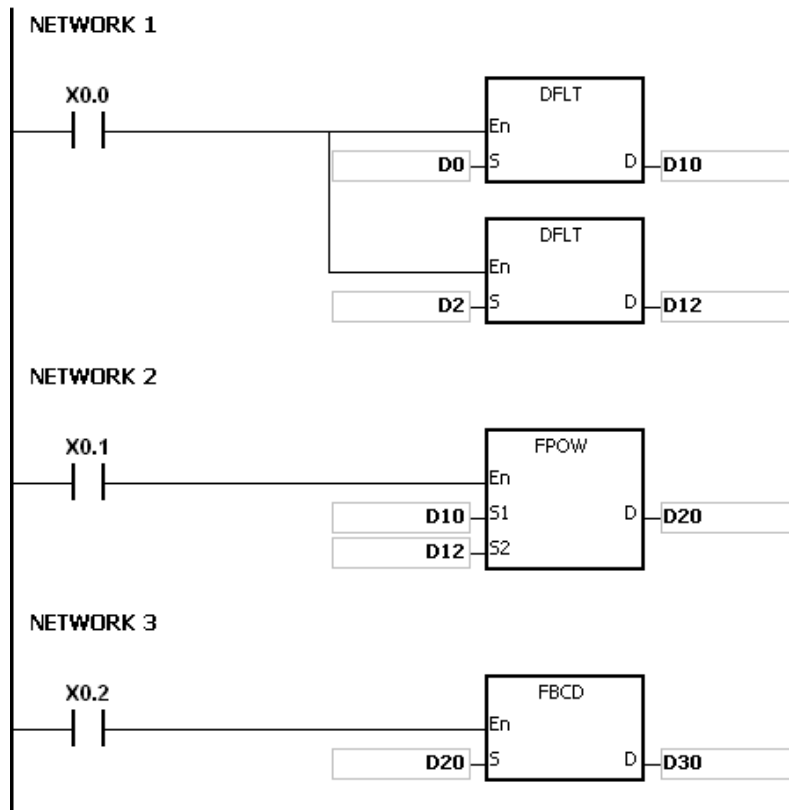
Explanation:

1. The single-precision floating-point number in **S₁** is raised to the power of the value in **S₂**, and the operation result is stored in **D**.
2. $D = \text{POW}[S_1 + 1, S_1]^{[S_2 + 1, S_2]}$
3. The value in **S₁** only can be a positive value, whereas the value in **S₂** can be a positive value or a negative value. The register specified by **D** should be a 32-bit register, and the values in **S₁** and **S₂** should be floating-point numbers.
4. $S_1^{S_2} = D$
5. Suppose the values in **S₁** and **S₂** are 5 and 3 respectively. $D = 5^3 = 125$.
6. If the absolute value of the operation result is large than the value which can be represented by the maximum floating-point number, SM602 is ON.
7. If the absolute value of the operation result is less than the value which can be represented by the minimum floating-point number, SM601 is ON.
8. If the absolute value of the conversion result is large than the value which can be represented by the maximum floating-point number, the value in **D** is 16#7F7FFFFFFF.
9. If the absolute value of the conversion result is less than the value which can be represented by the minimum floating-point number, the value in **D** is 16#FF7FFFFFFF.
10. If the operation result stored in **D** is 0, SM600 is ON.

Example:

1. When X0.0 is ON, the values in (D1, D0) and (D3, D2) are converted into the floating-point numbers, and the conversion results are stored in (D11, D10) and (D13, D12) respectively.

2. When X0.1 is ON, the floating-point number in (D11, D10) is raised to the power of the floating-point number in (D13, D12), and the operation result is stored in (D21, D20).
3. When X0.2 is ON, the binary floating-point number in (D21, D20) is converted into the binary-coded decimal floating-point number, and the conversion result is stored in (D31, D30).

**Additional remark:**

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.

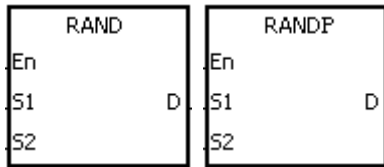
FB/FC	Instruction		Operand	Description
FC	RAND	P	S₁, S₂, D	Random number

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂	●	●			●	●		●			●	○	●	○	○		
D	●	●			●	●		●			●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



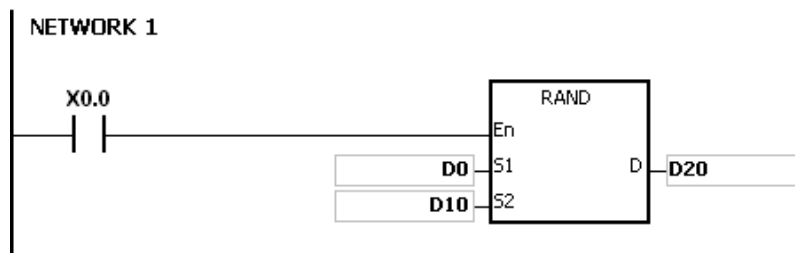
- S₁** : Minimum value
- S₂** : Maximum value
- D** : Device in which the result is stored

Explanation:

- The instruction is used to generate the random number within the range between the minimum value in **S₁** and the maximum value in **S₂**, and the result is stored in **D**.
- If the value in **S₁** is larger than the value in **S₂**, the values in **S₁** and **S₂** are taken as the maximum value and the minimum value respectively when the instruction is executed.

Example:

- When X0.0 is ON, the random number within the range between the minimum value in D0 and the maximum value in D10 is generated, and the result is stored in D20.



Additional remark:

The values in **S₁** and **S₂** should be within the range between 0 and 0~32767. If the value in **S₁** or **S₂** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

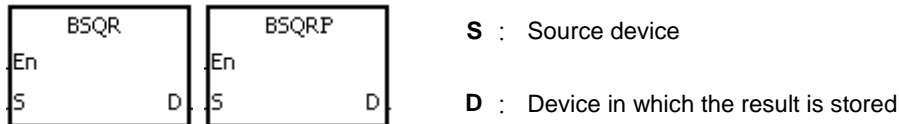
FB/FC	Instruction			Operand	Description
FC	D*	BSQR	P	S, D	Square root of the binary-coded decimal number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D	●	●			●	●	●	●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

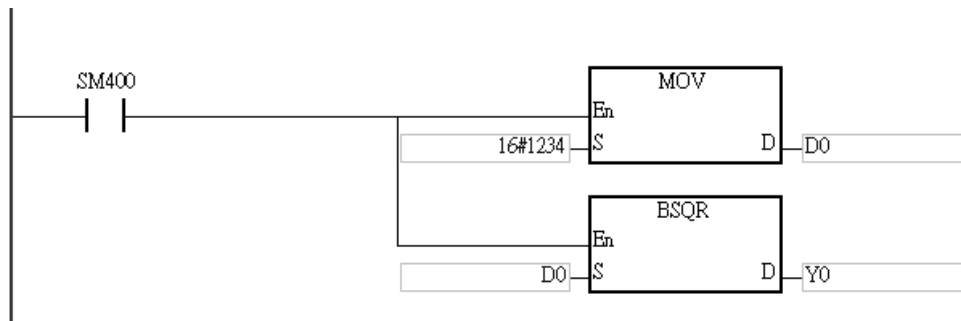


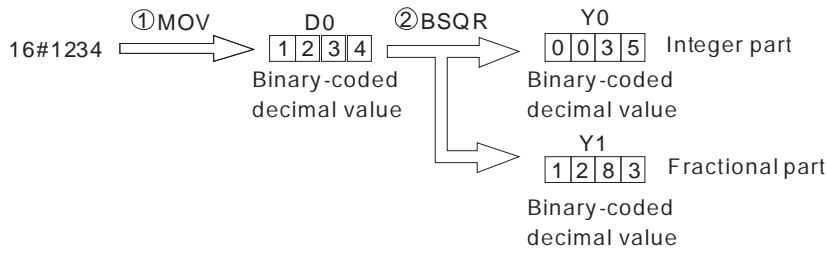
Explanation:

1. After the square root of the value in the device specified by **S** is calculated, the integer part is stored in the device specified by **D**, and the fractional part is stored in the device specified by **D+1**.
2. The 16-bit value in **S** should be within the range between 0 and 9,999, and the 32-bit value in **S** should be within the range between 0 and 99,999,999.
3. If the instruction BSQR is used, the square root is rounded down to the fourth decimal place.
4. If the instruction DBSQR is used, the square root is rounded down to the eighth decimal place.
5. If the operation result stored in **D** is 0, SM600 is ON.

Example 1:

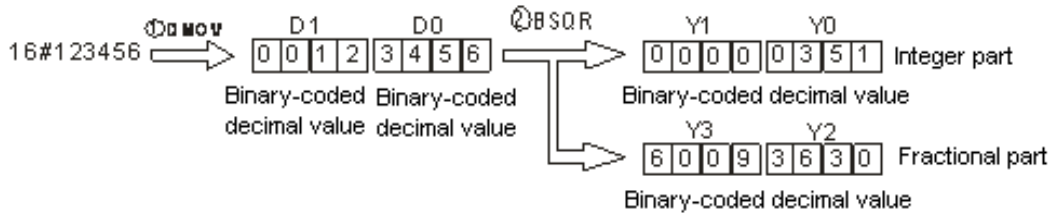
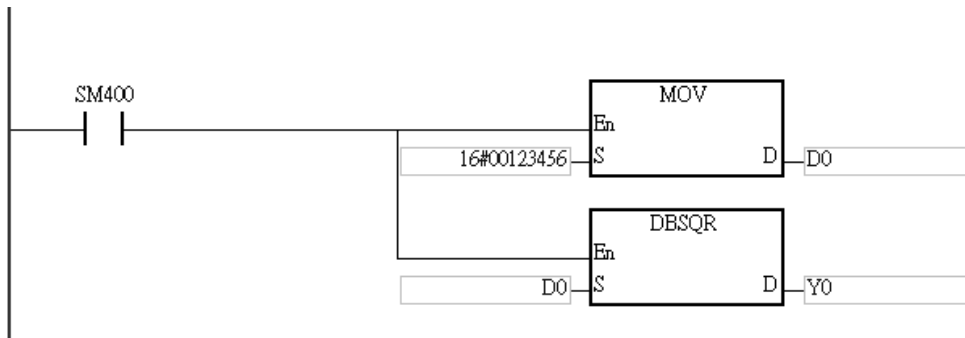
After the square root of the value in D0 is calculated, the integer part is stored in Y0, and the fractional part is stored in Y1.





Example 2:

After the square root of the value in D0 is calculated, the integer part is stored in Y0, and the fractional part is stored in Y1.



Additional remark:

1. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
2. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
3. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of DWORD/DINT.

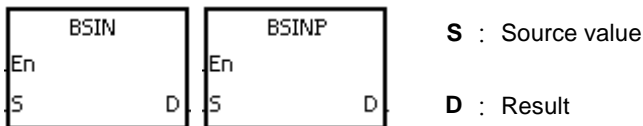
FB/FC	Instruction		Operand	Description
FC	BSIN	P	S, D	Sine of the binary-coded decimal number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●			●	○	●	○	○		
D	●	●			●	●		●			●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

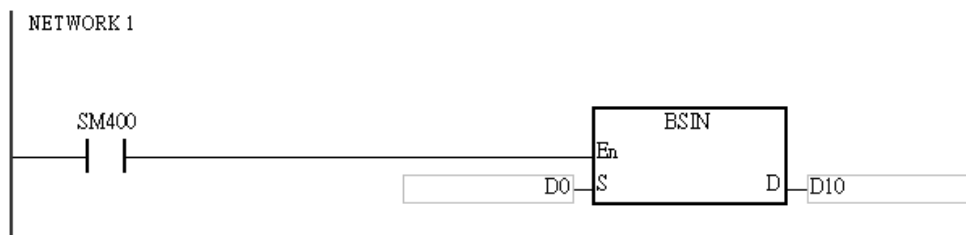


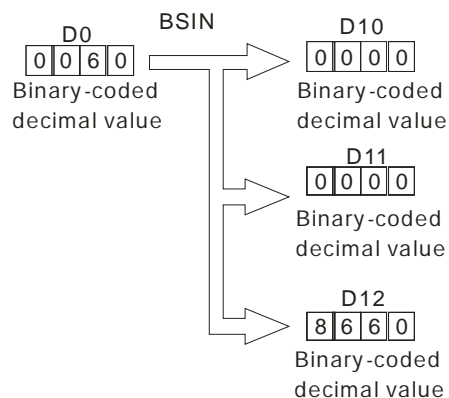
Explanation:

1. The source value specified by **S** is a degree, and the instruction is used to get the sine of the source value specified by **S**. After the sine value is gotten, the sign is stored in **D**, the integer part is stored in **D+1**, and the fractional part is stored in **D+2**.
2. The range of degrees: $0^\circ \leq \text{Degree} < 360^\circ$
3. The operation result is rounded off to the fifth decimal place.
4. If the conversion result is 0, SM600 is ON.

Example:

The instruction is used to get the sine of the value in D0. After the sine value is gotten, the sign is stored in D10, the integer part is stored in D11, and the fractional part is stored in D12.





Additional remark:

1. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
2. If the value in **S** is not within the range between 0° and 360°, the operation error occurs, SM0 is ON, and the error code in SR0 is 16#2003.
3. If users declare the operand **D** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

3

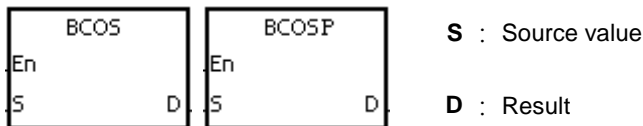
FB/FC	Instruction		Operand				Description					
FC		BCOS	P	S, D				Cosine of the binary-coded decimal number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

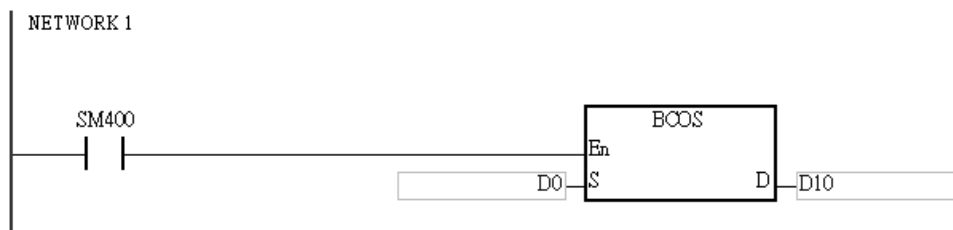


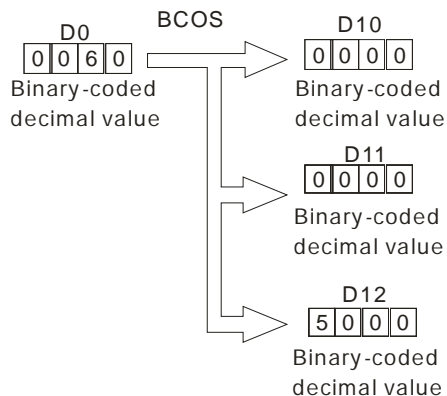
Explanation:

1. The source value specified by **S** is a degree, and the instruction is used to get the cosine of the source value specified by **S**. After the cosine value is gotten, the sign is stored in **D**, the integer part is stored in **D+1**, and the fractional part is stored in **D+2**.
2. The range of degrees: $0^\circ \leq \text{Degree} < 360^\circ$
3. The operation result is rounded off to the fifth decimal place.
4. If the conversion result is 0, SM600 is ON.

Example:

The instruction is used to get the cosine of the value in D0. After the cosine value is gotten, the sign is stored in D10, the integer part is stored in D11, and the fractional part is stored in D12.





Additional remark:

1. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
2. If the value in **S** is not within the range between 0° and 360°, the operation error occurs, SM0 is ON, and the error code in SR0 is 16#2003.
3. If users declare the operand **D** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

3

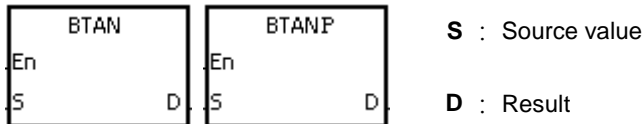
FB/FC	Instruction		Operand	Description
FC	BTAN	P	S, D	Tangent of the binary-coded decimal number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

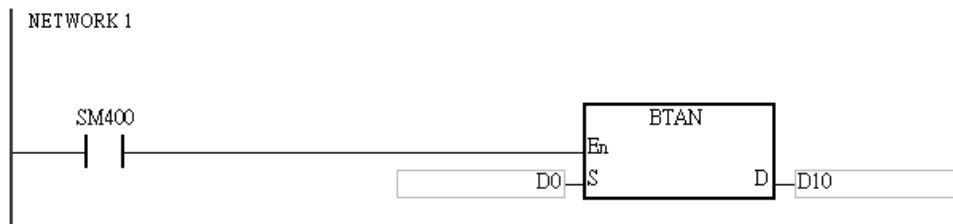


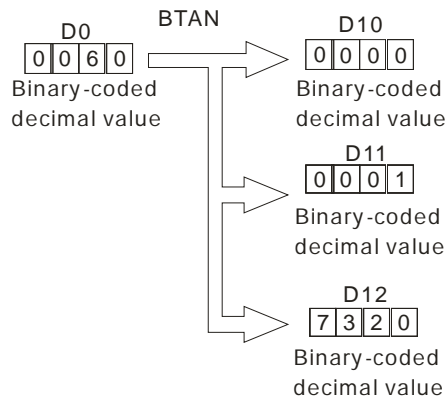
Explanation:

1. The source value specified by **S** is a degree, and the instruction is used to get the tangent of the source value specified by **S**. After the tangent value is gotten, the sign is stored in **D**, the integer part is stored in **D+1**, and the fractional part is stored in **D+2**.
2. The range of degrees: $0^\circ \leq \text{Degree} < 360^\circ$
3. The operation result is rounded off to the fifth decimal place.
4. If the conversion result is 0, SM600 is ON.

Example:

The instruction is used to get the tangent of the value in D0. After the tangent value is gotten, the sign is stored in D10, the integer part is stored in D11, and the fractional part is stored in D12.





Additional remark:

1. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
2. If the value in **S** is not within the range between 0° and 360°, the operation error occurs, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the value in **S** is equal to 90° or 270°, the operation error occurs, SM0 is ON, and the error code in SR0 is 16#2003.
4. If users declare the operand **D** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

3

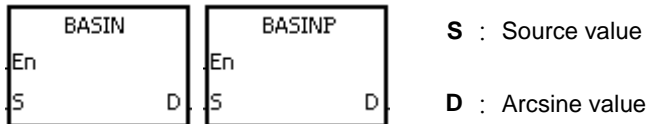
FB/FC	Instruction		Operand				Description						
FC		BASIN	P	S, D				Arcsine of the binary-coded decimal number					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

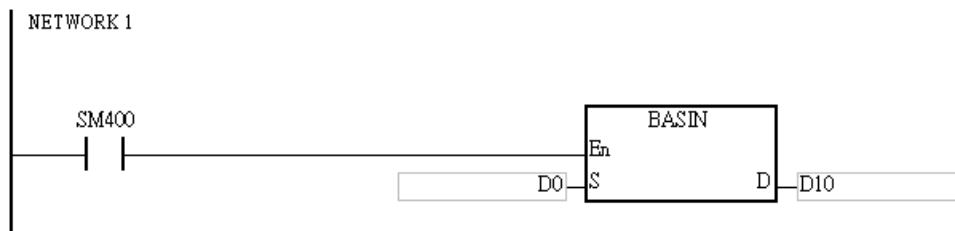


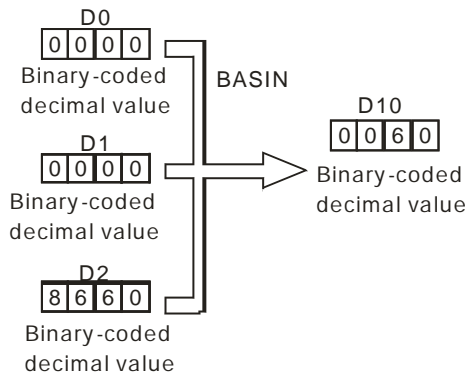
Explanation:

1. The source value specified by **S** is a binary-coded decimal value, and the instruction is used to get the arcsine of the source value specified by **S**. The operation result (the degree) is stored in **D**.
2. The value in **S** represents the sign, i.e. 0 represents the positive sign, and 1 represents the negative sign. The integer part is stored in **S+1**, and the fractional part is stored in **S+2**.
3. The operation result is rounded off to the nearest whole digit.
4. The operation result is a binary-coded decimal value (the degree) within the range between 0° and 90°, or within the range between 270° and 360°.

Example:

The value in D0 represents the sign, the integer part is stored in D1, and the fractional part is stored in D2. After the instruction BASIN is executed, the arcsine value is rounded off to the nearest whole digit, and the result is stored in D10.





Additional remark:

1. Take 0.5 for example. When it is entered, users need to enter 0, 0, and 16#5000 into **S**, **S+1**, **S+2** respectively.
2. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
3. The value specified by the operand **S** should be within the range between -1.0 and +1.0. If the value specified by the operand **S** is not within the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If users declare the operand **S** in ISPSOft, the data type will be ARRAY [3] of WORD/IN.

3

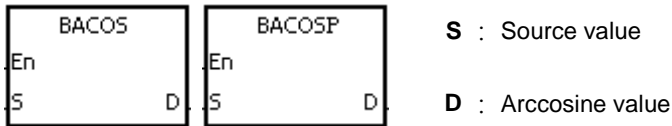
FB/FC	Instruction			Operand				Description				
FC		BACOS	P	S, D				Arccosine of the binary-coded decimal number				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

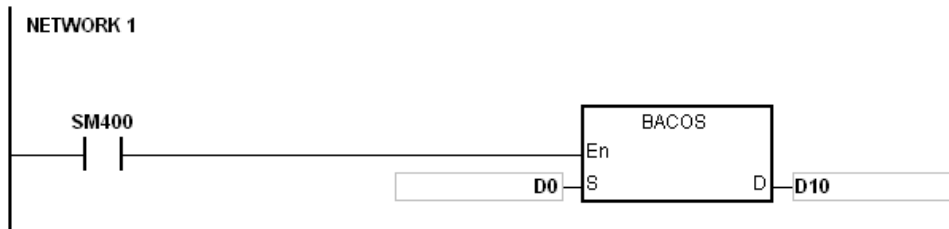


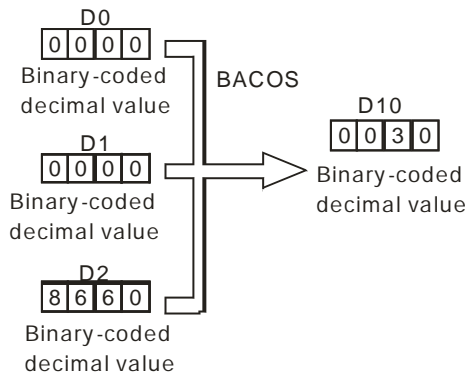
Explanation:

1. The source value specified by **S** is a binary-coded decimal value, and the instruction is used to get the arccosine of the source value specified by **S**. The operation result (the degree) is stored in **D**.
2. The value in **S** represents the sign, i.e. 0 represents the positive sign, and 1 represents the negative sign. The integer part is stored in **S+1**, and the fractional part is stored in **S+2**.
3. The operation result is rounded off to the nearest whole digit.
4. The operation result is a binary-coded decimal value (the degree) within the range between 0° and 180°.

Example:

The value in D0 represents the sign, the integer part is stored in D1, and the fractional part is stored in D2. After the instruction BACOS is executed, the arccosine value is rounded off to the nearest whole digit, and the result is stored in D10.





Additional remark:

1. Take 0.5 for example. When it is entered, users need to enter 0, 0, and 16#5000 into **S**, **S+1**, **S+2** respectively.
2. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
3. The value specified by the operand **S** should be within the range between -1.0 and +1.0. If the value specified by the operand **S** is not within the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If users declare the operand **S** in ISPSOft, the data type will be ARRAY [3] of WORD/IN.

3

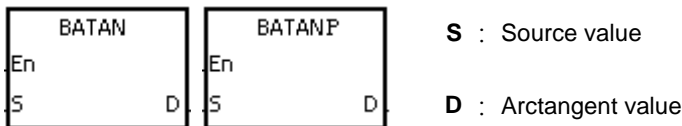
FB/FC	Instruction			Operand				Description					
FC		BATAN	P	S, D				Arctangent of the binary-coded decimal number					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

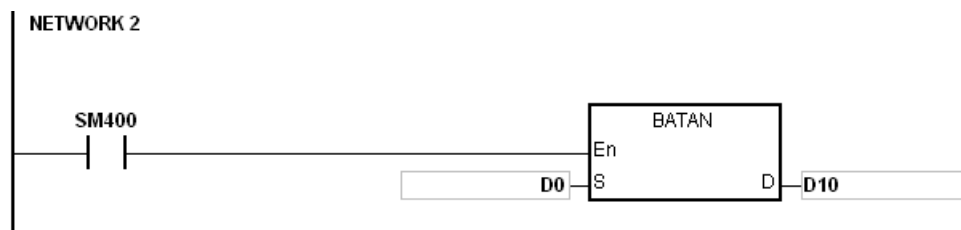


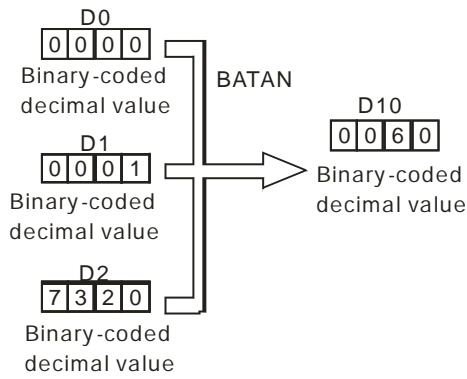
Explanation:

1. The source value specified by **S** is a binary-coded decimal value, and the instruction is used to get the arctangent of the source value specified by **S**. The operation result (the degree) is stored in **D**.
2. The value in **S** represents the sign, i.e. 0 represents the positive sign, and 1 represents the negative sign. The integer part is stored in **S+1**, and the fractional part is stored in **S+2**.
3. The operation result is rounded off to the nearest whole digit.
4. The operation result is a binary-coded decimal value (the degree) within the range between 0° and 90°, or within the range between 270° and 360°.

Example:

The value in D0 represents the sign, the integer part is stored in D1, and the fractional part is stored in D2. After the instruction BATAN is executed, the arctangent value is rounded off to the nearest whole digit, and the result is stored in D10.





Additional remark:

1. Take 0.5 for example. When it is entered, users need to enter 0, 0, and 16#5000 into **S**, **S+1**, **S+2** respectively.
2. If the value in **S** is not a binary-coded decimal value (The binary-coded decimal value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, SM0 is ON, and the error code in SR0 is 16#200D.
3. If users declare the operand **S** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

3

3.19 Real-time Clock Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>TRD</u>	–	✓	Reading the time	3
FC	<u>TWR</u>	–	✓	Writing the time	3
FC	<u>T+</u>	–	✓	Adding the time	7
FC	<u>T-</u>	–	✓	Subtracting the time	7
FC	<u>HOUR</u>	<u>DHOUR</u>	–	Running-time meter	7
FC	<u>TCMP</u>	–	✓	Comparing the time	11
FC	<u>TZCP</u>	–	✓	Time zone comparison	9

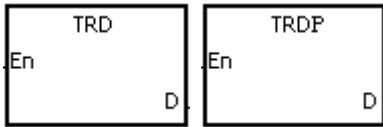
FB/FC	Instruction		Operand				Description							
FC		TRD	P	D				Reading the time						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



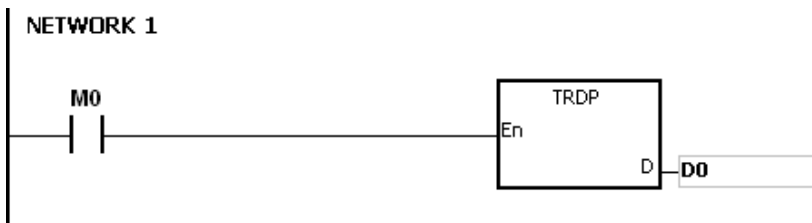
D : Device in which the result

Explanation:

- D**: The device in which the current time is stored
- The operand **D** occupies seven consecutive devices.
- The built-in real-time clock in the CPU module provides the data relating to the year, the week, the month, the day, the minute, and the second. The data is stored in SR391~SR397. The instruction TRD is used to read the current time into the seven registers.
- The first two digits of the year number for A.D. from the right are stored in SR391.

Example:

When M0 is ON, the current time is read from the real-time clock into D0~D6. The value 1 in SR397 represents Monday, the value 2 represents Tuesday, and by analogy, the value 7 represents Sunday.



Special data register	Item	Value
SR391	Year (A.D.)	00~99
SR392	Month	1~12
SR393	Day	1~31
SR394	Hour	0~23
SR395	Minute	0~59
SR396	Second	0~59
SR397	Week	1~7



General data register	Item
D0	Year (A.D.)
D1	Month
D2	Day
D3	Hour
D4	Minute
D5	Second
D6	Week

Additional remark:

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 second in the real-time clock is within the range between 0 and 29, the value of the second is cleared to zero. If the value of the second in the real-time clock is within the range between 30 and 59, the value of the minute increases by one, and the value of the second is cleared to zero.
3. If users declare the operand **D** in ISPSOft, the data type will be ARRAY [7] of WORD/INT.

FB/FC	Instruction		Operand				Description					
FC		TWR	P	S				Writing the time				

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



Explanation:

- S**: The device into which the setting value is written
- The operand S occupies seven consecutive devices.
- When users want to adjust the built-in real-time clock in the CPU module, they can use the instruction to write the correct current time into the built-in real-time clock.
- When the instruction is executed, the new setting time is instantly written into the real-time clock in the PLC. Therefore, when the instruction is executed, users have to make sure that the new setting time is consistent with the time when the new setting time is written into the real-time clock.

Example:

When M0 is ON, the correct current time is written into the built-in real-time clock in the PLC.



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

New setting time Real time clock

Additional remark:

- If the value in **S** exceeds the range, the operation error occurs, the instruction is not executed, SM is ON, and the error code in SR/AR is 16#2003.

2. If S+6 exceeds the device range, the operation error occurs, the instruction is not executed, SM is ON, and the error code in SR/AR is 16#2003.
3. If users declare the operand **S** in ISPSOft, the data type will be ARRAY [7] of WORD/INT.

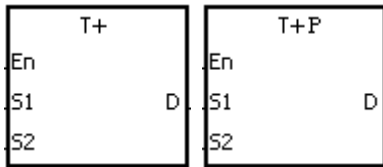
FB/FC	Instruction		Operand				Description				
FC		T+	P	S₁, S₂, D				Adding the time			

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



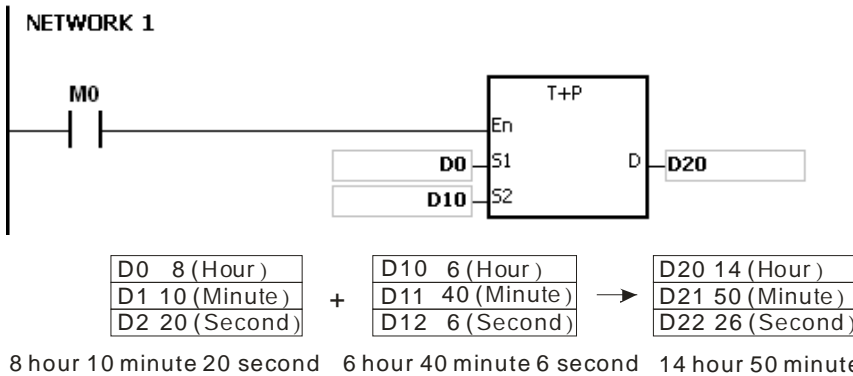
- S₁** : Source device
- S₂** : Source device
- D** : Device in which the result is stored

Explanation:

- The value of the hour, the value of the minute, and the value of the second in the real-time clock specified by S₂ are added to the value of the hour, the value of the minute, and the value of the second in the real-time clock specified by S₁, and the sum is stored in the register specified by D.
- The operands S₁, S₂, and D each occupy three consecutive devices.
- If the sum is larger than or equal to 24 hours, SM602 is ON, and the result gotten from the subtraction of 24 hours from the sum is stored in D.
- If the sum is 0 (0 hour 0 minute 0 second), SM600 is ON.

Example:

When M0 is ON, the instruction T+ is executed. The value of the hour, the value of the minute, and the value of the second in D10~D12 are added to the value of the hour, the value of the minute, and the value of the second in D0~D2, and the sum is stored in D20~D22.



Additional remark:

- If the value in S₁ or S₂ exceeds the range, the operation error occurs, the instruction is not executed, SM0 is

ON, and the error code in SR0 is 16#2003.

2. If S1+2, S2+2, or D+2 exceeds the device range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If users declare the operand S1 in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
4. If users declare the operand S2 in ISPSOft, the data type will be ARRAY [3] of WORD/IN.
5. If users declare the operand D in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

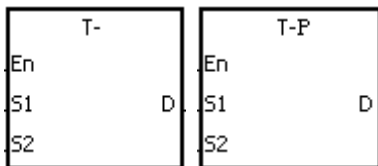
FB/FC	Instruction		Operand			Description
FC	T-	P	S ₁ , S ₂ , D			Subtracting the time

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁ , S ₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁ , S ₂	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



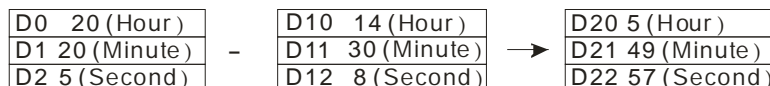
- S₁** : Source device
- S₂** : Source device
- D** : Device in which the result is stored

Explanation:

- The value of the hour, the value of the minute, and the value of the second in the real-time clock specified by **S₂** are subtracted from the value of the hour, the value of the minute, and the value of the second in the real-time clock specified by **S₁**, and the difference is stored in the register specified by **D**.
- The operands **S₁**, **S₂**, and **D** all occupy three consecutive devices.
- If the difference is a negative, SM601 is ON, and the result gotten from the addition of 24 hours to the difference is stored in D.
- If the difference is 0 (0 hour 0 minute 0 second), SM600 is ON.

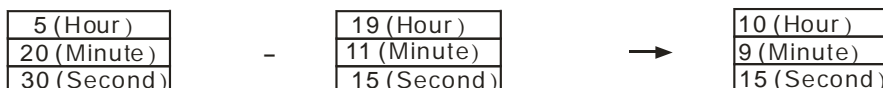
Example:

- When M0 is ON, the instruction T- is executed. The value of the hour, the value of the minute, and the value of the second in D10~D12 are subtracted from the value of the hour, the value of the minute, and the value of the second in D0~D2, and the difference is stored in D20~D22.

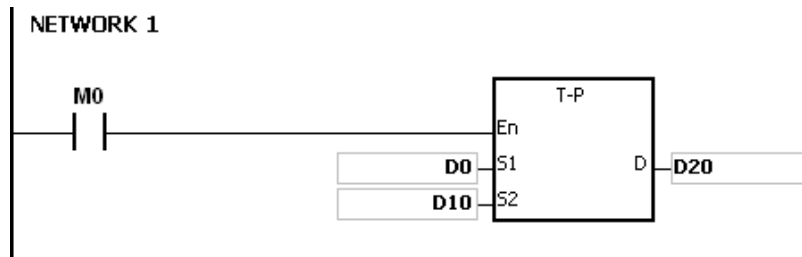


20 hour 20 minute 50 second 14 hour 30 minute 8 second 5 hour 49 minute 57 second

- 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 remark:**

1. If the value in **S₁** or **S₂** exceeds the range, the operation error occurs, the instruction is not executed, **SM0** is ON, and the error code in **SR0** is 16#2003.
2. If **S1+2**, **S2+2**, or **D+2** exceeds the device range, the operation error occurs, the instruction is not executed, **SM0** is ON, and the error code in **SR0** is 16#2003.
3. If users declare the operand **S1** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
4. If users declare the operand **S2** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
5. If users declare the operand **D** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

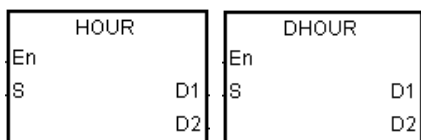
FB/FC	Instruction			Operand			Description		
FC	D*	HOUR		S, D ₁ , D ₂			Running-time meter		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D ₁		●	●*				●	●*						
D ₂	●/●*	●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●	●	●	●		●	○	●	○	○		
D ₁	●	●						●	●		●	○	●				
D ₂	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



- S** : Time after which the output device is ON
- D₁** : Current time
- D₂** : Output device

Explanation:

1. **S**: The time after which the output device is ON (Unit: Hour)
D₁: The current time (Unit: Hour)
D₂: The output device
2. **S**: The time after which the output device is ON (Unit: Hour)
 The operand **S** used in the 16-bit instruction should be within the range between 1 and 32,767.
 The operand **S** used in the 32-bit instruction should be within the range between 1 and 2,147,483,647.
3. The instruction HOUR:
D₁: The current time (Unit: Hour)
 The value in **D₁** should be within the range between 0 and 32,767.
D₁+1: The current time which is less than one hour (Unit: Second)
 The value in **D₁+1** should be within the range between 0 and 3,599.
D₁+2 is for system use only. The value in it can not be altered when the instruction is executed. Otherwise, an error will occur.
 When the current time is 32,767 hour 3,599 second, the timer stops counting. After the values in **D₁** and **D₁+1** are cleared to 0, the timer starts to count again.
4. The instruction DHOUR :
(D₁+1, D₁): The current time (Unit: Hour)
 The value in **(D₁+1, D₁)** should be within the range between 0 and 2,147,483,647.

D_{1+2} : The current time which is less than one hour (Unit: Second)

The value in D_{1+1} should be within the range between 0 and 3,599.

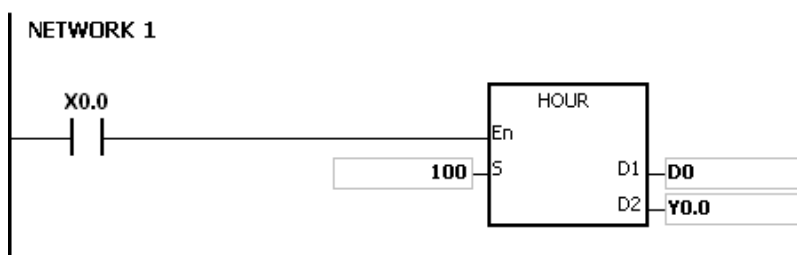
D_{1+3} is for system use only. The value in it can not be altered when the instruction is executed. Otherwise, an error will occur.

When the current time is 2,147,483,647 hour 3,599 second, the timer stops counting. After the values in D_1 , D_{1+1} , and D_{1+2} are cleared to 0, the timer starts to count again.

5. When the time for which the input contact has been ON reaches the setting time, the output device is ON. When the time for which the input contact has been ON does not reach the setting time, the output device is not ON. This function allows users to manage the running time of the machine and the maintenance.
6. After the output device is ON, the timer continues to count.
7. When the on-line editing is used, please reset the conditional contact to initialize the instruction.

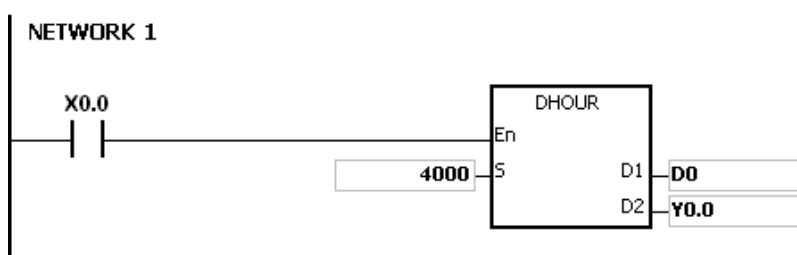
Example 1:

The 16-bit instruction HOUR: When X0.0 is ON, the timer starts to count. When the time for which X0.0 has been ON reaches 100 hours, Y0.0 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 can not be altered. Otherwise, an error will occur.



Example 2:

The 32-bit instruction DHOUR: When X0.0 is ON, the timer starts to count. When the time for which X0.0 has been ON reaches 4000 hours, Y0.0 is ON. The current time is recorded in (D1, D0), and the current time which is less than one hour is recorded in D2. D3 is for system use. The value in it can not be altered. Otherwise, an error will occur.



Additional remark:

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 D1 used in the instruction HOUR is less than 0, the state of the output device is unchanged.
3. If D_{1+2} used in the instruction HOUR exceeds the device range, the 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_{1+1} , D1) used in the instruction DHOUR is less than 0, the state of the output device is unchanged.
5. If D_{1+3} used in the instruction DHOUR exceeds the device range, the operation error occurs, the instruction

is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

6. If the operand D1 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
7. If the operand D1 used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of DWORD/DINT.

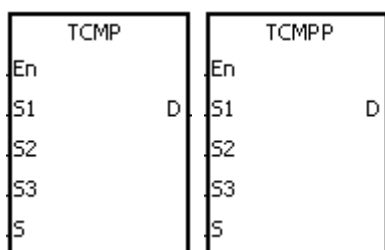
FB/FC	Instruction			Operand				Description				
FC		TCMP	P	S₁, S₂, S₃, S, D				Comparing the time				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂, S₃		●					●							
S		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂, S₃	●	●			●	●		●	●		●	○	●	○	○		
S	●	●			●	●		●	●		●	○	●				
D	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



- S₁** : Hour of the setting time
S₂ : Minute of the setting time
S₃ : Second of the setting time
S : Current time
D : Comparison result

Explanation:

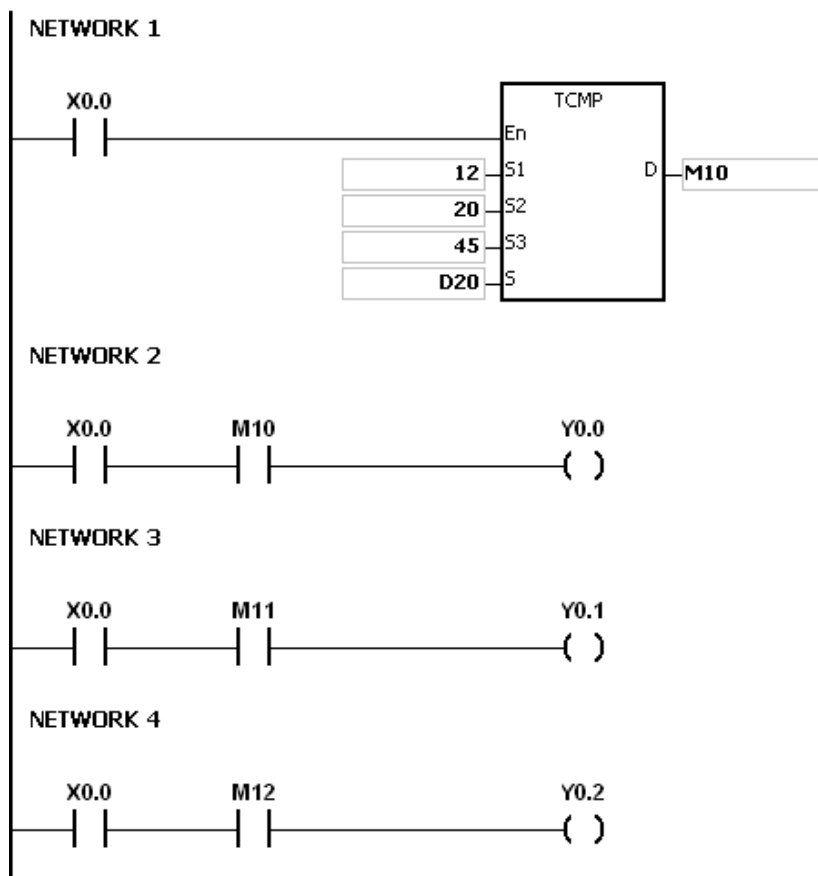
- The value of the hour, the value of the minute, and the value of the second specified by **S₁~S₃** are compared with the value of the hour, the value of the minute, and the value of the second in the devices starting from the device specified by **S**, and the comparison result is stored in **D**.
- The hour of the current time is in the device specified by **S**, and the value of the hour should be within the range between 0 and 23. The minute of the current time is in the device specified by **S+1**, and the value of the minute should be within the range between 0 and 59. The second of the current time is in the device specified by **S+2**, and the value of the second should be within the range between 0 and 59.
- The operand **D** occupies three consecutive devices. The comparison result is stored in D, D+1, and D+2.
- Users generally use the instruction TRD to read the current time from the real-time clock first, and then they use the instruction TCMP to compare the time.
- If the setting time in S1~S3 is larger than the current time in S, D is ON, D+1 is OFF, and D+2 is OFF.
- If the setting time in S1~S3 is equal to the current time in S, D is OFF, D+1 is ON, and D+2 is OFF.
- If the setting time in S1~S3 is less than the current time in S, **D** is OFF, **D+1** is OFF, and **D+2** is ON.

Example:

- When X0.0 is ON, the instruction is executed. The setting time 12 hour 20 minute 45 second is compared with the current time in D20~D22, and the comparison result is stored in M10~M12. When X0.0 is switched from ON to OFF, the instruction is not executed. Besides, the state of M10, the state of M11, and the state of

M12 remain the same as those before X0.0's being ON.

- If users want to get the comparison result \geq , \leq , or \neq , they can connect M10~M12 in series or in parallel.



Additional remark:

- If **S**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If **D**+2 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If the value in **S** exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If the values in **S1**~**S3** exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
- If users declare the operand **D** in ISPSOft, the data type will be ARRAY [3] of BOOL.

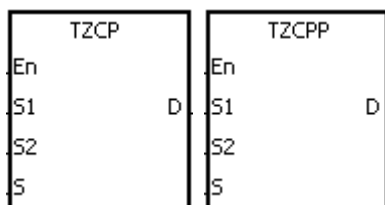
FB/FC	Instruction			Operand				Description				
FC		TZCP	P	S₁, S₂, S, D				Time zone comparison				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂, S		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂, S	●	●			●	●		●	●		●	○	●				
D	●	●	●	●				●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : Lower limit time

S₂ : Upper limit time

S : Current time

D : Comparison result

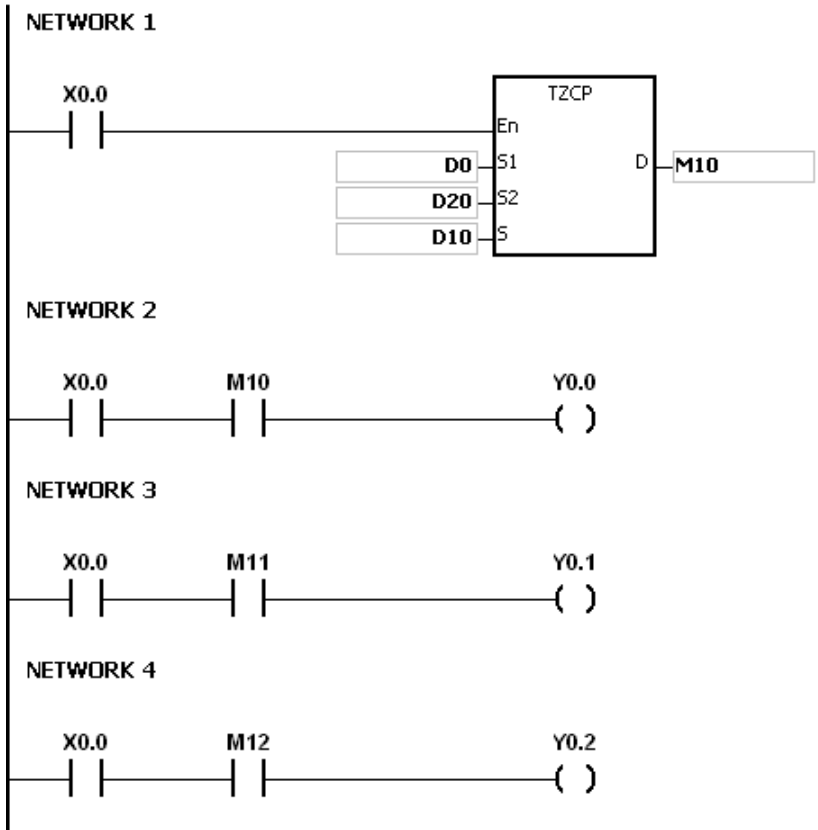
Explanation:

- The instruction is used to compare the current time specified by **S** with the lower limit time specified by **S₁**, and compare the current time specified by **S** with the upper limit time specified by **S₂**, and the comparison result is stored in **D**.
- The hour of the lower limit time is in the device specified by **S₁**, the minute of the lower limit time **e** is in the device specified by **S₁+1**, and the second of the lower limit time is in the device specified by **S₁+2**.
- The hour of the upper limit time is in the device specified by **S₂**, the minute of the upper limit time **e** is in the device specified by **S₂+1**, and the second of the upper limit time is in the device specified by **S₂+2**.
- The hour of the current time is in the device specified by **S**, the minute of the current time **e** is in the device specified by **S+1**, and the second of the current time is in the device specified by **S+2**.
- The time in the device specified by **S₁** must be less than the time in the device specified by **S₂**. If the time in the device specified by **S₁** is larger than the time in the device specified by **S₂**, the time in the device specified by **S₁** will be taken as the upper/lower limit time during the execution of the instruction **TZCP**.
- Users generally use the instruction **TRD** to read the current time from the real-time clock first, and then they use the instruction **TZCP** to compare the time.
- If the current time in the device specified by **S** is less than the lower limit time in the device specified by **S₁**, and is less than the upper limit time in the device specified by **S₂**, **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 **S₁**, and is larger than the upper limit time in the device specified by **S₂**, **D+2** is ON. In other conditions, **D+1** is ON.

Example:

When **X0.0** is ON, the instruction **TZCP** is executed. **M10**, **M11**, or **M12** is ON. When **X0.0** is OFF, the instruction **TZCP** is not executed, the state of **M10**, the state of **M11**, and the state of **M12** remain the same as those before

X0.0's being ON.



Additional remark:

1. If S₁+2, S₂+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 S1, S2, and S exceed the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003
3. If users declare the operand S1 in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
4. If users declare the operand S2 in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
5. If users declare the operand S in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
6. If users declare the operand D in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

3.20 Peripheral Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>TKY</u>	<u>DTKY</u>	–	Ten-key keypad	7
FC	<u>HKY</u>	<u>DHKY</u>	–	Sixteen-key keypad	9
FC	<u>DSW</u>	–	–	DIP switch	9
FC	<u>ARWS</u>	–	–	Arrow keys	9
FC	<u>SEGL</u>	–	–	Seven-segment display with latches	7

FB/FC	Instruction			Operand			Description		
FC	D*	TKY		S, D ₁ , D ₂			Ten-key keypad		

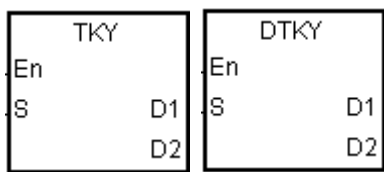
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S	●/●*													
D ₁		●	●*				●	●*						
D ₂	●/●*													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●	●	●				●	●				●				
D ₁	●	●			●	●		●	●		●	○	●				
D ₂		●	●	●				●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

3

Graphic expression:



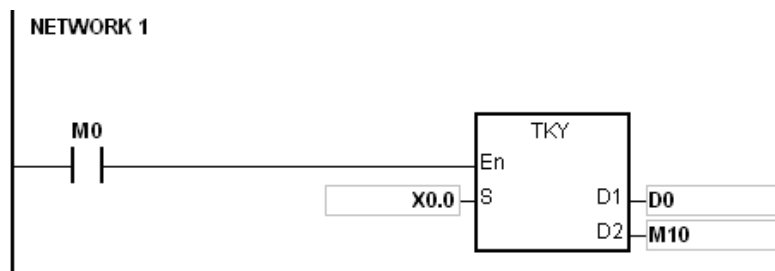
- S : Initial device
- D₁ : Device in which the value is stored
- D₂ : Output signal

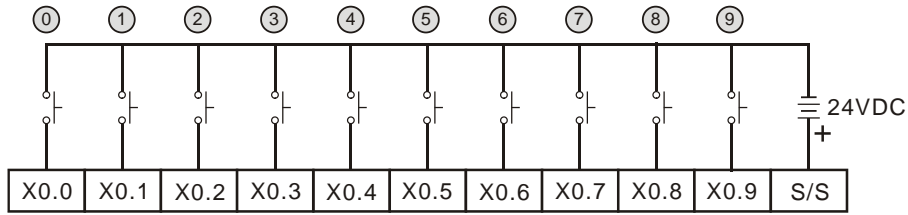
Explanation:

- The ten external inputs starting from the input specified by S represents 0~9 in the decimal system. They are connected to ten keys. Users can enter a four-digit decimal value or an eight-digit decimal value by pressing the keys in order. The decimal value is stored in D₁, and the output signals are stored in D₂.
- The operand S occupies ten bits.
- The operand D₂ occupies eleven bits. Please do not change the states of the bits during the execution of the instruction.
- When the conditional contact is not enabled, the eleven bits starting from the bit specified by D₂ is OFF.
- When the on-line editing is used, please reset the conditional contact to initialize the instruction.

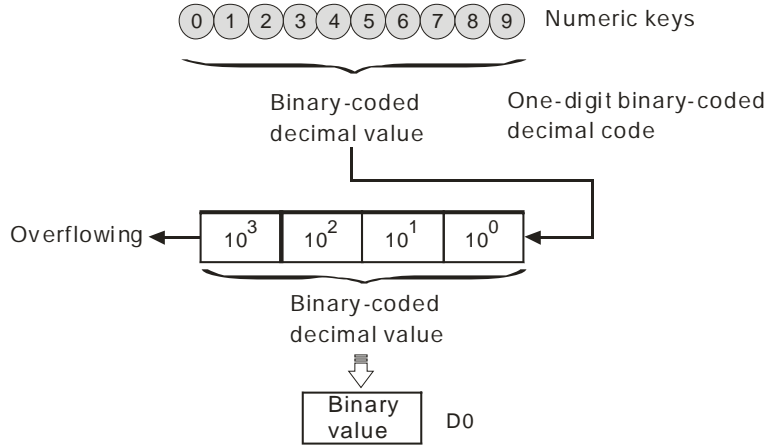
Example:

- The ten external inputs starting from X0.0 is connected to ten keys which represent 0~9 in the decimal system. When M0 is ON, the instruction is executed. The value that users enter is stored as a binary value in D0, and the output signals are stored in M10~M19.

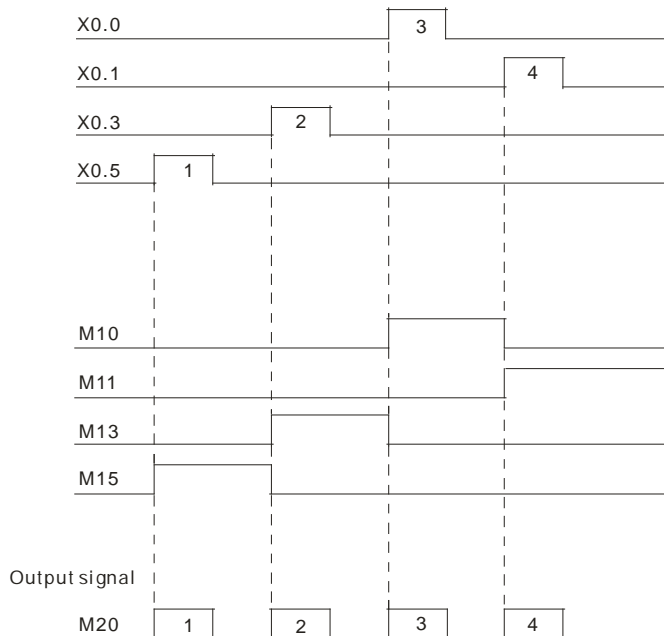




Note: The digital input module AH16AM10N-5A is used in this example.



2. If the keys connected to X0.5, X0.3, X0.0, and X0.1 are pressed in the order shown in the timing chart, the result 5,301 is stored in D0. The maximum value which 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 X0.2 is pressed and before other keys are pressed, M12 is ON. The same applies to other keys.
4. When a key connected to the input within the range between X0.0 and X0.9 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 which was stored in D0 is unchanged. However, M10~M20 are switched OFF.



Additional remark:

1. If users declare the operand **S** in ISPSoft, the data type will be ARRAY [10] of BOOL.
2. If users declare the operand D2 in ISPSoft, the data type will be ARRAY [11] of BOOL.

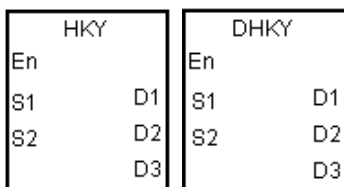
FB/FC	Instruction			Operand				Description				
FC	D*	HKY		S, D ₁ , D ₂ , D ₃				Sixteen-key keypad				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁	●/●*													
S ₂		●/●*					●/●*							
D ₁	●/●*													
D ₂			●*				●	●*						
D ₃	●/●*													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●																
S ₂	●	●			●	●		●	●				●				
D ₁		●															
D ₂	●	●			●	●		●	●		●	○	●				
D ₃		●	●	●				●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	AH Motion CPU

Graphic expression:



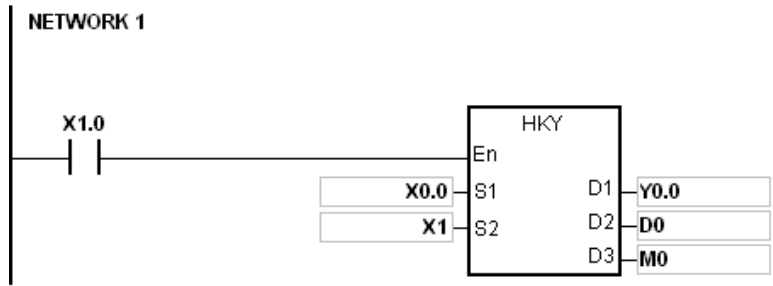
- S₁ : Initial input device
- S₂ : For system use only
- D₁ : Initial output device
- D₂ : Device in which the value is stored
- D₃ : Output signal

Explanation:

- The four external inputs starting from the input specified by **S** are connected to the four external outputs starting from the output specified by **D₁** to form a 16-key keypad. The value that users enter by pressing the keys is stored in **D₂**, and the output signals are stored in **D₃**. If several keys are pressed simultaneously, the value which is smaller is stored.
- The value that users enter by pressing the keys is temporarily stored in **D₂**. If the 16-bit instruction HKY is executed, the maximum value which can be stored in **D₂** is 9,999. If the value exceeds four digits, the first digit from the left overflows. If the 32-bit instruction DHKY is executed, the maximum value which can be stored in **D₂** is 9,999. If the value exceeds eight digits, the first digit from the left overflows.
- After the execution of the instruction is complete, SM692 is ON. That is to say, SM692 is ON for a scan cycle after the execution of the matrix scan is complete.

Example:

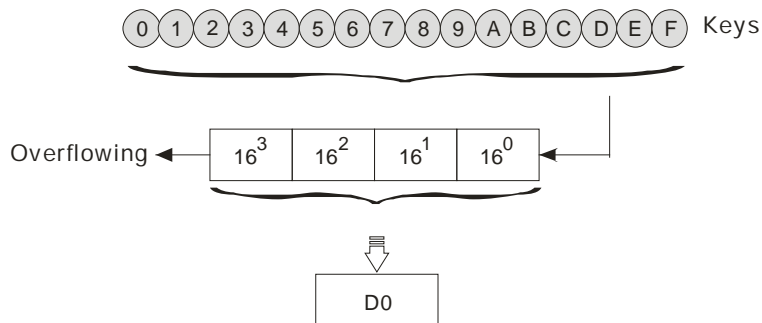
- The four external inputs X0.0~X0.3 are connected to the four external outputs Y0.0~Y0.3 to form a 16-key keypad. When X1.0 is ON, the instruction is executed. The value that users enter is stored as a binary value in D0, and the output signals are stored in M0~M7.



2. The function of SM691:

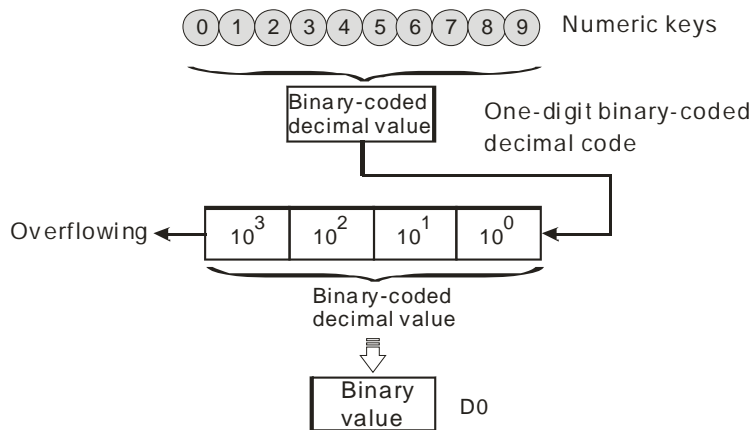
- If SM691 is ON, 0~F are taken as hexadecimal values in the execution of the instruction HKY.

Numeric keys:



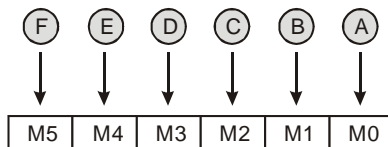
- If SM691 is OFF, A~F are taken as function keys in the execution of the instruction HKY.

Numeric keys:



Function keys:

- When A is pressed, M0 keeps ON. When D is pressed, M0 is switched OFF, and M3 keeps ON.
- If several function keys are pressed, the key which is pressed first has priority.

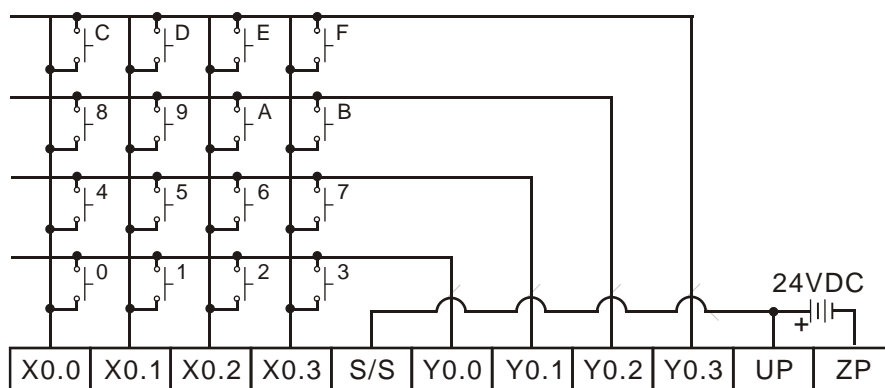


3. Output signals:

- 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.

4. When the conditional contact X1.0 is switched OFF, the value which was stored in D0 is unchanged. However, M0~M7 are switched OFF.

5. The external wiring:



Note: The transistor output module AH16AP11T-5A is used in this example.

Additional remark:

1. When the instruction is executed, errors could occur due to an improper scan time which is too long or too short. To avoid this error, you can adjust the scan time according to the suggestions below:
 - When the scan time is too short, the I/O state might not be real-time updated. In this case, you can fix the scan time at a proper length to allow the key input signal to be updated.
 - When the scan time is too long, the key input signal could be delayed. In this case, you can use this instruction in a timed interrupt task to update the key input signal in a certain time interval.
2. If users declare the operand **S** in ISPSOft, the data type will be ARRAY [4] of BOOL.
3. If users declare the operand **D1** in ISPSOft, the data type will be ARRAY [4] of BOOL.
4. If users declare the operand **D3** in ISPSOft, the data type will be ARRAY [8] of BOOL.

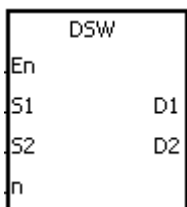
FB/FC	Instruction		Operand				Description				
FC	DSW		S, D₁, D₂, n				DIP switch				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁	●													
S ₂		●					●							
D ₁	●													
D ₂		●					●							
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●																
S ₂	●	●			●	●		●	●				●				
D ₁		●															
D ₂	●	●			●	●		●	●				●				
n	●	●						●	●			●	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S₁ : Initial input device
- S₂ : For system use only
- D₁ : Initial output device
- D₂ : Device in which the value is stored
- n : Number of DIP switches

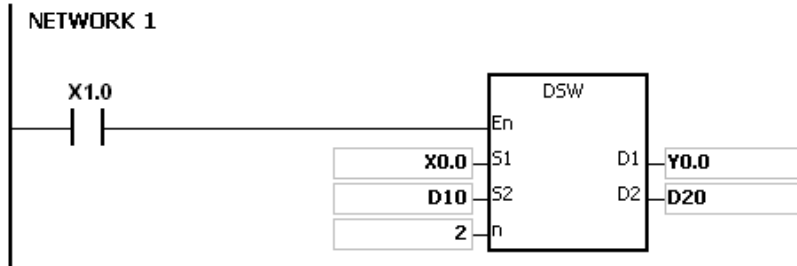
Explanation:

- The four or eight external inputs starting from the input specified by S₁ are connected to the four external outputs starting from the output specified by D₁ to form a four-digit DIP switch or two four-digit DIP switches. The value that users enter by pressing the DIP switch is stored in D₂. Whether there is one four-digit DIP switch or two four-digit DIP switches depends on n.
- If n is 1, the operand D2 occupies one register. If n is 2, the operand D2 occupies two registers.
- S2 and S2+1, which are for system use only, occupy two devices. Please do not alter the values in these devices.
- After the execution of the instruction is complete, SM694 is ON for a scan cycle.
- When the conditional contact is not enabled, the four external outputs starting from the output specified by D1 keep OFF.
- When the on-line editing is used, please reset the conditional contact to initialize the instruction.

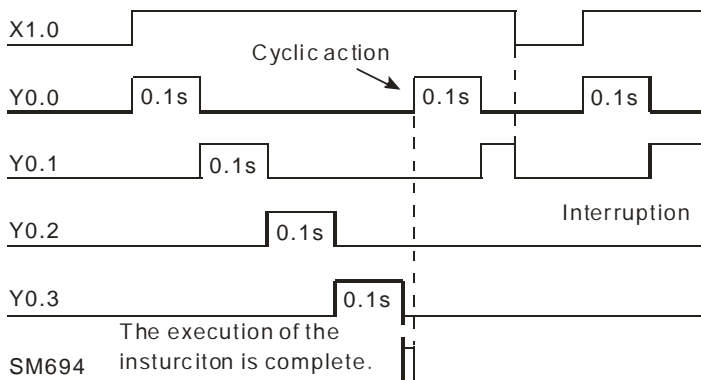
Example:

- X0.0~X0.3 are connected to Y0.0~Y0.3 to form the first DIP switch, and X0.4~X0.7 are connected to

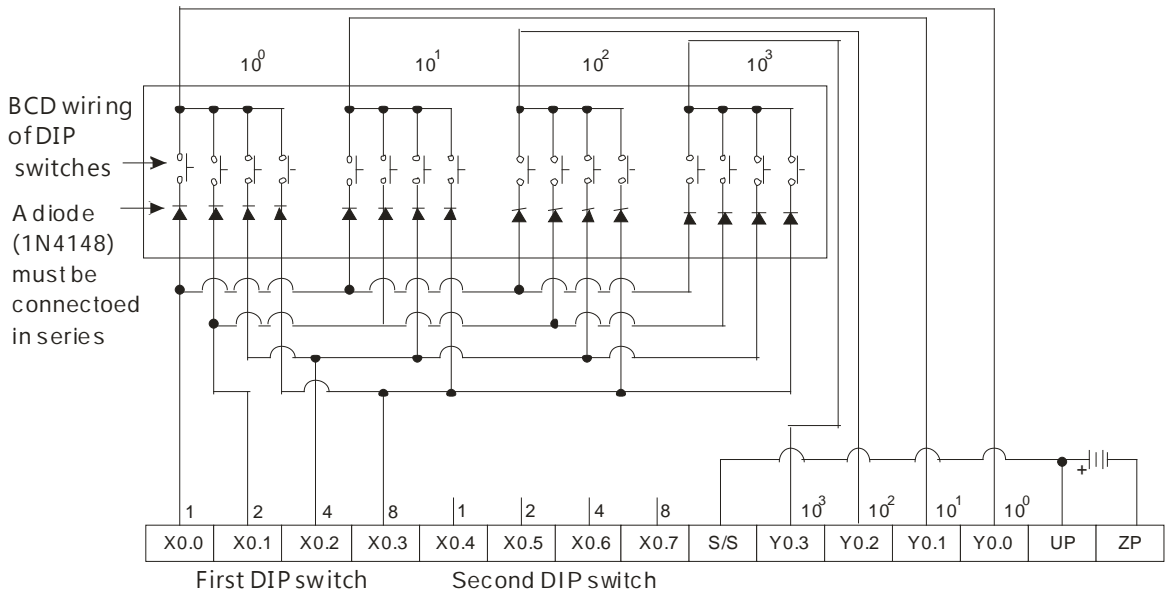
Y0.0~Y0.3 to form the second DIP switch. When X1.0 is ON, the instruction is executed. The value that users enter by pressing the first DIP switch is converted into the binary value, and the conversion result is stored in D20. The value that users enter by pressing the second DIP switch is converted into the binary value, and the conversion result is stored in D21.



2. When X1.0 is ON, Y0.0~Y0.3 are ON cyclically. After the execution of the instruction is complete, SM694 is ON for a scan cycle.
3. The outputs Y0.0~Y0.3 must be transistors.



4. The DIP switches:



Note: the transistor output module AH16AP11T is used in this example.

Additional remark:

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 users declare the operand D1 in ISPSOft, the data type will be ARRAY [4] of BOOL.

FB/FC	Instruction		Operand				Description				
FC		ARWS	S₁, S₂, D₁, D₂, n				Arrow keys				

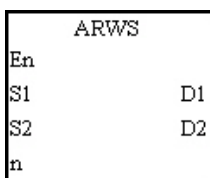
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁	●													
S ₂		●					●							
D ₁		●					●							
D ₂	●													
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●	●	●				●	●				●				
S ₂	●	●			●	●		●	●				●				
D ₁	●	●			●	●		●	●			○	●				
D ₂		●															
n	●	●						●	●		●		●	○	○		

3

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S₁ : Initial input device
- S₂ : For system use only
- D₁ : Device in which the setting value is stored
- D₂ : Initial output device
- n : Positive/Negative logic

Explanation:

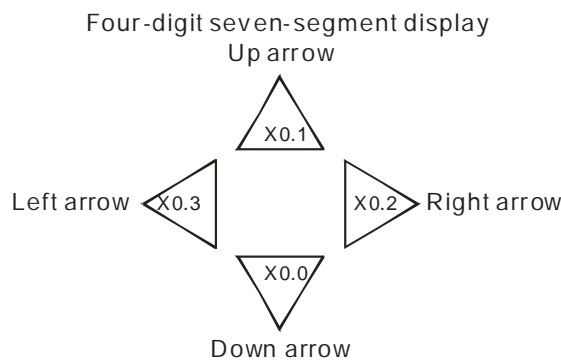
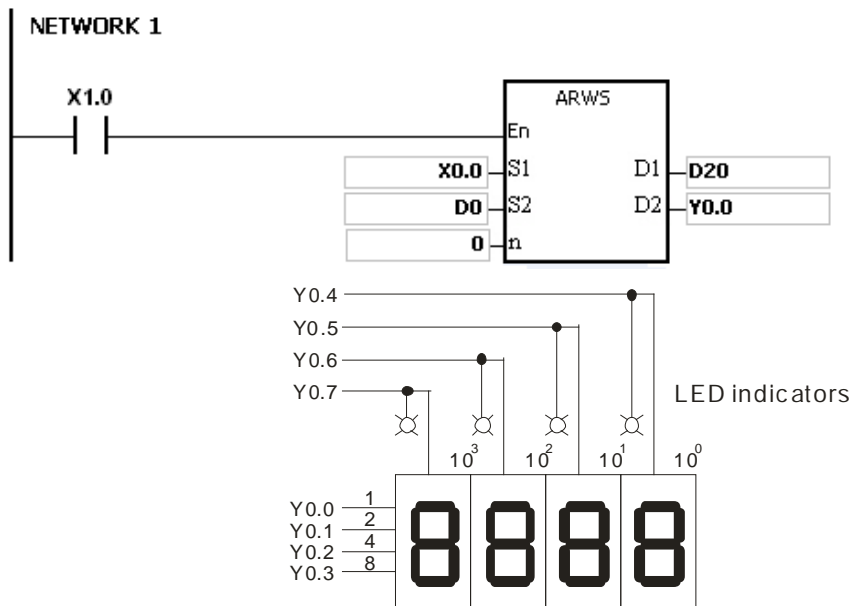
- If the instruction is executed, S₁ is defined as the down arrow, S₁+1 is defined as the up arrow, S₁+2 is defined as the right arrow, and S₁+3 is defined as the left arrow. The setting value is stored in D1, and it should be within the range between 0 and 9,999.
- The operand S1 occupies four consecutive bit devices.
- S2 is for system use only. Please do not alter the value in it.
- The operand D2 occupies eight consecutive bit devices.
- When the conditional contact is not enabled, the eight bit devices starting from the bit device specified by D2 keep OFF.
- The operand n should be within the range between 0 and 3. Please refer to the additional remark on the instruction SEGL for more information.
- When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example:

- If the instruction is executed, X0.0 is defined as the down arrow, X0.1 is defined as the up arrow, X0.2 is

defined as the right arrow, and X0.3 is defined as the left arrow. The setting value is stored in D20, and it should be within the range between 0 and 9,999.

2. When X1.0 is ON, the digit in the place 10³ is selected. If the left arrow is pressed, the places are selected in sequence (10³→10⁰→10¹→10²→10³→10⁰).
3. If the right arrow is pressed, the places are selected in sequence (10³→10²→10¹→10⁰→10³→10²). The LED indicators with the corresponding places are connected to Y0.4~Y0.7. When the digits in the places are selected in sequence, the LED indicators are ON in sequence.
4. If the up arrow is pressed, the digit in the place selected changes (0→1→2→...8→9→0→1). If the down arrow is pressed, the digit in the place selected changes (0→9→8→...1→0→9). The new digit is shown on seven-segment display.



The four keys are used to select the place and change the digit

Additional remark:

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 users declare the operand S1 in ISPSOft, the data type will be ARRAY [4] of BOOL.
3. If users declare the operand D2 in ISPSOft, the data type will be ARRAY [8] of BOOLL.

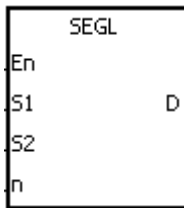
FB/FC	Instruction			Operand			Description		
FC	SEGL			S, D, n			Seven-segment display with latches		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							
D	●													
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●	●		●	○	●				
S ₂	●	●			●	●		●	●				●				
D		●															
n	●	●						●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



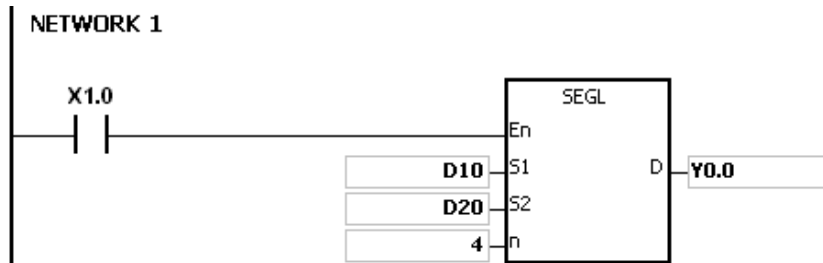
- S₁ : Source device
- S₂ : For system use only
- D : Initial output device
- n : Positive/Negative logic

Explanation:

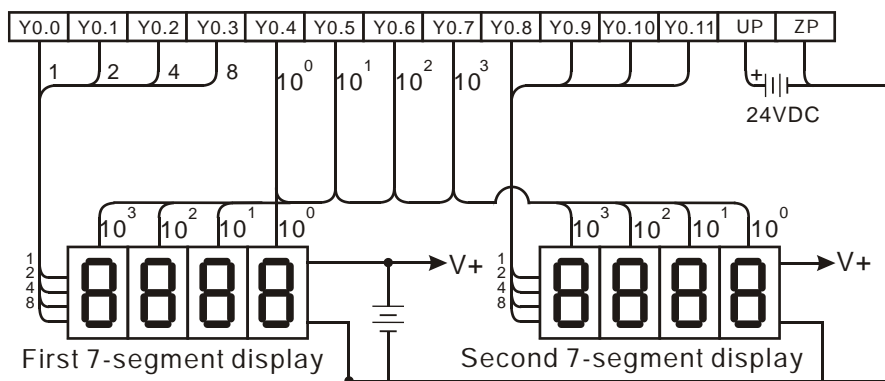
- The eight external outputs starting from the output specified by D are connected to a four-digit seven-segment display, or the twelve external outputs starting from the output specified by D are connected to two four-digit seven-segment displays. Every place is equipped with a driver which converts a binary-coded decimal value into seven-segment data, and every driver is equipped with a latch which can be used to store state information.
- The value in S1 is the value which will be shown on first seven-segment display, and the value in S1+1 is the value which will be shown on second seven-segment display.
- S2 is for system use only. Please do not alter the value in it.
- The operand n should be within the range between 0 and 7. Please refer to the additional remark for more information.
- 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 depend on n.
- 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.
- When the instruction is executed, the outputs are ON cyclically. If the conditional contact is switched from OFF to ON during the execution of the instruction, the outputs are ON cyclically again.
- After the execution of the instruction is complete, SM693 is ON for a scan cycle.

Example:

- When X1.0 is ON, the instruction is executed. Y0.0~Y0.4 form a circuit. The value in D10 is converted into the binary-coded decimal value, and the conversion result is shown on first seven-segment display. The value in D11 is converted into the binary-coded decimal value, and the conversion result is shown on second seven-segment display. If the value in D10 or D11 exceeds 9,999, the operation error occurs.



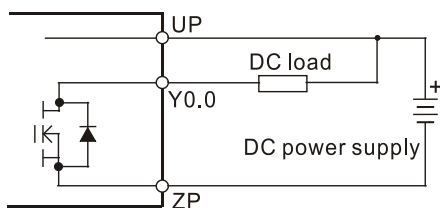
- When X1.0 is ON, Y0.4~Y0.7 are ON cyclically. It takes twelve scan cycles for Y0.4~Y0.7 to be ON. After the execution of the instruction is complete, SM693 is ON for a scan cycle.
- If there is on four-digit seven-segment display, n is within the range between 0 and 3.
 - After the pins 1, 2, 4, and 8 are connected in parallel, they are connected to Y0.0~Y0.3 on the PLC, and the latches are connected to Y0.4~Y0.7 on the PLC.
 - When X1.0 is ON, the instruction is executed. Y0.4~Y0.7 are ON cyclically, and the value in D10 is shown on seven-segment display.
- If there are two four-digit seven-segment displays, n is within the range between 4 and 7.
 - After the pins 1, 2, 4, and 8 are connected in parallel, they are connected to Y0.8~Y0.11 on the PLC, and the latches are connected to Y0.4~Y0.7 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.
- The wiring:



Note: The transistor output module AH16AN01T-5A is used in this example.

Additional remark:

- 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. Besides, an output has to connect a pull-up resistor to the DC power supply (less than 30 V DC). Therefore, when an output is ON, a signal of low potential is output.



- 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	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 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	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 latch:

Positive logic		Negative logic	
Latch	Signal	Latch	Signal
1	0	0	1

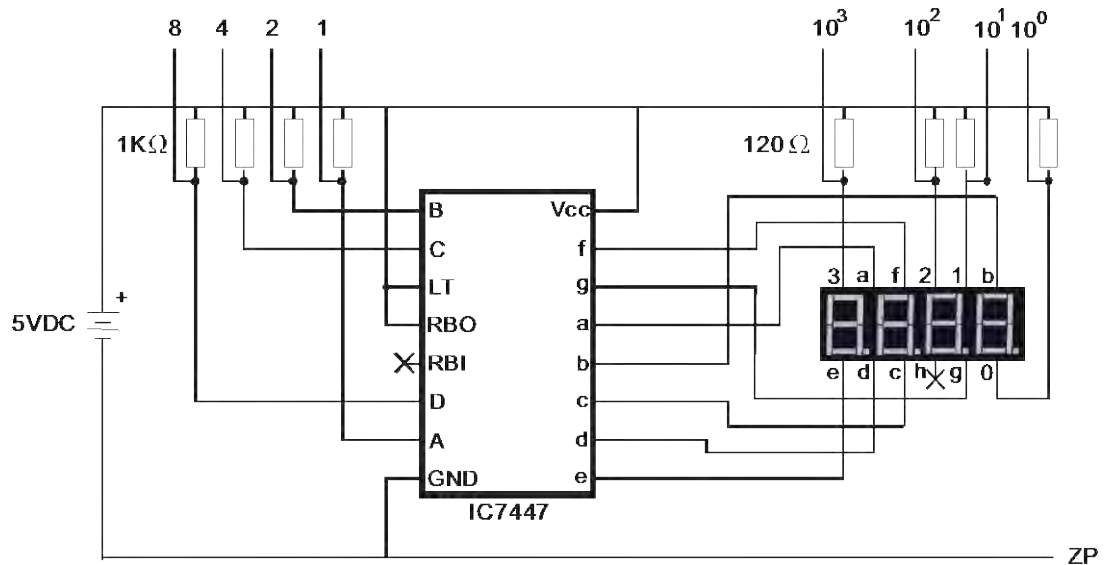
- The setting value of the parameter n:

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

- The connection of the common-anode four-digit seven-segment display with IC 7447 is as follows.



ZP

3.21 Communication Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>RS</u>	–	–	Transmitting the user-defined communication command	9
FC	<u>LRC</u>	–	✓	Longitudinal parity check	7
FC	<u>CRC</u>	–	✓	Cyclic Redundancy Check	7
FC	<u>MODRW</u>	–	–	Reading/Writing the MODBUS data	11
FC	<u>COMRS</u>	–	–	Sending and receiving communication data	11

FB/FC	Instruction			Operand				Description						
FC		RS		S, m, D, n				Transmitting the user-defined communication command						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●				●				
m	●	●			●	●		●	●				●	○	○		
D	●	●			●	●		●	●				●				
n	●	●			●	●		●	●				●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S** : Initial transmission device
- m** : Number of data which is sent
- D** : Initial reception device
- n** : Number of data which is received

Explanation:

- The instruction is for the CPU module equipped with RS-232/422/485. Data can be sent and received when the data **S, m, D,** and **n** are set. When the index register E is used as a modifier for the initial transmission device, do not alter the values in the E registers during the execution of the instruction. Otherwise, a read error or a write error will occur.
- You can set **m** to 0 if only data receiving is needed, or set **n** to 0 if only data transmission is needed.
- The instruction can be used for several times in the program, but when using different instructions such as MODRW and FWD in the same communication port, only one instruction can be executed at a time.
- During the execution of the RS transmission, the alteration of the sending or receiving data cannot be made.
- The length of the data transmission (**m** and **n**) cannot be greater than 500 words.
- Via a special flag (SM106 or SM107), 8-bit mode or 16-bit mode can be selected.
- If the devices to be connected support MODBUS communication protocol, you can use MODRW instruction for convenient communication. Please refer to the detailed descriptions of MODRW instruction.

Communication Format Setup

Before executing the serial communication instructions, users need to setup the communication format including the RS-232/485, the baud rate and more. There are 2 methods to set the communication format, users can set the

PLC communication port directly via the HWCONFIG or set the communication format according to the specific special auxiliary relays in the program.

1. Set the communication format via the HWCONFIG for the communication port. (Please refer to ISPSOft User Manual for the setups.)
2. Set the communication format according to the special auxiliary relays in the program. Please refer to the additional remarks for the setups, examples and the register formats.

Descriptions of the Data Transmission Format

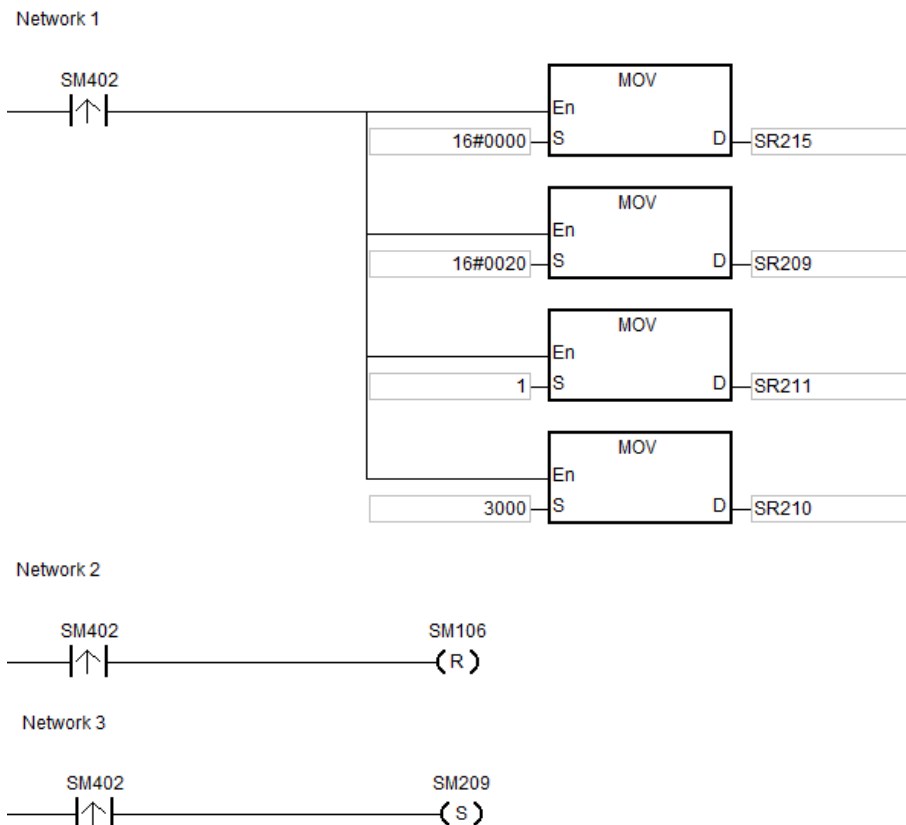
Data transmission formats include 8-bit mode and 16-bit mode. At 8-bit mode, the 16-bit data is divided into the high 8-bit data and the low 8-bit data. The high 8-bit data is ignored, and the low 8-bit data can be sent or received. When at 16-bit mode, the placements of the high and low bit are very important, since the result will be affected.

8-bit Mode · Data Transmission 0x01234567							
D10 (High)	D10 (Low)	D11 (High)	D11 (Low)	D12 (High)	D12 (Low)	D13 (High)	D13 (Low)
*	16#01	*	16#23	*	16#45	*	16#67
16-bit Mode · Data Transmission 0x1234567							
D10		D11		D12		D13	
16#2301		16#6745					

Examples for Setting up the Communication Format: RS-232, 9600, 7, N, 1.

1. Use the communication port RS-232 (SR215=0).
2. Set the baud rate and format to 9600, 7, N, 1 (SR209=16#0020).
3. Set the number of times the command is resent to 1 (SR211=1).
4. Set to communication timeout to 3000ms (SR210=3000).
5. Set the mode to 16-bit (SM106=OFF).
6. The setup is successful (SM209=ON).

The configurations can be done via ISPSOft → HWCONFIG → COM Port, and these steps can be ignored.



Example:

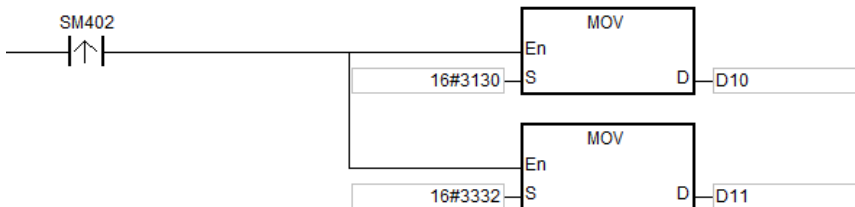
1. Set the communication format via the HWCONFIG or the special registers for the communication port. (As the above example stated.)
2. Write the transmission data in the registers of D10 and D11 and then set the SM96 (sending request flag) to ON.
3. When X0.1 and X0.3 are set to ON, the RS commands the PLC to be in the standby status, waiting for data to be sent and received. When in execution, the D10 will start to send n bytes of data to the external device. When the transmission is done, the SM96 will be reset automatically. (Please do not use RST instruction to reset SM96 in the PLC program.) When data is received, it will be saved in the successive registers, starting from D100.
4. When the reception is complete, the flag SM100 will be switched to ON automatically. Then users can start managing the data. After the management is done, reset the SM100 to OFF to make it available for further transmissions. Please do not use RST instruction to reset SM100 continuously in the PLC program.
5. When the PLC receives any command containing the special character in SR621, the system will trigger I32 for interruption and D30 will increase by 1.

Example:

Network 4



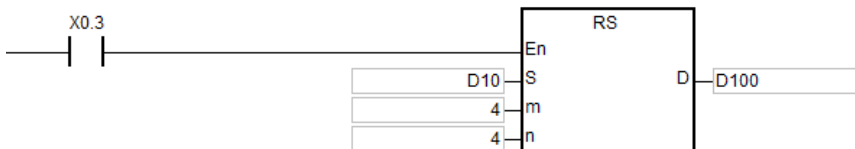
Network 5



Network 6



Network 7



Network 8



Example of the Interruption I32

Network 1



Additional remark:

1. If the values in M and n are out of range, operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#200B will be stored in SR0.
2. Descriptions of the instructions RS / MODRW and their relative flags on the communication ports, COM1 and COM2:

Flags		Functions	Execution
COM1	COM2		
SM96	SM97	Data sending request. When users use the instruction RS/MODRW to send and receive data, the SM96/SM97 should be set to ON via the pulse instruction. After setup, the PLC will start to send and receive data. When the transmission is done, the system will switch the flags SM96/SM97 to	Users set the flag to ON, and the system automatically resets the flag to OFF.

Flags		Functions	Execution
COM1	COM2		
		OFF automatically.	
SM98	SM99	Waiting to receive data. When the status of the SM98/SM99 is ON which means the PLC is waiting to receive data.	The system sets, resets and executes automatically.
SM100	SM101	When the reception is complete, the flag SM100 will be switched to ON automatically. Then users can start managing the data. After the management is done, reset the SM100 to OFF to make it available for further transmissions.	The system sets and users need to reset.
SM102	SM103	An error occurs during the reception of the data in RS/MODRW instructions. Error codes will be recorded on the Error Log.	The system sets, resets and executes automatically.
SM104	SM105	Receiving timeout. If users had set the timeout setting (the specified time in the SR210/SR211, SR213/SR214) and the system experiences the receiving timeout, SM104/SM105 will be ON. Users will need to set the SM104/SM105 to OFF, once the problem is solved.	The system sets and users need to reset.
SM106	SM107	8/16-bit selection. ON is for 8-bit mode and OFF is for 16-bit mode.	Users do the setups and the system will execute accordingly.
SM108	SM109	Receiving complete. ON: When the PLC receives words, the system will check if there's any word containing interrupt characters (SR621/SR622, Low Byte). If yes, the system will run the interruption subroutine. And then the system will stop receiving data. OFF (Default): When the PLC receives words, the system will check if there's any word containing interrupt characters (SR621/SR622, Low Byte). If yes, the system will run the interruption subroutine. After that the system can continue receiving data till the reception is complete (receiving data length is n).	Users do the setups and the system will execute accordingly.
SM209	SM211	Communication protocol changed. According to the settings of the special data register, SR201, SR202, SR209, SR210, SR211, SR212, SR213, SR214, SR215, SR216, SR210, and SR212 can be used for resets. Set the SM209/SM211 to ON, and the communication protocols for COM1/COM2 will be changed according to the settings in the above special data registers. SM209/SM211 will be set to OFF automatically after the protocol is changed.	Users set the flag to ON, and the system automatically resets the flag to OFF.

3. Descriptions of special registers for setting up the instructions RS/MODRW and their relative setups on the communication ports, COM1 and COM2:

D		Descriptions
COM1	COM2	
SR201	SR202	The node ID of the PLC as a Master or a Slave.
SR210	SR213	Specify the time (ms) to set the timeout. When the value is greater than 0, the timeout setting for RS/MODRW instruction is enabled. In the receiving mode, PLC will set the SM104/SM105 to ON, when the first character is not received during the specified time, or if the time between 2 characters is greater than the specified time. This function can remind users to take actions when timeout occurs. Once the management is done, users should reset the SM104/SM105 to OFF. The timeout value for instruction RS can be set to 0 to disable the timeout functionality. However the value for instruction MODRW should be set between 100 and 65535.
SR621	X	Interrupt request. When the PLC receives any word containing interrupt characters (SR621, Low Byte), the system will trigger I32 for interruption. But when n is 0, the request will not be executed.
X	SR622	Interrupt request. When the PLC receives any word containing interrupt characters (SR622, Low Byte), the system will trigger I33 for interruption. But when n is 0, the request will not be executed.

4. SR215 and SR216 are used for recording the codes for the PLC COM interface. Refer to the table below:

Code	0	1	2
Interface	RS-232	RS-485	RS-422

5. SR209 and SR212 : RS-485/RS-232 communication protocol. Refer to the table below for setup:

b0	Data Length			7 (Value = 0)		8 (Value = 1)	
b1 b2	Parity Bit			00	:	None	
				01	:	Odd Parity	
				10	:	Even Parity	
b3	stop bits			1 bit (Value = 0)		2 bits (Value = 1)	
b4 b5 b6 b7	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		RS-232 not supported	

	1000	(16#8)	:	460800	RS-232 not supported
	1001	(16#9)	:	921600	RS-232 not supported
b8~b15	Undefined (Reserved)				

6. PLC Baud Rate and RTU Timeout Timer 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	460800	1
38400	2	921600	1
57600	1		

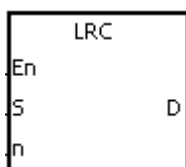
FB/FC	Instruction			Operand			Description		
FC	LRC			S, n, D			Longitudinal parity check		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●				●				
n	●	●			●	●		●	●				●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S : Initial device to which the LRC is applied

n : Number of bytes

D : Initial device in which the operation result is stored

Explanation:

1. Please refer to the additional remark on the instruction LRC for more information about the LRC check code.
2. The operand n should be an even number, and should be within the range between 1 and 1000. If n is not within the range, the 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 hexadecimal data in the device specified by S is divided into the high 8-bit data and the low 8-bit data. The LRC is applied to every byte, and the operation result is stored 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 hexadecimal data in the device specified by S is divided into the high 8-bit data (invalid data) and the low 8-bit data. The LRC is applied to every byte, and the operation result is stored 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 mode: SM210 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.

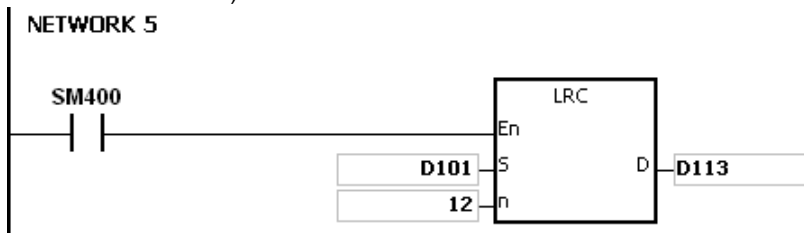
PLC ⇨ VFD-S

The PLC sends “ : 01 03 2101 0006 D4 CR LF”.

The PLC sends the data.

Register	Data		Description
D100 Low 8 bits	' :	16#3A	STX
D101 Low 8 bits	'0'	16#30	ADR 1
D102 Low 8 bits	'1'	16#31	ADR 0
D103 Low 8 bits	'0'	16#30	CMD 1
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
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. It can be calculated by means of the instruction LRC. (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 remark:

1. 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) DATA 0	' 2 '	Data: The $n \times 8$ -bit data is composed of $2n$ ASCII codes.
	' 1 '	
	' 0 '	
	' 2 '	
	' 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 values starting from the communication address to the data are added up. The two's complement of the sum gotten 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 $16\#D7$.

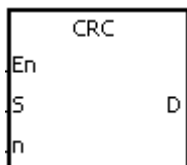
FB/FC	Instruction		Operand				Description					
FC		CRC	S, n, D				Cyclic Redundancy Check					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●				●				
n	●	●			●	●		●	●				●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S** : Initial device to which the CRC is applied
- n** : Number of bytes
- D** : Initial device in which the operation result is stored

Explanation:

- Please refer to the additional remark on the instruction CRC for more information about the CRC check code.
- The operand n should be within the range between 1 and 1000. If n is not within the range, the operation error occurs, the instruction is not executed, SM0 and SM1 are ON, and the error code in SR0 is 16#200B.
- The 16-bit conversion mode: When SM606 is OFF, the hexadecimal data in the device specified by S is divided into the high 8-bit data and the low 8-bit data. The CRC is applied to every byte, and the operation result is stored in the high 8-bit and the low 8-bit in the device specified by D. The number of bytes depends on n.
- The 8-bit conversion mode: When SM606 is ON, the hexadecimal data in the device specified by S is divided into the high 8-bit data (invalid data) and the low 8-bit data. The CRC is applied to every byte, and the operation result is stored in the low 8-bit in the two registers. The number of bytes depends on n.

Example:

- The PLC is connected to the VFD-S series AC motor drive (RTU mode: SM210 is ON; 16-bit mode: SM606 is ON.). The value 16#12, which will 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.

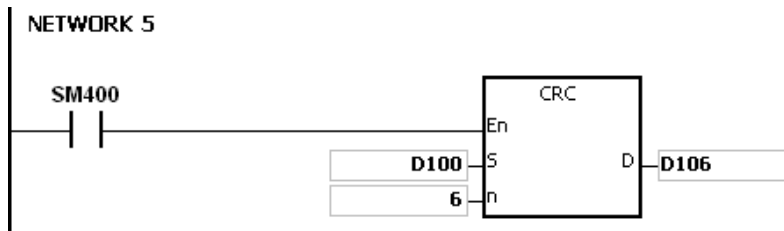
PLC⇒VFD-S

The PLC sends 01 06 2000 0012 02 07.

The PLC sends the data.

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. It can be calculated by means of the instruction CRC.
(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 remark:

1. The format of the communication data in the RTU mode:

START	Time interval
Address	Communication address: 8-bit binary address
Function	Function code: 8-bit binary code
DATA (n-1)	Data: nx8-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 as follows.

Step 1: Suppose the data in the 16-bit register (the register in which 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 which is shifted to the right is 0, the data gotten 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~step 5, and 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

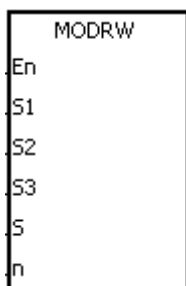
FB/FC	Instruction		Operand				Description				
FC	MODRW		S₁, S₂, S₃, S, n				Reading/Writing the MODBUS data				

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁, S₂, S₃		●					●							
S	●	●					●							
n		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S₁, S₂, S₃	●	●			●	●		●	●				●	○	○		
S	●	●			●	●		●	●				●				
n	●	●			●	●		●	●				●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S₁** : Unit address
- S₂** : Function code
- S₃** : Device address
- S** : Register involved in the reading/writing of the data
- n** : Data length

Explanation:

- The operand **S₁** should be within the range between 0 and 255.
- S₂**: The function code

For example:

- (16#01): The PLC reads the data from several bit devices which are not discrete input devices.
- (16#02): The PLC reads the data from several bit devices which are discrete input devices.
- (16#03): The PLC reads the data from several word devices which are not input registers.
- (16#04): The PLC reads the data from several word devices which are input registers.
- (16#05): The PLC writes the state into a bit device.
- (16#06): The PLC writes the data into a word device.
- (16#0F): The PLC writes the states into several bit devices.
- (16#10): The PLC writes the data into several word devices.

Only the function codes mentioned above are supported, and other function codes can not be executed. Please refer to the examples below.

- S₃**: The device address

If the device address is illegal, the error occurs. The error code is stored in the error log.

- S**: The register involved in the reading/writing of the data

The data which will be written into the external equipment is stored in the register in advance.

The data which is read from the external equipment is stored in the register.

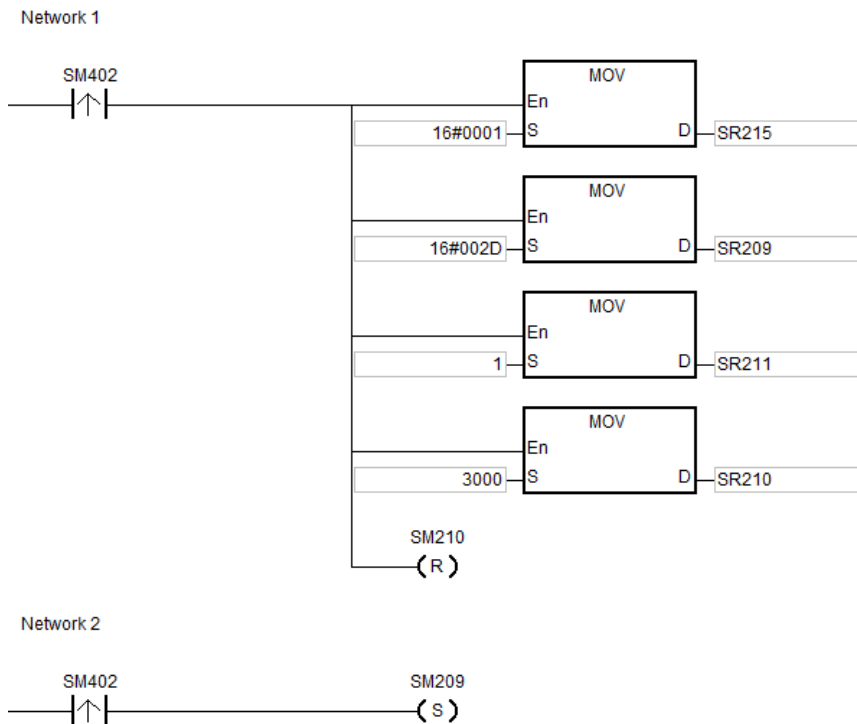
5. **n**: The length of the data

The size of the data can not be larger than 240 bytes. For the communication commands related to the coils (outputs), the unit of the data is the bit, and **n** should be within the range between 1 and 1920. For the communication commands related to the registers, the unit of the data is the word, and **n** should be within the range between 1 and 120.

6. The instruction can be used for several times in the program, but when using different instructions such as MODRW and FWD in the same communication port, only one instruction can be executed at a time.
7. If the communication timeout occurs, SM104 and SM105 will be switched to ON. After the problem is solved, users have to reset SM104 and SM105 to OFF. When setting the instruction MODRW, the setting for the timeout should be in the range of 100-65535ms.
8. When in the MODBUS ASCII mode, users should set up the data for transmission. This instruction will automatically add the initial word “:”, the checking word “LRC” and the ending word “CR LF”. The receiving data will be transformed to ASCII words and stored in the internal register. AH Motion CPU will automatically transform the data to HEX values and store them in **S**.
9. When in the MODBUS RTU mode, users should set up the data for transmission. This instruction will automatically add the checking word “CRC” in and the receiving data will be transformed to HEX values and stored them in **S**.

Examples for Setting up the Communication Format:

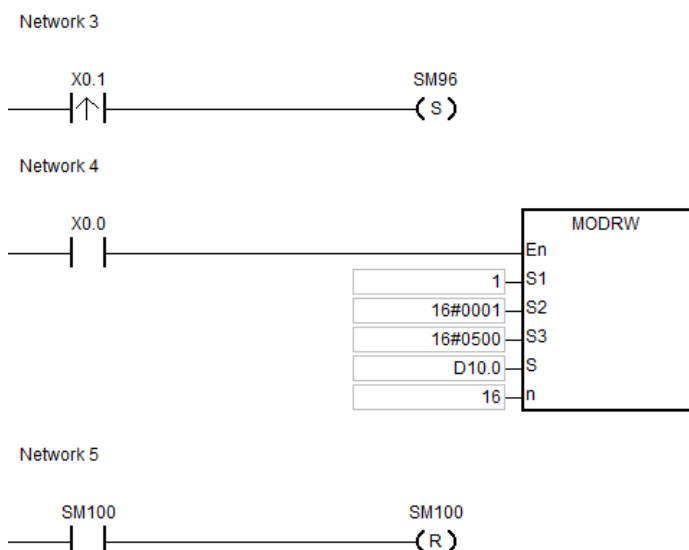
1. Set the communication protocol to RS-485 ASCII, 9600, 8, E, 1. Users can use HWCONFIG or the special registers to set up. (Please refer to the **ISPSOFT User Manual** for HWCONFIG setups and refer to RS instruction for SR, SM register setups.)
2. Use the communication port RS-485 (SR215=1).
3. Set the baud rate and the format to 9600, 8, E, 1 (SR209=16#002D).
4. Set the number of times the command is resent to 1 (SR211=1).
5. Set communication timeout to 3000ms (SR210=3000).
6. Set the mode to ASCII mode. (SM210=OFF).
7. The setup is successful (SM209=ON).



The configurations can be done via ISPSOft → HWCONFIG → COM Port, and these steps can be ignored.

Example 1:

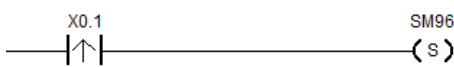
1. Function code 01 (16#01): The PLC reads the data from several bit devices which are not discrete input devices. (Here states 16 pieces of data as an example.)
2. Connect the AH Motion CPU and DVP-ES2 PLC.
When SM96 and X0.0 are both ON, PLC will read data from the DVP-ES2 PLC, starting from Y0 (16#0500) to Y15.
3. The replies from DVP-ES2 PLC will be stored from D10.0 to D10.15.
4. When the data reception of DVP-ES2 PLC is complete, PLC will check the data format. If the format is correct, SM100 will be set to ON. If not, SM102 will be set to ON.



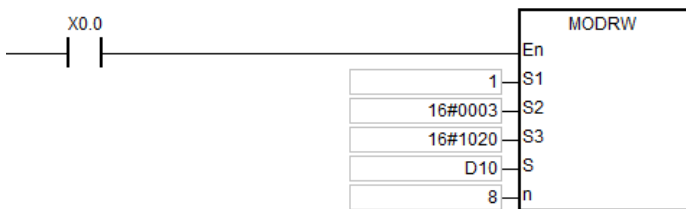
Example 2:

1. Function code 03 (16#03): The PLC reads the data from several bit devices which are not discrete input devices. (Here states 8 pieces of data as an example.)
2. Connect the AH Motion CPU and DVP-ES2 PLC.
When SM96 and X0.0 are both ON, PLC will read data from the DVP-ES2 PLC, starting from D20 (16#1020) to D27.
3. The replies from DVP-ES2 PLC will be stored between D10 and D17.
4. When the data reception of DVP-ES2 PLC is complete, PLC will check the data format. If the format is correct, SM100 will be set to ON. If not, SM102 will be set to ON.

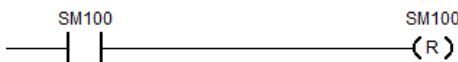
Network 3



Network 4

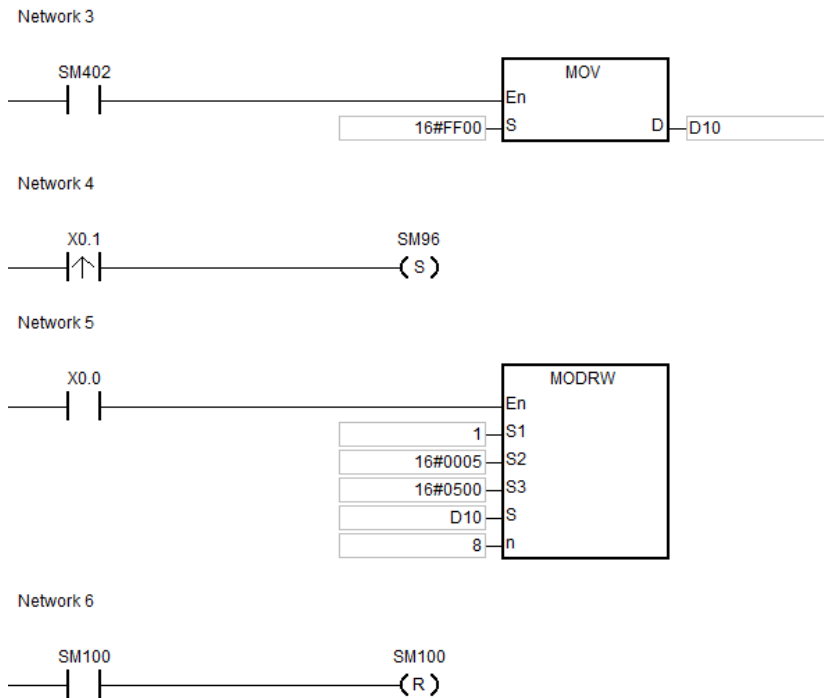


Network 5



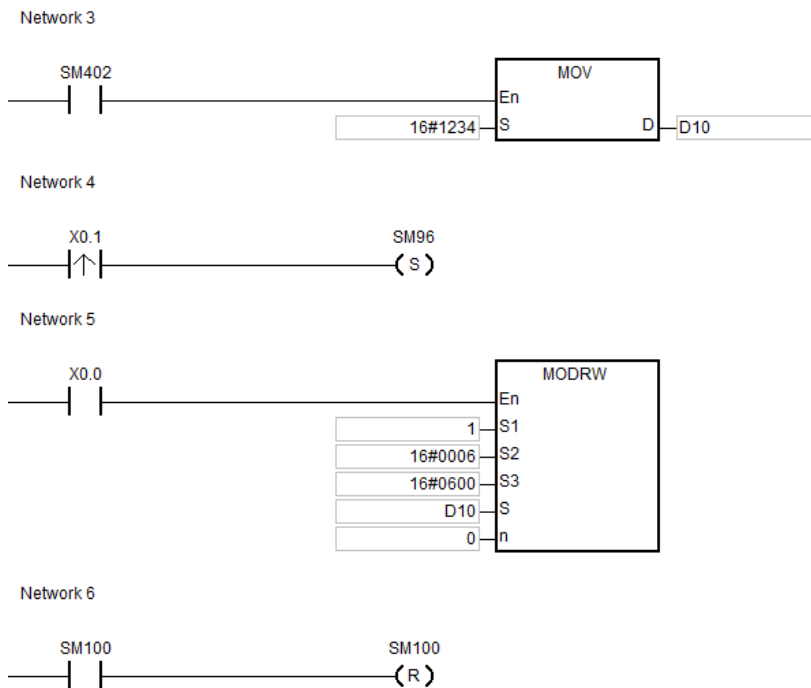
Example 3:

1. Function code 05 (16#05): write bit data to DVP-ES2 PLC.
2. ASCII Mode: connect AH Motion CPU and DVP-ES2 PLC.
When SM96 and X0.0 are both ON, PLC will write data to Y0 (16#0500) on the DVP-ES2 PLC.
3. When the data reception of DVP-ES2 PLC is complete, PLC will check the data format. If the format is correct, SM100 will be set to ON. If not, SM102 will be set to ON.
4. When DVP-ES2 PLC receives this instruction, the Y0 will be set to ON.
5. This function code is to write. The values in n will not be used.



Example 4:

1. Function code 06 (16#06): write single register data to DVP-ES2 PLC.
2. ASCII Mode: connect AH Motion CPU and DVP-ES2 PLC.
When SM96 and X0.0 are both ON, PLC will write data to T0 (16#0600) on the DVP-ES2 PLC.
3. When the data reception of DVP-ES2 PLC is complete, PLC will check the data format. If the format is correct, SM100 will be set to ON. If not, SM102 will be set to ON.
4. When DVP-ES2 PLC receives this instruction, PLC will write D10 data to T0.
5. This function code is to write. The values in n will not be used.



Additional Remark:

1. If the values in S1 and S2 are out of range, operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#2003 will be stored in SR0.
2. If the specified length of n cannot be written or read by S, operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#2003 will be stored in SR0.
3. If the values in n are out of range, operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#200B will be stored in SR0.
4. When S2 is set to the unit Bit to read and write, S should be a Bit device. Or operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#2003 will be stored in SR0.
5. When S2 is set to the unit Word to read and write, S should be a Word device. Or operation errors will occur and then the instruction will not be executed. SM0=ON, and the error code 16#2003 will be stored in SR0.
6. When the instruction is 0x05 and 0x06, n is invalid. The instruction will only write 1 Bit or Word.
7. When executing the instruction MODRW, the sending flags (SM96, SM97) are both OFF and then the instruction will not be executed.
8. When timeout occurs, the communication flags (SM104, SM105) will be set to ON, and the waiting flags (SM98, SM99) will be set to OFF.
9. When receiving error occurs, the error flags (SM102, SM103) will set to ON, and the waiting flags (SM98, SM99) will be set to OFF.
10. When S2 is set to the unit Word to read and write, S should be a Word device. When S2 is set to the unit Bit to read and write, S should be a Bit device.
11. Descriptions of the instructions RS-485/RS-232, MODRW, the special registers and their relative flags on the communication ports, COM1 and COM2:

Flags		Functions
COM1	COM2	
SM96	SM97	Data sending request.
SM98	SM99	Waiting to receive data.
SM100	SM101	Reception complete
SM102	SM103	Reception Error
SM104	SM105	Receiving timeout
SM209	SM211	Communication protocol changed

Please refer to RS instruction for more information on the communication flags.

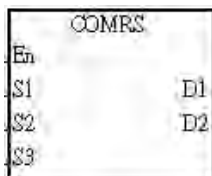
FB/FC	Instruction			Operand				Description						
FC	COMRS			$S_1 \cdot S_2 \cdot S_3 \cdot D_1 \cdot D_2$				Sending and receiving communication data						

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S_1, S_2, S_3		●					●							
D_1, D_2		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S_1	●	●			●	●		●	●				●	○	○		
S_2	●	●			●	●		●	●								
S_3	●	●			●	●		●	●				●	○	○		
D_1	●	●			●	●		●	●								
D_2	●	●			●	●		●	●								

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



- S_1 : Communication port number (1~2)
- S_2 : Source of the data which is sent
- S_3 : Length of the data which is sent
- S : Initial device in which communication data received is stored
- n : Condition of ending the receiving of data

Explanation:

1. The instruction is only applicable to version 1.03 and above.
2. If specific characters are used as the condition of ending the receiving of data, it is suggested that the instruction should be applied to ASCII data or set a timeout as the condition of ending the receiving of data.
3. The instruction COMRS only supports the built-in communication ports of an AH500 series CPU module. (COM1 in AHCPU5x0-EN/AHCPU510-EN/AHCPU520-EN/AHCPU530-EN, and COM1 and COM2 in AHCPU500-RS2/AHCPU510-RS2/AHCPU520-RS2/AHCPU530-RS2)
4. S_1 : Communication port range
 $S_1=1$: COM1
 $S_1=2$: COM2
 If the value in S_1 is neither 1 nor 2, the instruction will not be executed.
5. S_2 : Source of the data which is sent
 S_3 : Length of the data which is sent
 Example: If S_2 is D100 and S_3 is 10, the 10 characters in the 8-bit of D100~D109 will be sent via the communication port specified by S_1 .
6. If the receiving data length of S_3 is 0, no string will be sent. The maximum number of characters which can be sent is 1000.

7. **D₁**: Length of the data which is received.

D₁₊₁~D_{1+n}: Devices in which the data received is stored

If **D₁** is D200, **D₂** is 3, and the ending word of **D₂₊₁** is 16#0D0A, the data received will be stored in the low bytes of 8-bit in the devices starting from D201 (the high bytes will be unchanged), the receiving of data will not stop until the consecutive stop words 16#0D and 16#0A are received. The length of the data received will be written to D200 after 16#0D and 16#0A are received, and a completion flag will be set to ON after the receiving of data stops.

8. **D₂**: Mode of receiving data

D₂₊₁: Condition of ending the **receiving** of data

D₂ and **D₂₊₁** are described below.

D₂	Mode of receiving data	Setting value in D₂₊₁	Remark
0	Not receiving communication data	Unused	After the sending of data is complete, a completion flage will be set to ON.
1	When the data reception stops over the set value in D₂₊₁ , the reception is considered complete.	The value in D₂₊₁ is time. The unit of measurement for time is 1 millisecond. The value in D₂₊₁ should be in the range of 2 to 3000.	If the time is set to greater than 3000 milliseconds, it will be seen as 3000 ms. If the set time is less than 2 ms, it will be seen as 2ms.
2	The data received ends with a specific character.	The setting value in D₂₊₁ is a specific character.	If a specific character is 16#0A, the value in D₂₊₁ will be 16#000A.
3	The data received ends with two consecutive specific characters.	The setting value in D₂₊₁ is two specific characters.	If two specific characters are 16#0D and 16#0A, the value in D₂₊₁ will be 16#0D0A.
4	The data received starts with a specific character. When the data reception stops for over the set time period in D₂₊₁ , the reception is considered complete.	A specific character is stored in the high byte of 8-bit in D₂₊₁ , and time is stored in the low byte of 8-bit in D₂₊₁ . (The time should be set within the range of 2 to 255 milliseconds.)	If a start character is 16#3A, and time is 15 milliseconds, the value in D₂₊₁ will be 16#3A0F.
5	The data received starts with a specific character, and ends with a specific character.	The value in D₂₊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 D₂₊₁ will be 16#3A0A.
6	A specific received data length will be seen as the reception complete.	The value in D₂₊₁ is the length of the data which is received.	For a reception of 10 characters, the value in D₂₊₁ is 10.
Others	If the mode used is not supported, the instruction will not be executed.		

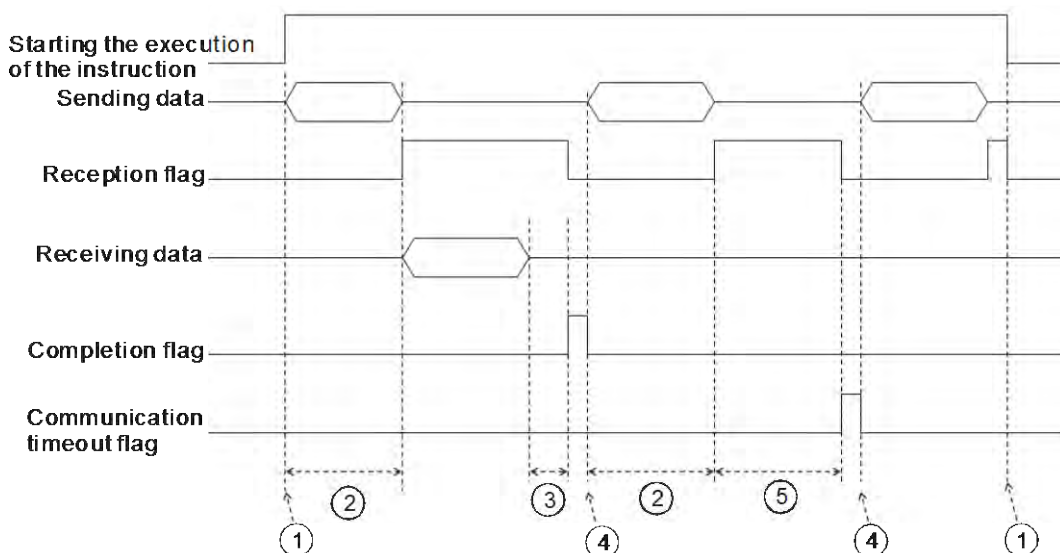
9. If the value in **D₂** is in the range of 1 to 5, the maximum number of characters which can be received is 1000. The execution will stop, when exceeding the limit.

10. Descriptions of the special auxiliary relay SM and the special data register and their relative flags on the communication ports, COM1 and COM2:

Communication port	COM1	COM2	Remark
Reception flag	SM98	SM99	When the PLC receives data, it sets a reception flag to ON. After the receiving of data is complete, the PLC will reset the reception flag to OFF.
Completion flag	SM100	SM101	When the receiving of data is complete, the PLC sets a completion flag to ON. Users have to reset the completion flag to OFF and the PLC will wait for the next communication data.
Communication timeout flag	SM104	SM105	When a timeout occurs, the PLC sets a communication timeout flag to ON. Users have to reset the communication timeout flag to OFF and the PLC will wait for the next communication data.
Timeout period	SR210	SR213	If the value is 0, the communication timeout function will be disabled. The unit of measurement for time is 1 millisecond.
Modes of sending/receiving	SM106	SM107	Modes of the data sent/received ON: 8-bit mode; OFF: 16-bit mode

11. Timing diagrams

- Mode of receiving data: 0
When data is sent, the sending of the data cannot be recalled, even if the condition to start sending is false. But a completion flag will not be set to ON after the sending of the data is complete.
- Mode of receiving data: 1 or 4



Explanation:

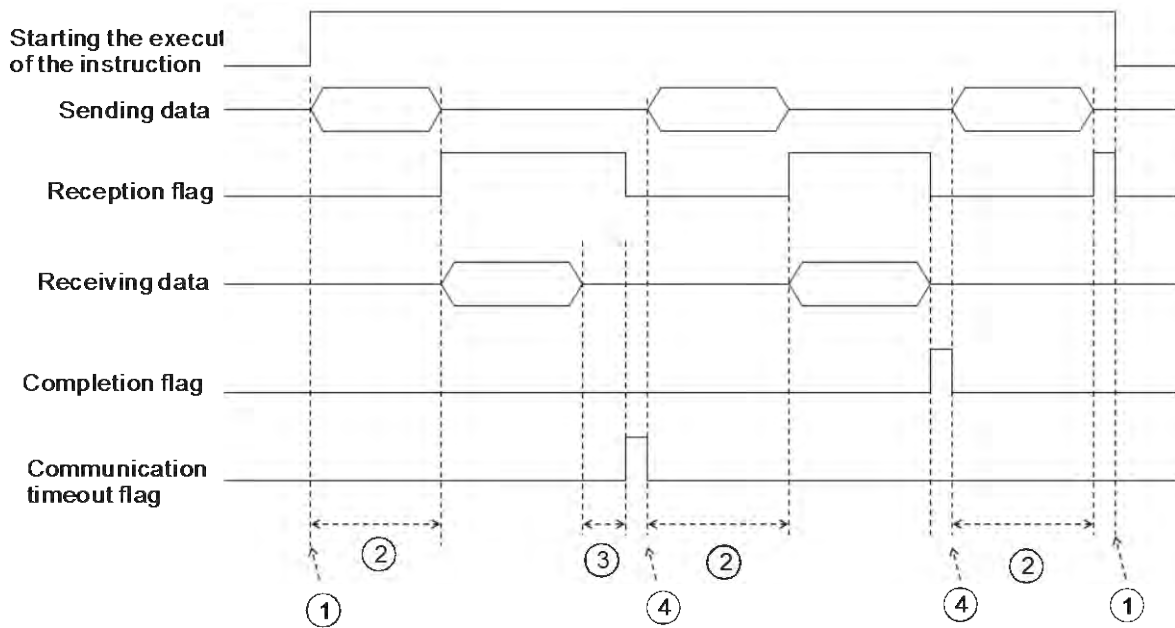
- ① → Users start/stop executing the instruction.
- ② → Timeout will not be measured during the data transmission time.

③→ Once the first character is received, the PLC will start to measure the duration. Whenever a word is received, the number of times will minus 1. When the result exceeds the value in D_2+1 , the completion flag will be set to ON.

④→If the instruction is still enabled after users reset the completion flag or the communication timeout 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 duration. The communication timeout flag will not be set to ON until the time measured exceeds the set timeout. It is suggested that the set timeout should be longer than the time set in D_2+1 .

- Mode of receiving data: 2, 3, 5, or 6



Explanation:

①→Users start/stop executing the instruction.

②→ Timeout will not be measured during the data transmission time.

③→ Once the first character is received, the PLC will start to measure the duration. Whenever a word is received, the number of times will minus 1. When the result exceeds the value in D_2+1 , the completion flag will be set to ON.

④→If the instruction is still enabled after users reset the completion flag or the communication timeout flag, the next communication data is sent automatically when the instruction is scanned in the next cycle.

12. Mode of sending data/Mode of receiving data

8-bit mode: The edited command will be stored in the initial transmission device and the command to be sent includes the head code and the tail codes. The 16-bit data is divided into the high 8-bit data and the low 8-bit data. The high 8-bit data is ignored, and the low 8-bit data can be sent or received. (Take standard MODBUS for example.)

Sending the data: (PLC→External devices)

D10 Low	D11 Low	D12 Low	D13 Low	D14 Low	D15 Low	D16 Low
Head code		Initial transmission device: The low 8-bit data in D10			Tail code 1 Tail code 2	
Length=7						

Receiving the data: (External devices→PLC)

D100 Low	D101 Low	D102 Low	D103 Low	D104 Low	D105 Low	D106 Low
Head code		Initial reception device: The low 8-bit data in D100			Tail code 1 Tail code 2	
Length=7						

16-bit mode: The edited command will be stored in the initial transmission device and the command to be sent includes the head code and the tail codes. The 16-bit data is divided into the high 8-bit data and the low 8-bit data. The high 8-bit data is ignored, and the low 8-bit data can be sent or received. (Take standard MODBUS for example.)

Sending the data: (PLC→External devices)

D10 Low	D10 High	D11 Low	D11 High	D12 Low	D12 High	D13 Low
Head code		Initial transmission device: The low 8-bit data in D10			Tail code 1 Tail code 2	
Length=7						

Receiving the data: (External devices→PLC)

D100 Low	D100 High	D101 Low	D101 High	D102 Low	D102 High	D103 Low
Head code		Initial reception device: The low 8-bit data in D100			Tail code 1 Tail code 2	
Length=7						

The data which the PLC receives from the external devices includes the head and the tail codes. Therefore, the setting of a length should take into account.

Additional remark:

1. There is no limit on the number of times the communication instruction COMRS can be executed. However, every communication port can only be enabled by one communication instruction and the communication instructions which follow will not be executed.
2. Communication instructions such as RS, MODRW, FWD, REV and so on will take the communication ports.
3. The instruction COMRS doesn't come with the checksum functionality. Users can use the provided

convenience instructions to go with the COMRS and to perform the checksum.

4. If the value in **D₂** is 2, 3, 5, or 6, it is suggested to set a timeout. After a timeout is set, the sending of data will keep trying if not receiving any ending word.
5. Whenever the instruction is just executed or the PLC begins to receive new communication data, the value in **D₁~D_{1+n}** will not be cleared automatically. Only after a completion flag is turned from OFF to ON can users know whether the data is received and the received data length. Users can use the instruction ZRST to clear the values in **D₁~D_{1+n}**.
6. If the value in **S₁** is not in the range of 1 to 2, the instruction will not be executed. SM0 is ON and the error code in SR0 is 16#2003.
7. If the number of devices in **S₂** is less than the designated length in **S₃**, the instruction will not be executed. SM0 is ON and the error code in SR0 is 16#2003.
8. If the value in **D₂** is not in the range of 0 to 6, the instruction will not be executed. SM0 is ON and the error code in SR0 is 16#2003.
9. If the value in **D₂** is 6 and the number of devices starting from **D₁** is less than the designated length in **D₂+1**, the instruction will not be executed. SM0 is ON and the error code in SR0 is 16#2003.
10. If the value in **D₂** is in the range of 1 to 5 and the received data quantity is greater than the number of devices starting from **D₁**, the data that is out of range cannot be stored and will be ignored.
11. If the completion flag is ON, the PLC will stop receiving data. Even if there is any data waiting for transmission, the action will not be performed and the data received will not be stored in **D₁~D_{1+n}**.
12. If the designated length in **S₃** is less than 0 or greater than 1000, the instruction will not be executed. SM0 is ON and the error code in SR0 is 16#2003.

3.22 Other Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>WDT</u>	–	✓	Watchdog timer	1
FC	<u>DELAY</u>	–	✓	Delaying the execution of the program	3
FC	<u>GPWM</u>	–	–	General pulse width modulation	7
FC	<u>TIMCHK</u>	–	–	Checking time	7
FC	<u>EPUSH</u>	–	✓	Storing the contents of the index registers	3
FC	<u>EPOP</u>	–	✓	Reading the data into the index registers	3

FB/FC	Instruction		Operand	Description
FC	WDT	P	—	Watchdog timer

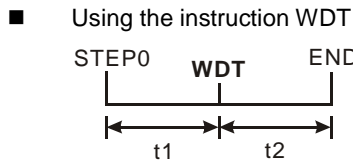
Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



Explanation:

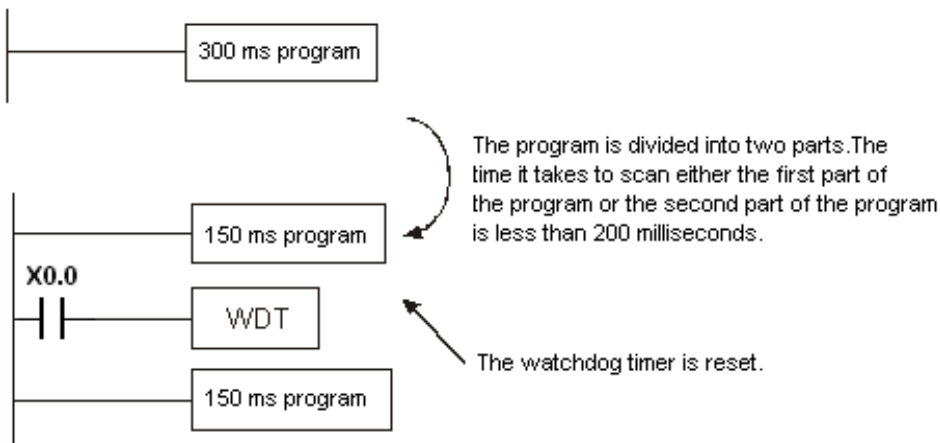
- In the AH Motion CPU, there is a watchdog timer which is used to monitor the operation of the system.
- The instruction WDT is used to reset the watchdog timer in the PLC. If the program scanning time exceeds 200 milliseconds, the error LED indicator is ON, and the PLC stops running.
- The particular point when the watchdog timer acts:
 - The system is abnormal.
 - The execution of the program takes much time, and therefore the scan time is larger than the setting value of the watchdog timer. There are two way users can use to improve the situation.



- Please refer to *ISPSOft User Manual* for more information about changing the setting value of the watchdog timer.

Example:

Suppose the program scanning time is 300 milliseconds. After the program is divided into two parts, and the instruction WDT is inserted between these two parts, the time it takes to scan either the first part of the program or the second part of the program is less than 200 milliseconds.



Additional remark:

Please refer to *ISPSOft User Manual* for more information related to the setting of the watchdog timer.

FB/FC	Instruction		Operand			Description
FC	DELAY	P	S			Delaying the execution of the program

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



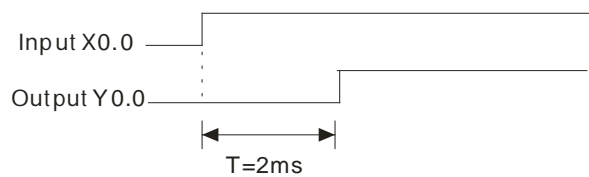
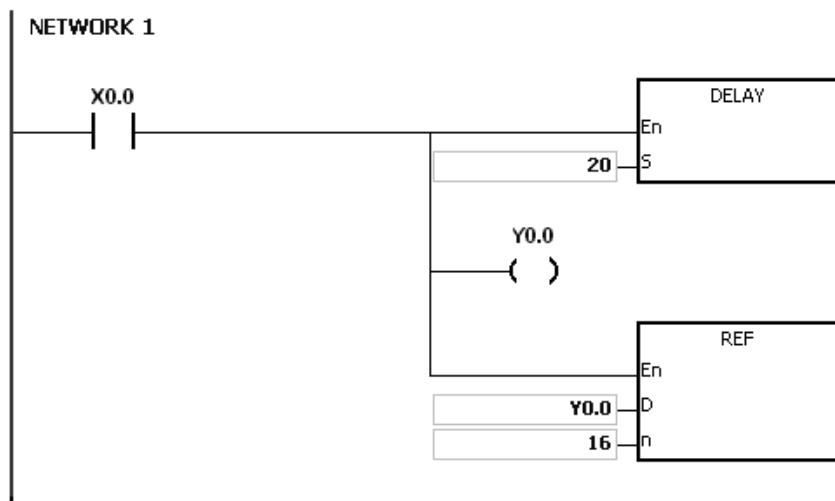
Explanation:

After the instruction DELAY is executed, the execution of the program following the DELAY is delayed for a period of time specified by users.

The unit of **S** is 0.1 milliseconds.

Example:

When X0.0 is ON, the instruction DELAY is executed. The execution of the program following DELAY is delayed for two milliseconds. That is, Y0.0 is ON and the states of Y0.0~Y0.15 are refreshed two milliseconds after the instruction DELAY is executed.



Additional remark:

1. If **S** is less than 0, there is no delay.

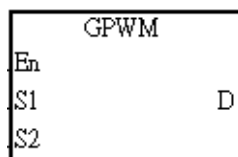
2. If S is larger than 1000, the instruction is not executed, SMO is ON, and the error code in SR0 is 16#2003.
3. Users can adjust the delay according to the practical condition.
4. The delay will increase due to the communication or other influences.

FB/FC	Instruction			Operand			Description		
FC		GPWM		S₁, S₂, D			General pulse width modulation		

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●		●		●				
S ₂	●	●			●	●		●	●				●				
D		●	●	●				●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:

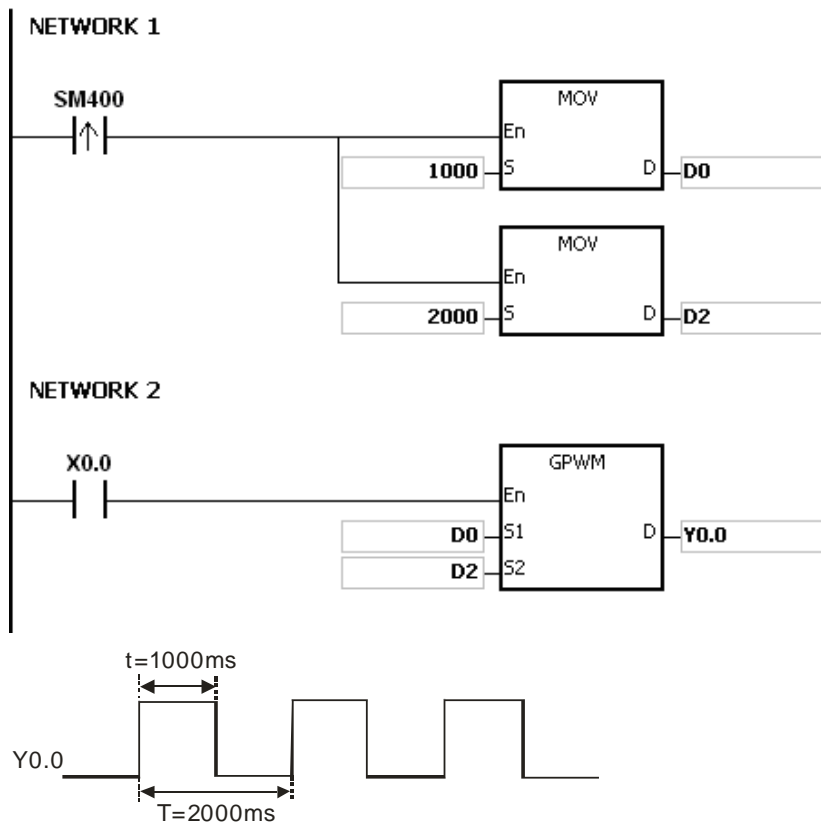
- S₁ : Pulse width
S₂ : Pulse cycle
D : Output device

Explanation:

- When the instruction GPWM is executed, every pulse with a width specified by S₁ and with a cycle specified by S₂ is output from the device specified by D.
- The pulse width specified by S₁ is t. t should be within the range between 0 and 3276 milliseconds.
- The pulse cycle specified by S₂ is T. T should be within the range between 1 and 32767 milliseconds, and S₁ should be less than S₂.
- S₂+1 and S₂+2 are parameters for system use. Please do not occupy them.
- If S₁ is less than 0, there is no pulse output. If S₁ is larger than S₂, the output device keeps ON.
- S₁ and S₂ can be altered during the execution of the instruction GPWM.
- If the conditional contact is not enabled, there is no pulse output.
- When the on-line editing is used, 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 X0.0 is ON, the pulses illustrated below are output from Y0.0. When X0.0 is OFF, Y0.0 is OFF.



3

Additional remark:

1. The instruction counts by the scan cycle. Therefore, the maximum error is one scan cycle. Besides, S1, S2, and (S2-S1) should be larger than the scan cycle. Otherwise, an error occurs when the instruction GPWM is executed.
2. If the instruction is used in the function block or the interrupt task, the inaccurate pulse output will occur.
3. If users declare the operand **S₂** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

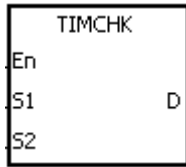
FB/FC	Instruction		Operand				Description									
FC		TIMCHK	S₁, S₂, D				Checking time									

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂		●					●							
D	●													

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●				●				
S ₂	●	●			●	●		●	●		●	○	●	○	○		
D	●	●	●	●				●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



S₁ : Time which passes

S₂ : Setting value

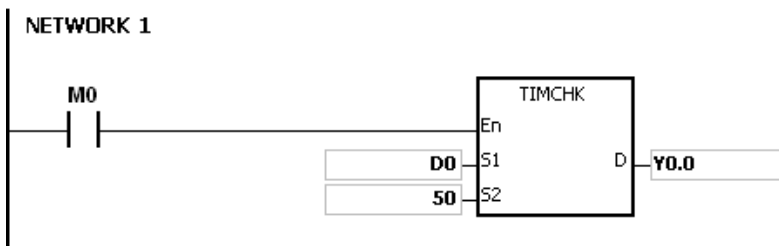
D : Output device

Explanation:

- When the conditional contact is ON, **S₁** starts to count. **D** is not ON until the value of **S₁** is larger than or equal to that of **S₂**. Even if the conditional contact is switched OFF later, the value of **S₁** remains and **D** is still ON.
- If the conditional contact is switched from OFF to ON, **S** is cleared to 0, and **D** is OFF.
- S₁** takes 100 milliseconds as the timing unit.
- S₁ + 1** and **S₁ + 2** are parameters for system use. Please do not occupy them.
- When the on-line editing is used, please reset the conditional contact to initialize the instruction.

Example:

When **M0** is ON, **D0** starts to count. **Y0.0** is not ON until the value in **D0** is larger than or equal to 50 (5 seconds). Even if the conditional contact is switched OFF later, the value in **D0** is unchanged, and **Y0.0** is still ON.



Additional remark:

- If **S** exceeds the device range, the instruction is not executed, **SM0** is ON, and the error code in **SR0** is 16#2003.

-
2. If users declare the operand S1 in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

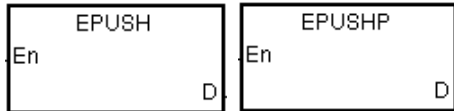
FB/FC	Instruction			Operand				Description						
FC	EPUSH	P		D				Storing the contents of the index registers						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

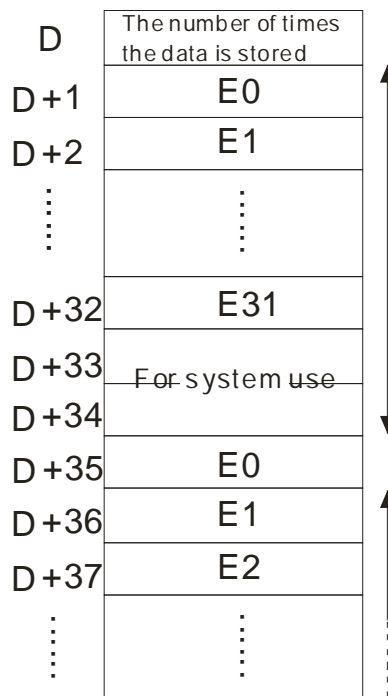
Graphic expression:



D : Device in which the value in the index register is stored

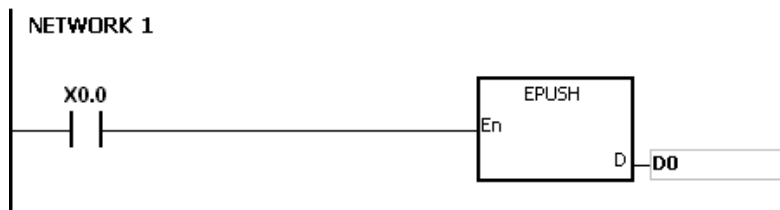
Explanation:

1. The values in E0~E31 are stored in the devices specified by the value in D.
2. The execution of the instruction involves thirty-four devices, and the last two devices are for system use. If the instruction is executed and the number of times the data is stored is n, which is the value in D, the data in E0~E31 is stored in D+34*n+1~D+34*n+32, and the value in D becomes n+1.
3. If the instruction EPUSH is executed several times, the data in E0~E31 is stored several times in the devices specified by the changeable value in D. Therefore, the range of devices should be wide enough.
4. If the instruction is used with the instruction EPOP, the value which is stored last in the device specified by the value in D is read first.



Example:

Suppose the value in D0 is 0. When X0.0 is ON for the first time, the data in E0~E31 is transmitted to D1~D32, and the value in D0 becomes 1. When X0.0 is switched from OFF to ON for the second time, the data in E0~E31 is transmitted to D35~D66, and the value in D0 becomes 2. When X0.0 is switched from OFF to ON for the nth time, the data in E0~E31 is transmitted to D+(the value in D0)*34+1~D+(the value in D0)*34+32.



Additional remark:

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 $D + ((\text{the value in } D) + 1) * 34 - 1$ exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction			Operand				Description						
FC	EPOP	P		D				Reading the data into the index registers						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

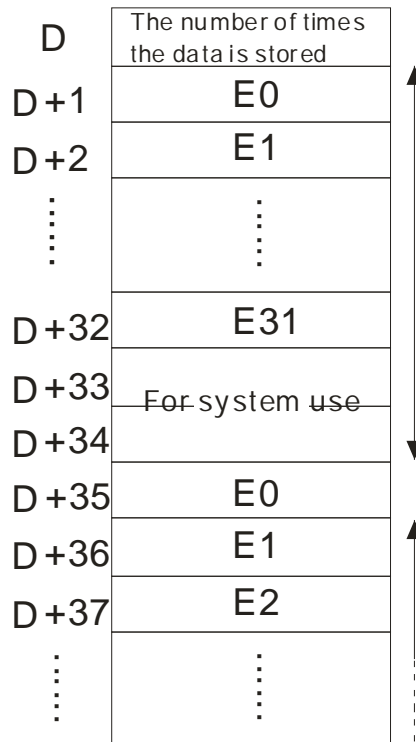
Graphic expression:



D : Device from which the value is read

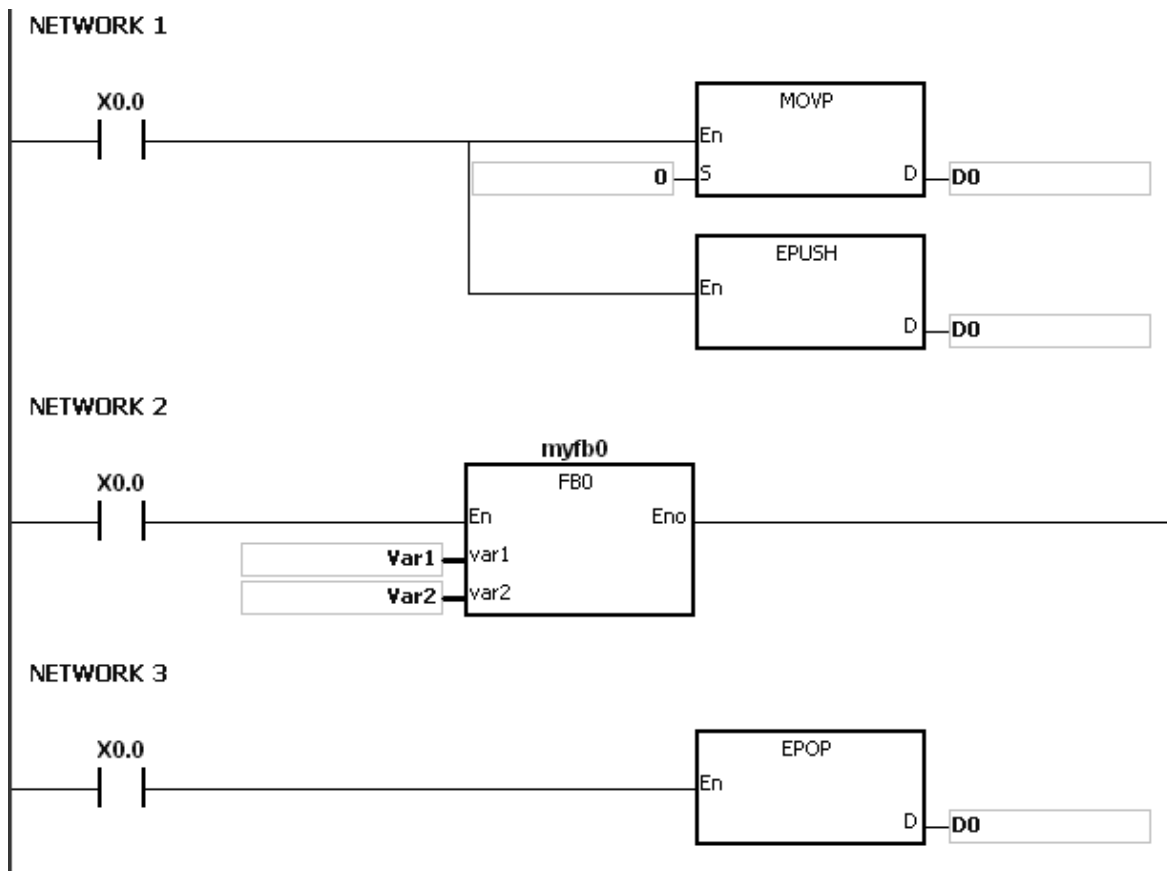
Explanation:

- The values in the devices specified by the value in **D** are read into E0~E31, and the value in **D** decreases by one.
- The execution of the instruction involves thirty-four devices, and the last two devices are for system use. If the instruction is executed and the number of times the data is stored is *n*, which is the value in **D**, the data in $D+34*(n-1)+1 \sim D+34*(n-1)+32$ is read into E0~E31, and the value in **D** becomes *n*-1.
- The value which is stored last in the device specified by the value in **D** is read first.



Example:

When X0.0 is ON, the value in D0 is set to 0, and the values in E0~E31 are transmitted to D1~D32. After the execution of FB0 is complete, the values in D1~D32 are read into D1~D32.



Additional remark:

1. If the value in **D** is less than or equal to 0, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If $D + (\text{the value in } D) * 34 - 1$ exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3.23 String Processing Instructions

FB/F C	Instruction			Pulse instruction	Description	Step
	16-bit	32-bit	64-bit			
FC	<u>BINDA</u>	<u>DBINDA</u>	–	✓	Converting the signed decimal number into the ASCII code	5
FC	<u>BINHA</u>	<u>DBINHA</u>	–	✓	Converting the binary hexadecimal number into the hexadecimal ASCII code	5
FC	<u>BCDDA</u>	<u>DBCDDA</u>	–	✓	Converting the binary-coded decimal number into the ASCII code	5
FC	<u>DABIN</u>	<u>DDABIN</u>	–	✓	Converting the signed decimal ASCII code into the signed decimal binary number	5-11
FC	<u>HABIN</u>	<u>DHABIN</u>	–	✓	Converting the hexadecimal ASCII code into the hexadecimal binary number	5-11
FC	<u>DABCD</u>	<u>DDABCD</u>	–	✓	Converting the ASCII code into the binary-coded decimal number	5-11
FC	<u>\$LEN</u>	–	–	✓	Calculating the length of the string	5-11
FC	<u>\$STR</u>	<u>\$DSTR</u>	–	✓	Converting the binary number into the string	7
FC	<u>\$VAL</u>	<u>\$DVAL</u>	–	✓	Converting the string into the binary number	7-13
FC	<u>\$FSTR</u>	–	–	✓	Converting the floating-point number into the string	7-8
FC	<u>\$FVAL</u>	–	–	✓	Converting the string into the floating-point number	5-11
FC	<u>\$RIGHT</u>	–	–	✓	The retrieve of the characters in the string begins from the right.	7-13
FC	<u>\$LEFT</u>	–	–	✓	The retrieve of the characters in the string begins from the left.	7-13
FC	<u>\$MIDR</u>	–	–	✓	Retrieving a part of the string	7-13
FC	<u>\$MIDW</u>	–	–	✓	Replacing a part of the string	7-13
FC	<u>\$SER</u>	–	–	✓	Searching the string	9-21
FC	<u>\$RPLC</u>	–	–	✓	Replacing the characters in the string	11-17
FC	<u>\$DEL</u>	–	–	✓	Deleting the characters in the string	9
FC	<u>\$CLR</u>	–	–	✓	Clearing the string	3
FC	<u>\$INS</u>	–	–	✓	Inserting the string	9-15
FC	–	<u>FMOD</u>	–	✓	Converting the floating-point number into the binary-coded decimal floating-point number	7-8
FC	<u>FREXP</u>	–	–	✓	Converting the Binary-coded decimal floating-point number into the floating-point number	7

FB/FC	Instruction			Operand				Description						
FC	D*	BINDA	P	S, D				Converting the signed decimal number into the ASCII code						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:

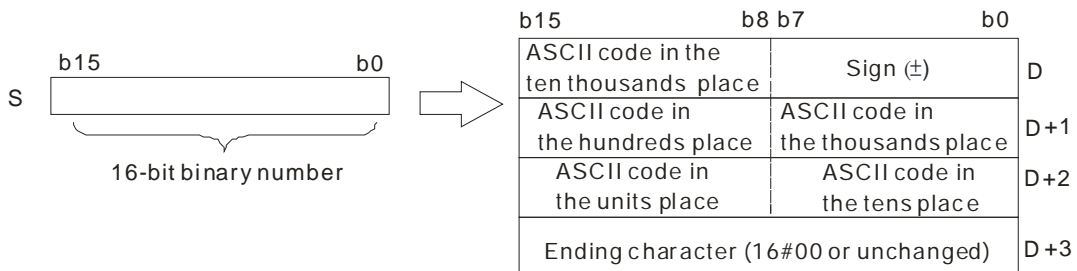


S : Source value

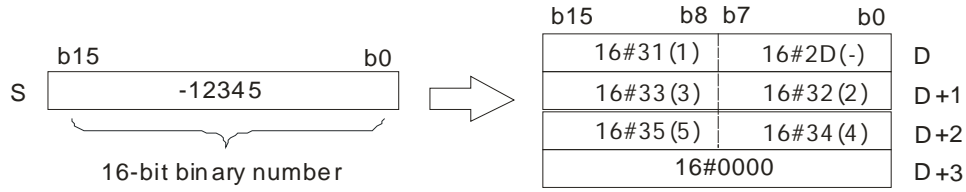
D : Device in which the conversion result is stored

Explanation:

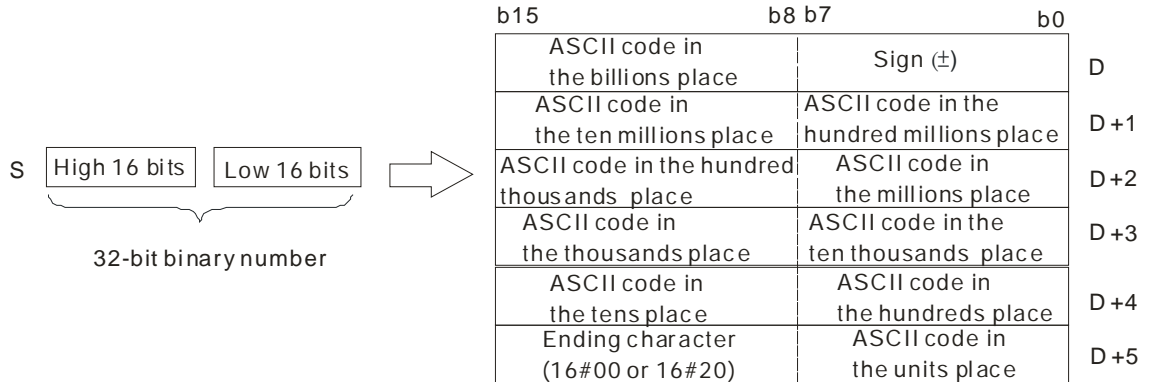
- The signed decimal binary number in **S** is converted into the ASCII code, and the conversion result is stored in **D**.
- The instruction supports SM690, which controls the ending character.
- The value in **S** used in the 16-bit instruction should be within the range between -32768 and 32767, and should be a six-digit binary number. The operand **D** occupies four word devices. The data is converted as follows.



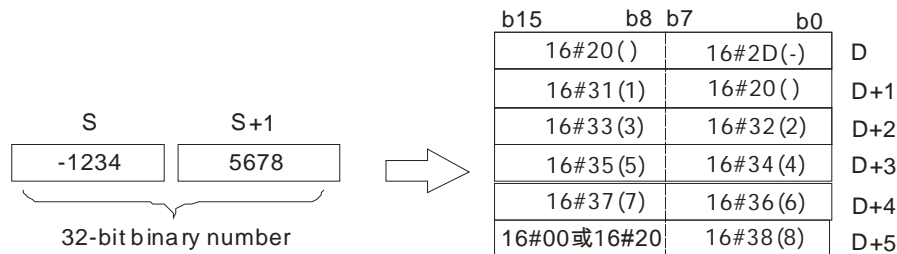
- If SM690 is OFF, 16#0000 is stored in **D+3**. If SM690 is ON, the value in **D+3** is unchanged. Besides, if the value in **S** is a positive value, the sign character in **D** is 16#20. If the value in **S** is a negative value, the sign character in **D** is 16#2D. For example, if the value in **S** is -12345 and SM690 is OFF, the conversion result is as follows.



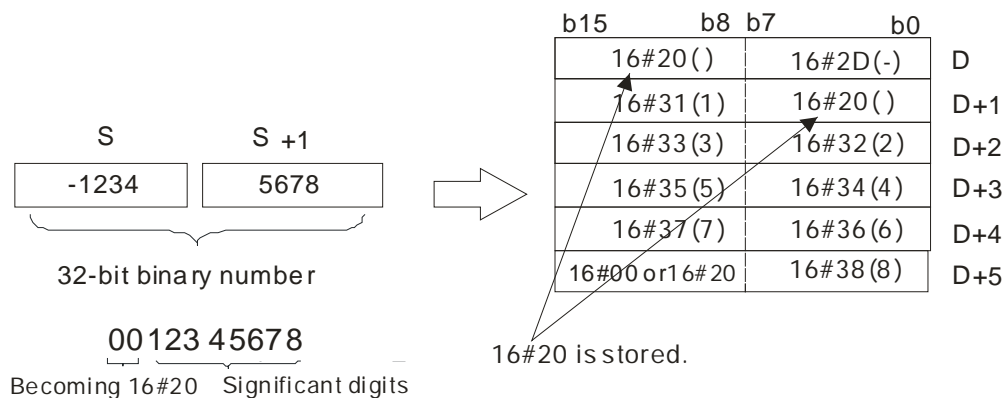
5. The value in **S** used in the 32-bit instruction should be within the range between -2147483648 and 2147483647, and should be an eleven-digit binary number. The operand **D** occupies six word devices. The data is converted as follows.



6. If SM690 is OFF, 16#0000 is stored in the high 8 bits in **D+5**. If SM690 is ON, 16#20 is stored in the high 8 bits in **D+5**. Besides, if the value in **S** is a positive value, the sign character in **D** is 16#20. If the value in **S** is a negative value, the sign character in **D** is 16#2D. For example, if the value in **S** is -12345678, the conversion result is as follows.



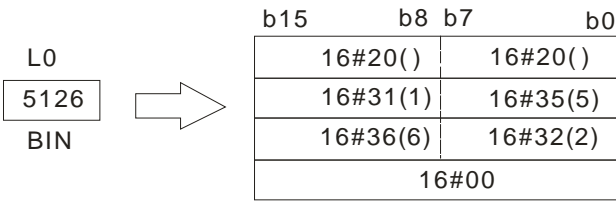
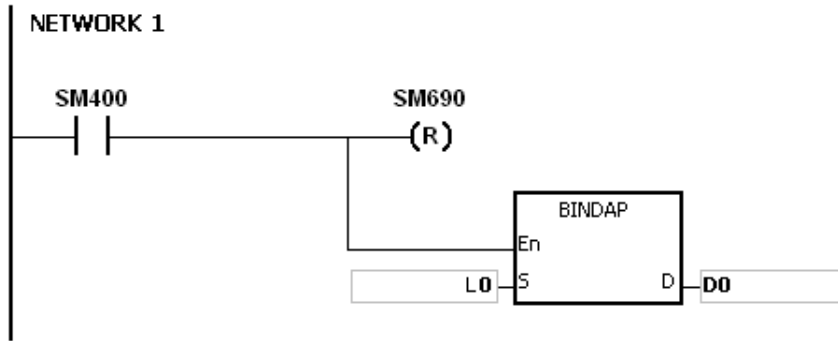
7. Take the 32-bit binary number -12345678 in **S** for example. The digit in the hundred millions place of the number and the digit in the billions place of the number are 0. When the instruction is executed, 16#20 is stored in the low 8 bits in **D+1** and the high 8 bits in **D**.



Example 1:

Suppose the value in L0 is 5126 and SM690 is OFF. When the PLC runs, the values in D0, D1, D2, and D3 are

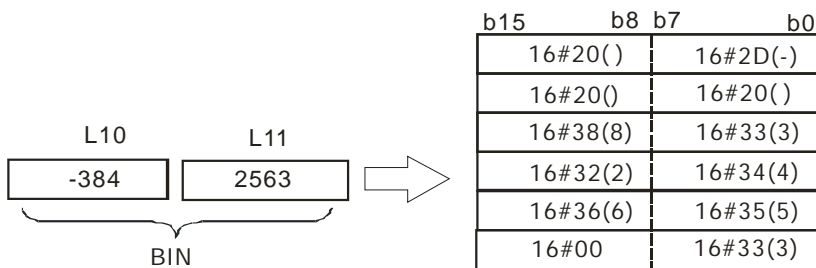
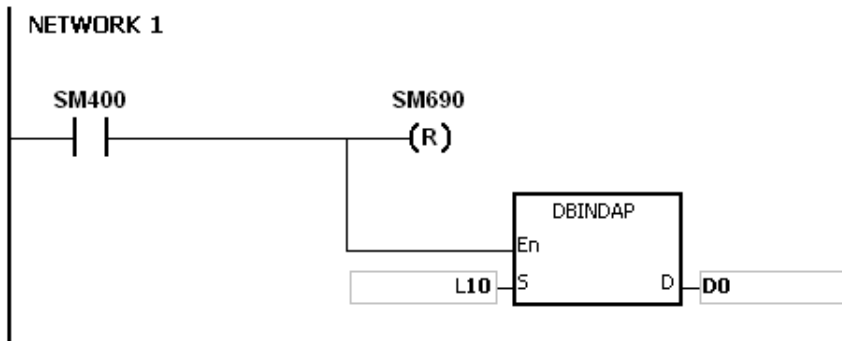
16#2020, 16#3135, 16#3135, and 16#0000 respectively.



3

Example 2:

Suppose the value in L10 is -3842563 and SM690 is OFF. When the PLC runs, the values in D0, D1, D2, D3, D4, and D5 are 16#202D, 16#2020, 16#3833, 16#3234, 16#3635, and 16#0033 respectively.



Additional remark:

1. If D+3 used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
2. If D+5 used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the operand D used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [4] of WORD/INT.
4. If the operand D used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [6] of WORD/INT.

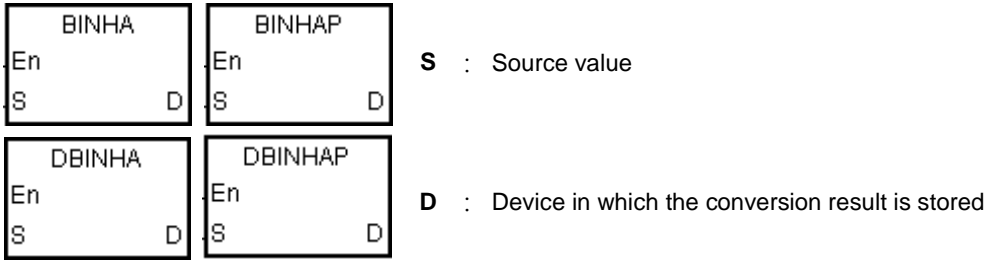
FB/FC	Instruction			Operand	Description
FC	D*	BINHA	P	S, D	Converting the binary hexadecimal number into the hexadecimal ASCII code

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

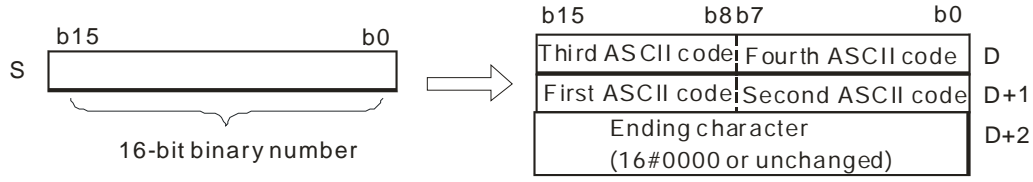
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

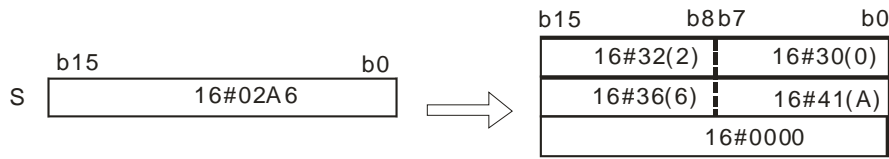


Explanation:

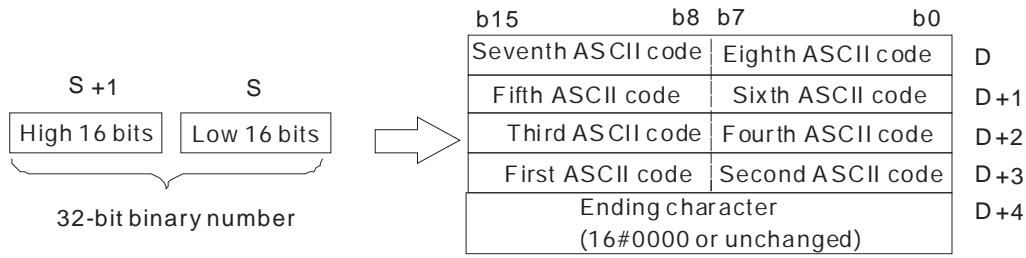
1. The hexadecimal binary number in **S** is converted to ASCII code, and the conversion result is stored in **D**.
2. The instruction supports SM690, which controls the ending character.
3. The value in **S** used in the 16-bit instruction should be within the range between 16#0000 and 16#FFFF, and should be a four-digit binary number. The operand **D** occupies three word devices. The data is converted as follows.



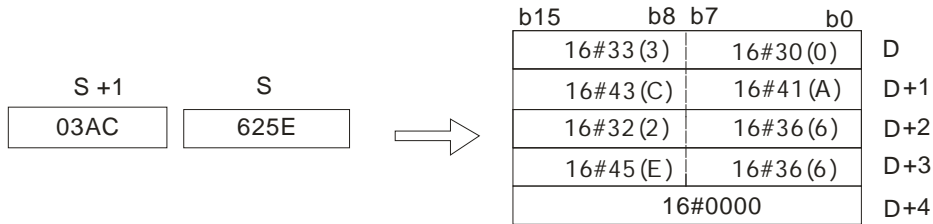
4. If SM690 is OFF, 16#0000 is stored in **D+2**. If SM690 is ON, the value in **D+2** is unchanged. For example, if the value in **S** is 16#02A6 and SM690 is OFF, the conversion result is as follows.



5. The value in **S** used in the 32-bit instruction should be within the range between 16#00000000 and 16#FFFFFFFF, and should be an eight-digit binary number. The operand **D** occupies five word devices. The data is converted as follows.

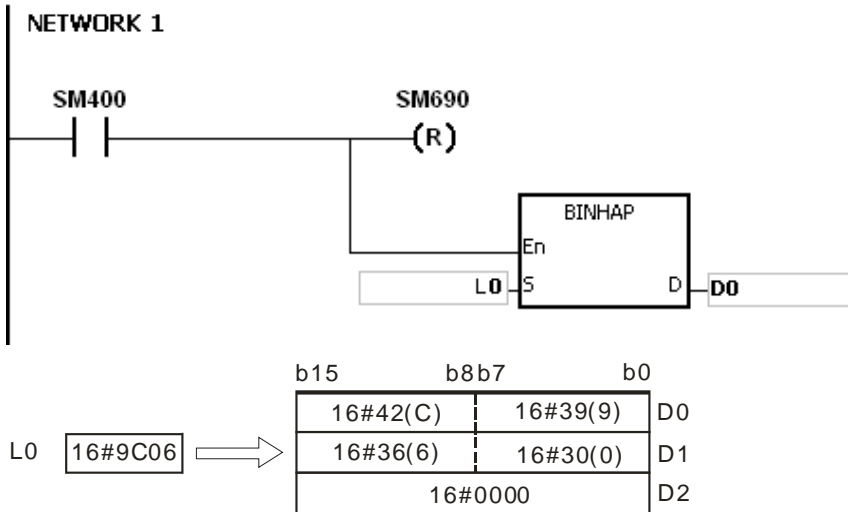


6. If SM690 is OFF, 16#0000 is stored in D+4. If SM690 is ON, the value in D+4 is unchanged. For example, if the value in S is 16#03AC625E and SM690 is OFF, the conversion result is as follows.



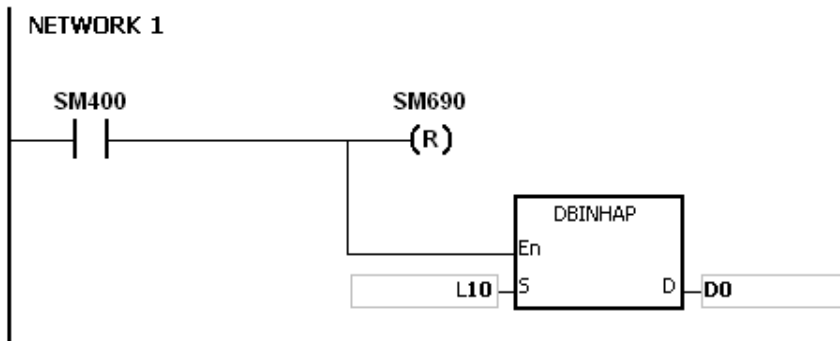
3 Example 1:

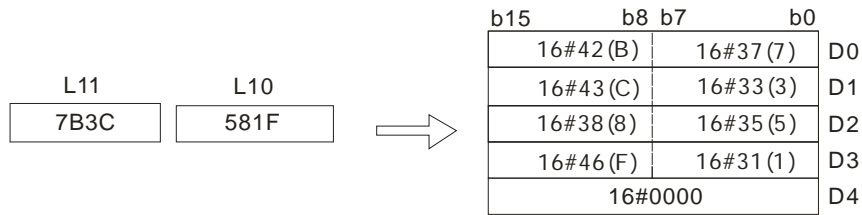
Suppose the value in L0 is 16#9C06 and SM690 is OFF. When PLC runs, the values in D0, D1, and D2, are 16#2020, 16#3135, 16#3135, and 16#0000 respectively.



Example 2:

Suppose the value in L10 is 16#7B3C581F and SM690 is OFF. When the PLC runs, the values in D0, D1, D2, D3, and D4 are 16#4237, 16#4333, 16#3835, 16#4631, and 16#0000 respectively.



**Additional remark:**

1. If **D+2** used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
2. If **D+4** used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
4. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of WORD/INT.

FB/FC	Instruction			Operand			Description									
FC	D*	BCDDA	P	S, D			Converting the binary-coded decimal number into the ASCII code									

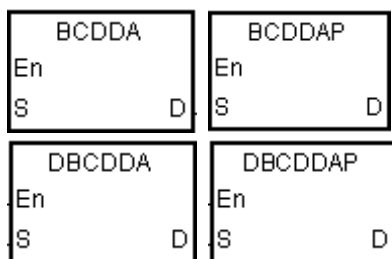
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D		●/●*					●/●*							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:

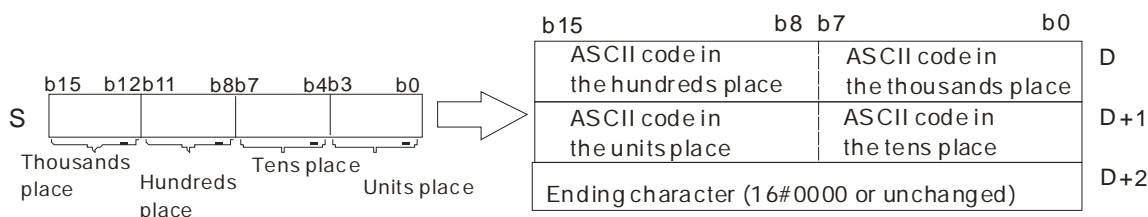


S : Source value

D : Device in which the conversion result is stored

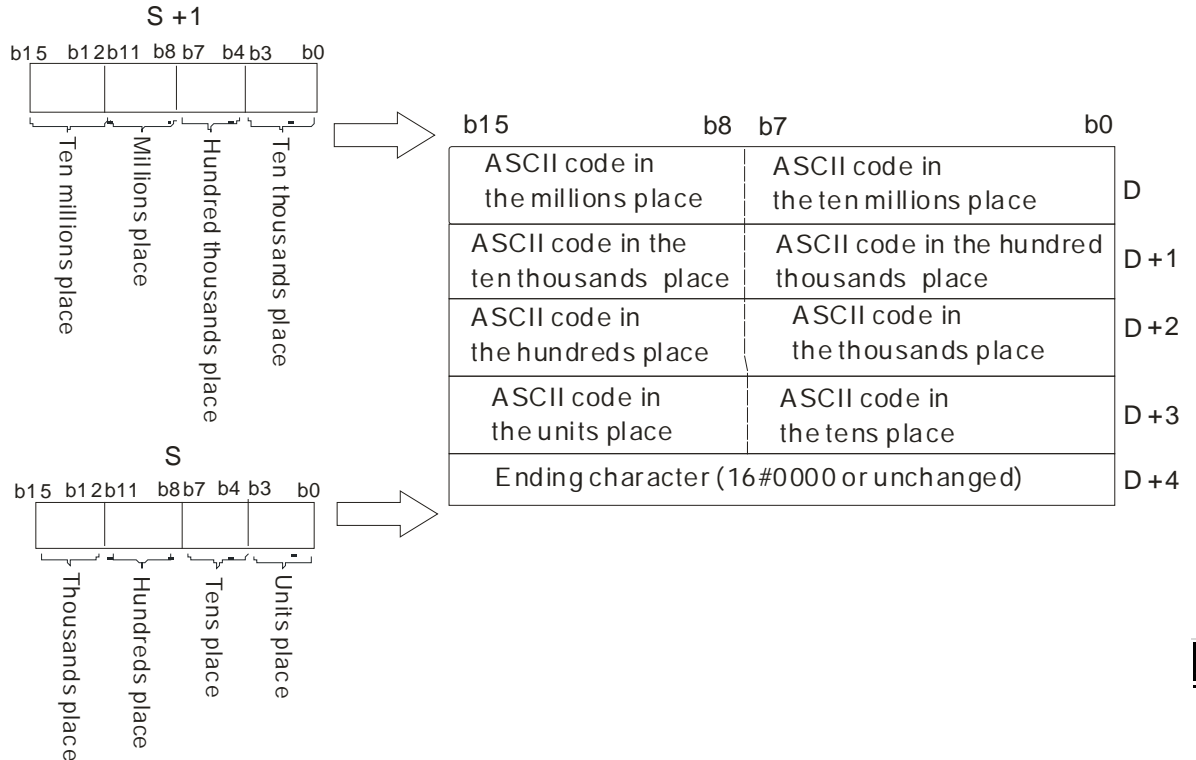
Explanation:

- The binary-coded decimal number in S is converted into the ASCII code, and the conversion result is stored in D.
- The instruction supports SM690, which controls the ending character.
- The binary-coded decimal value in S used in the 16-bit instruction should be within the range between 0 and 9999, and should be a four-digit binary-coded decimal value. The operand D occupies three word devices. The data is converted as follows.



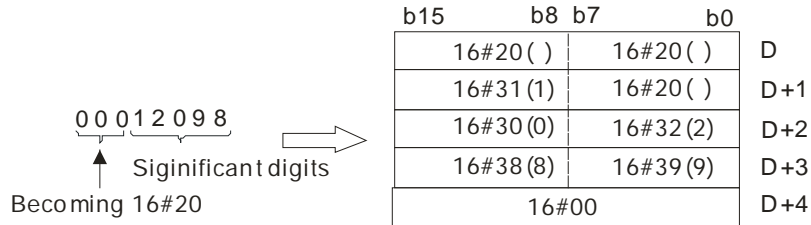
If SM690 is OFF, 16#0000 is stored in D+2. If SM690 is ON, the value in D+2 is unchanged.

- The binary-coded decimal value in S used in the 32-bit instruction should be within the range between 0 and 99999999, and should be an eight-digit binary-coded decimal value. The operand D occupies five word devices. The data is converted as follows.



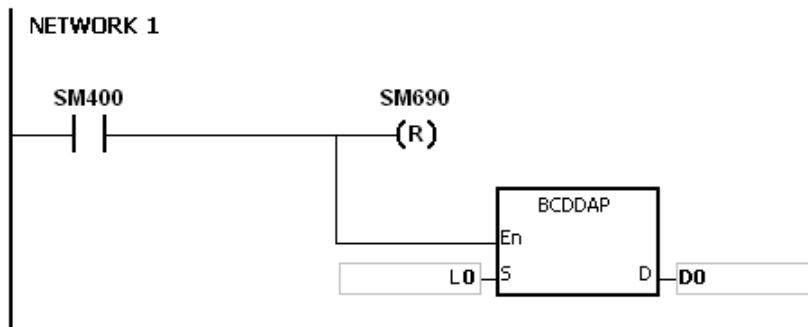
If SM690 is OFF, 16#0000 is stored in **D+5**. If SM690 is ON, the value in **D+5** is unchanged.

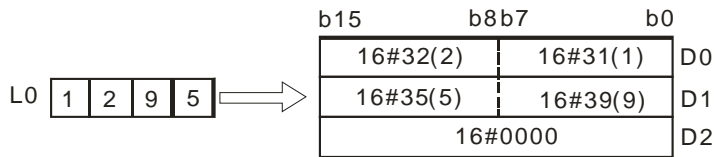
- Take the binary-coded decimal number 12098 in **S** for example. The digit in the hundred thousands place of the number, the digit in the millions place of the number, and the digit in the ten millions place of the number are 0. When the instruction is executed, 16#20 is stored in the low 8 bits in **D+1**, the high 8 bits in **D**, and the low 8 bits in **D**.



Example 1:

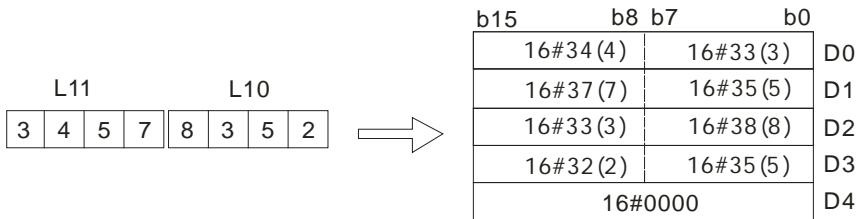
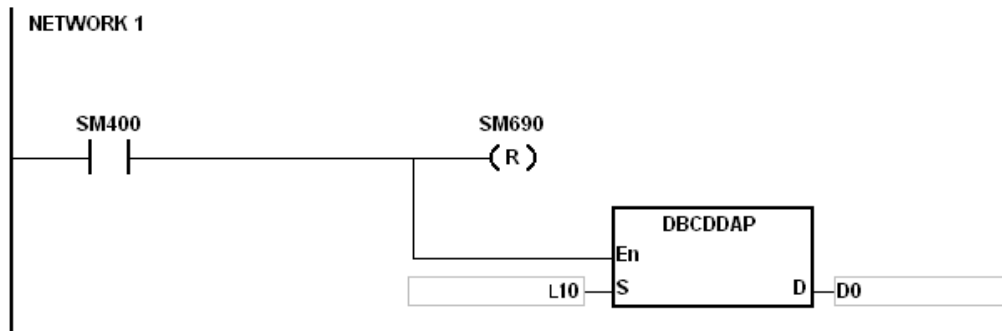
Suppose the binary-coded decimal value in **L0** is 1295 and SM690 is OFF. When PLC runs, the values in **D0**, **D1**, and **D2** are 16#3231, 16#3539, 16#3135, and 16#0000 respectively.





Example 2:

Suppose the binary-coded decimal value in L10 is 34578352 and SM690 is OFF. When the PLC runs, the values in D0, D1, D2, D3, and D4 are 16#3433, 16#3735, 16#3338, 16#3235, and 16#0000 respectively.



Additional remark:

1. If the value in **S** used in the 16-bit instruction is not within the range between 0 and 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. If the value in **S** used in the 32-bit instruction is not within the range between 0 and 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. If **D+2** used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
4. If **D+4** used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the operand **D** used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
6. If the operand **D** used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of WORD/INT.

FB/FC	Instruction			Operand	Description
FC	D*	DABIN	P	S, D	Converting the signed decimal ASCII code into the signed decimal binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

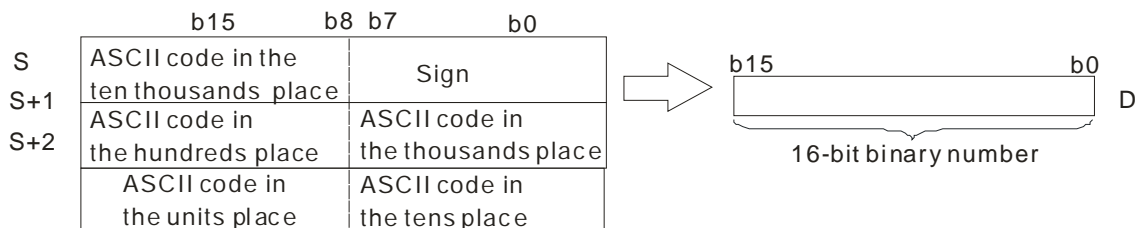


S : Source value

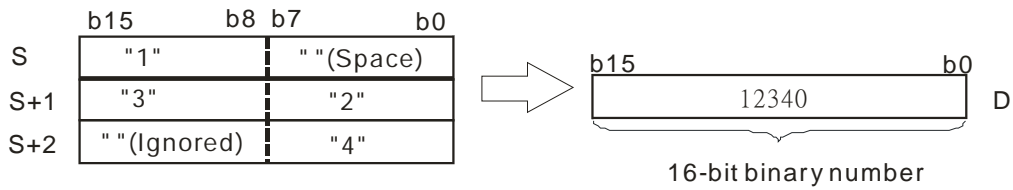
D : Device in which the conversion result is stored

Explanation:

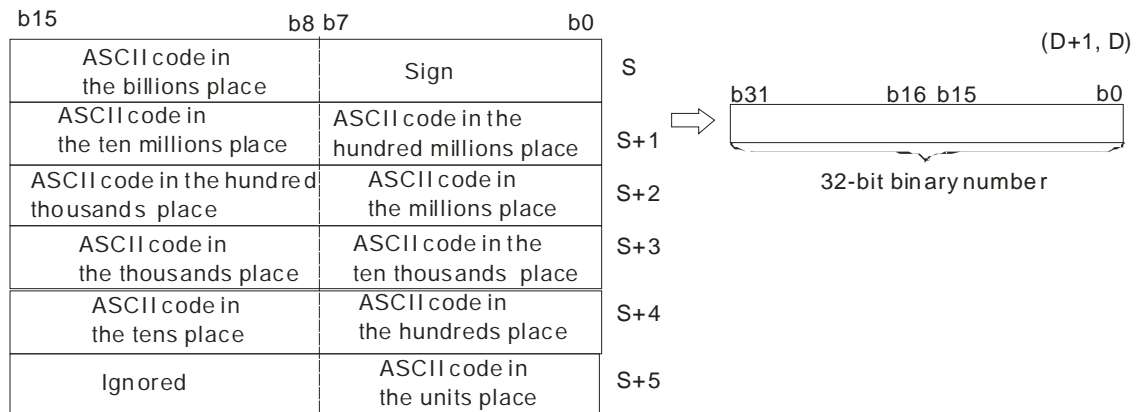
- The signed decimal ASCII code in **S** is converted into the signed decimal binary number, and the conversion result is stored in **D**.
- The operand **S** used in the 16-bit instruction occupies three word devices, and the decimal ASCII code in **S** should be within the range between -32768 and 32767. If **S** is a string, the string should be within the range between “-32768” and “32767”.



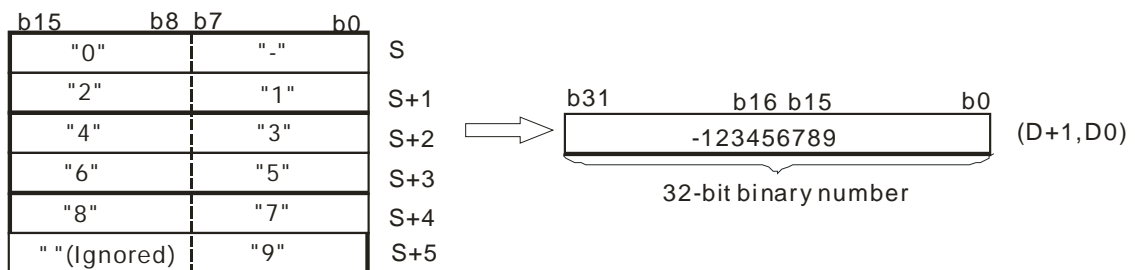
- If **S** used in the 16-bit instruction is a string and the number of characters contained in the string is less than 6, the characters which the string lacks are regarded as 0. The first character is a sign character. If the first character is “ ” (a space), the sign is a positive sign. If the first character is “-”, the sign is a negative sign. Take the string “1234” for example.



4. The operand **S** used in the 32-bit instruction occupies six word devices, and the decimal ASCII code in **S** should be within the range between -2147483648 and 2147483647. If **S** is a string, the string should be within the range between "-2147483648" and "2147483647".



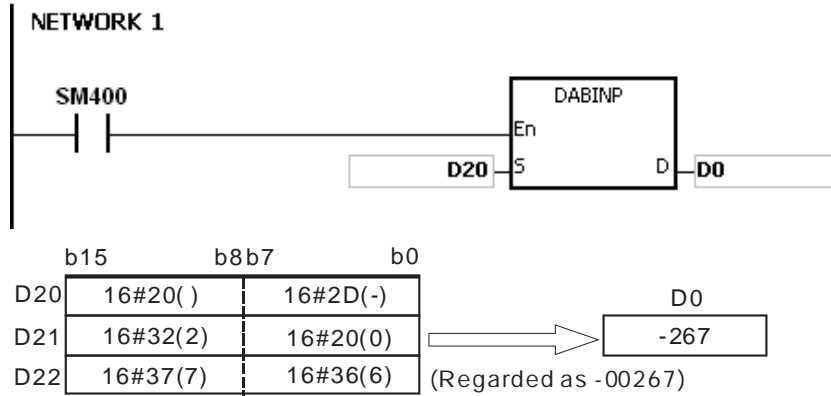
5. If **S** used in the 32-bit instruction is a string and the number of characters contained in the string is less than 11, the characters which the string lacks are regarded as 0. The first character is a sign character. If the first character is " " (a space), the sign is a positive sign. If the first character is "-", the sign is a negative sign. Take the string "-0123456789" for example.



6. If the value in **S** is 16#20 or 16#00, the value is processed as 16#30.
7. If the sign character is 16#20, 16#30 or 16#2B, the conversion result is a positive value. If the sign character is 16#2D, the conversion result is a negative value.
8. If **S** used in the 16-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 6. If **S** used in the 32-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 11.

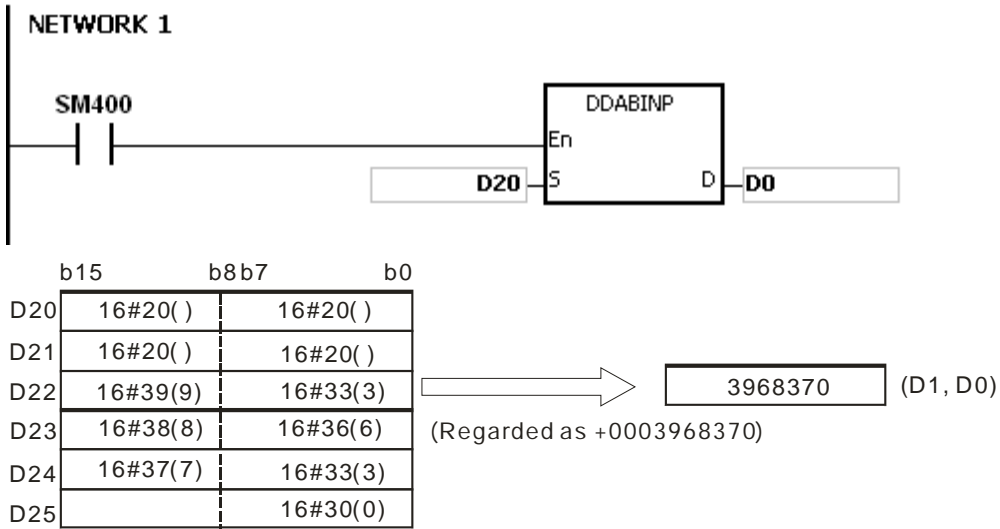
Example 1:

Suppose the values in D20, D21, and D22 are 16#202D, 16#3220, and 16#3736. When the PLC runs, the value in D0 is -267.



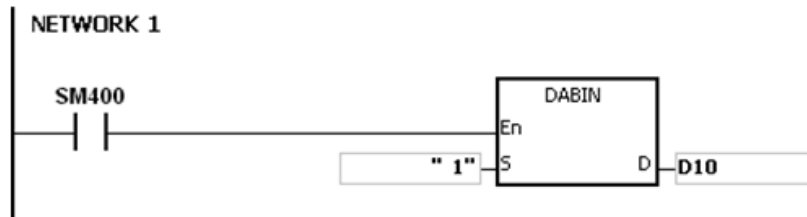
Example 2:

Suppose the values in D20, D21, D22, D23, D24 and D25 are 16#2020, 16#2020, and 16#3933, 16#3836, 16#3733, and 16#3330. When the PLC runs, the value in D0 is 3968370.



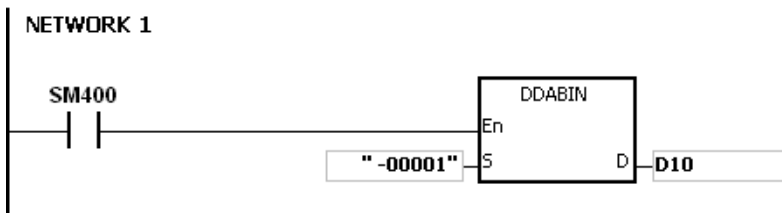
Example 3:

Suppose **S** is the string "1". The first character is ". Since the number of characters contained in the string is less than 6, the string is regarded as "10000". When the PLC runs, the value in D10 is 10000.



Example 4:

Suppose **S** is the string "-00001". The first character is ". Since the number of characters contained in the string is less than 11, the string is regarded as "-0000100000". When the PLC runs, the value in (D11, D10) is -100000.



Additional remark:

1. If the sign character in S is neither 16#20 nor 16#30, 16#2B, 16#2D, the operation error occurs, the instruction is executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the ASCII code in S is not 16#20, 16#0, or within the range between 16#30 and 16#39, 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 device range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If S+2 used in the 16-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
5. If S+5 used in the 32-bit instruction exceeds the device range, SM0 is ON, and the error code in SR0 is 16#2003.
6. If the operand S used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [3] of WORD/INT.
7. If the operand S used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [6] of WORD/INT.

FB/FC	Instruction			Operand	Description
FC	D*	HABIN	P	S, D	Converting the hexadecimal ASCII code into the hexadecimal binary number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

Graphic expression:

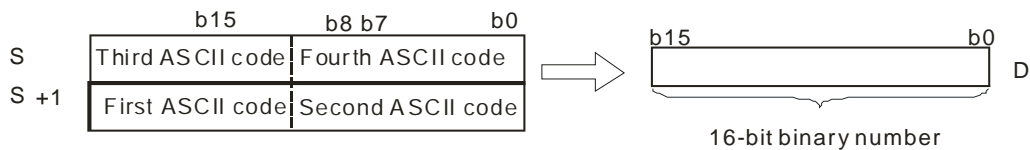


S : Source value

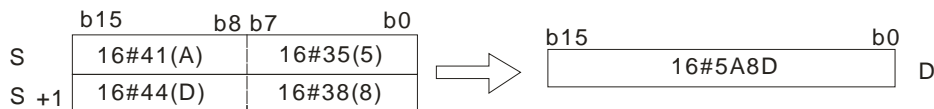
D : Device in which the conversion result is stored

Explanation:

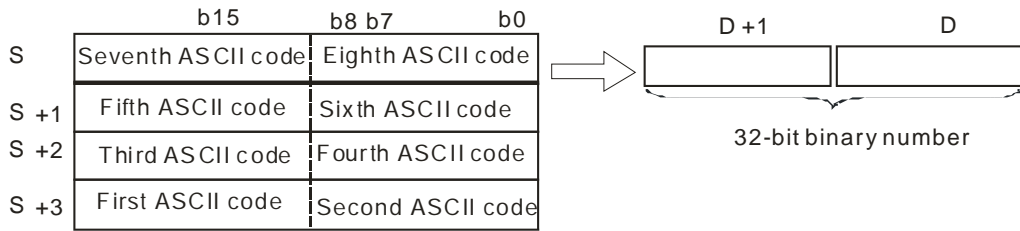
1. The hexadecimal ASCII code in **S** is converted into the hexadecimal binary number, and the conversion result is stored in **D**.
2. The operand **S** used in the 16-bit instruction occupies two word devices, and the hexadecimal ASCII code in **S** should be within the range between 0000 and FFFF. If **S** is a string, the string should be within the range between “0” and “FFFF”.



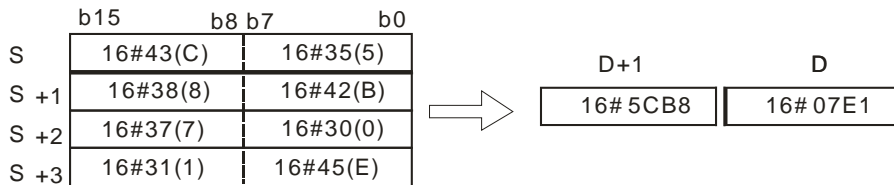
If the ASCII code in **S~S+1** is 5A8D, the conversion result is as follows.



3. The operand **S** used in the 32-bit instruction occupies four word devices, and the hexadecimal ASCII code in **S** should be within the range between 00000000 and FFFFFFFF. If **S** is a string, the string should be within the range between “0” and “FFFFFFF”.



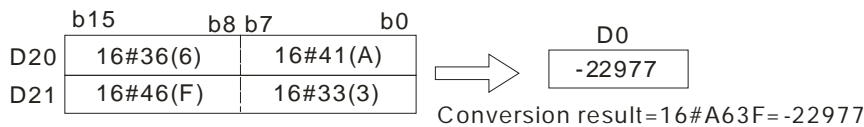
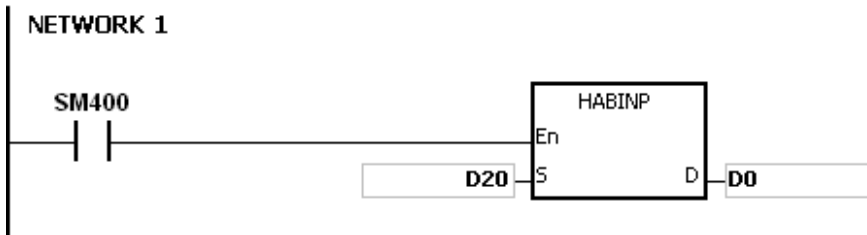
If the ASCII code in **S-S+3** is 5CB807E1, the conversion result is as follows.



- If **S** used in the 16-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 4. If **S** used in the 32-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 8.

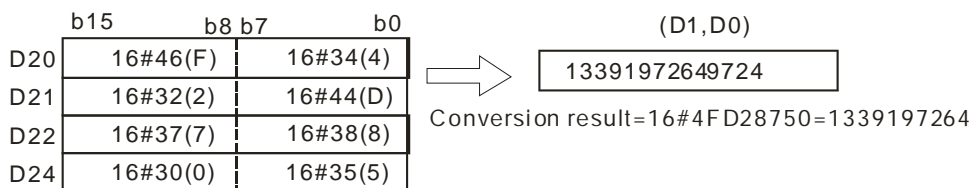
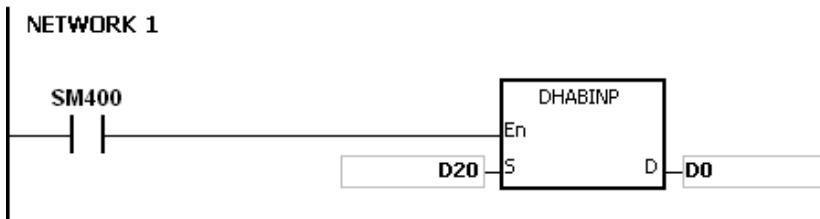
Example 1:

Suppose the values in D20 and D21 are 16#3641 and 16#4633 respectively. When the PLC runs, the value in D0 is -22977.



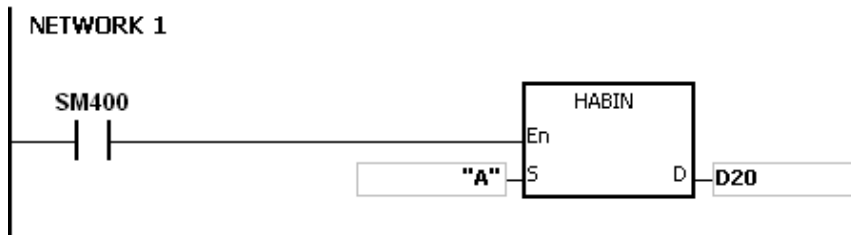
Example 2:

Suppose the values in D20, D21, D22, and D23 are 16#4634, 16#3244, 16#3738, and 16#3035 respectively. When the PLC runs, the value in (D1, D0) is 1339197264.

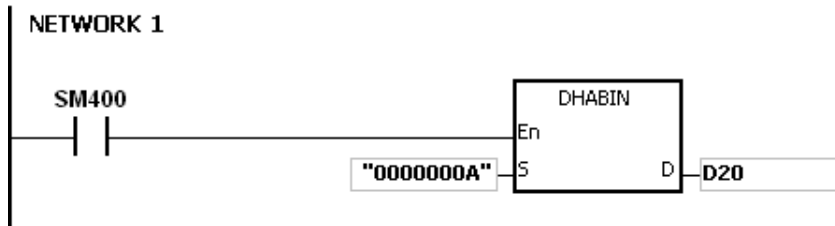


Example 3:

Suppose **S** is the string "A". Since the number of characters contained in the string is less than 4, the string is regarded as "A000". When the PLC runs, the value in D20 is -24576.

**Example 4:**

Suppose **S** is the string "0000000A". When the PLC runs, the value in (D21, D20) is 10.

**Additional remark:**

1. If the ASCII code in **S** is not within the range between 16#30 and 16#39, or within the range between 16#41 and 16#46, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the operand **S** used during the execution of the 16-bit instruction is declared in ISPSoft, the data type will be ARRAY [2] of WORD/INT.
3. If the operand **S** used during the execution of the 32-bit instruction is declared in ISPSoft, the data type will be ARRAY [4] of WORD/INT.

FB/FC	Instruction			Operand				Description						
FC	D*	DABCD	P	S, D				Converting the ASCII code into the binary-coded decimal number						

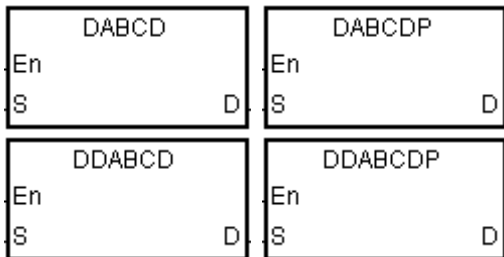
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S		●	●*				●	●*						
D		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:

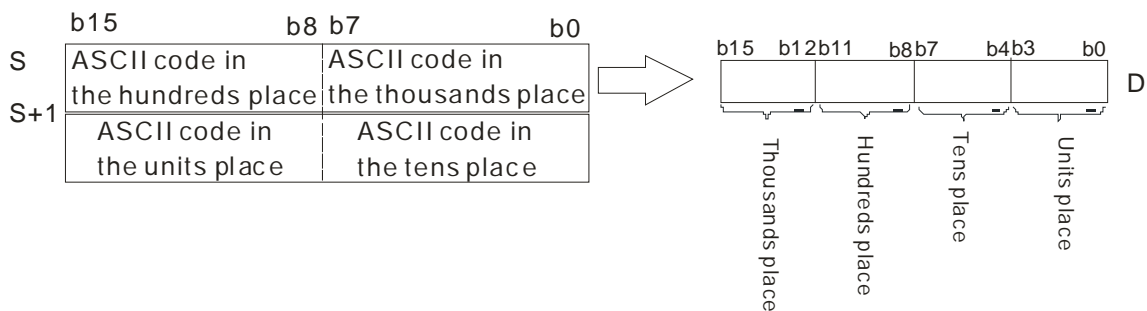


S : Source value

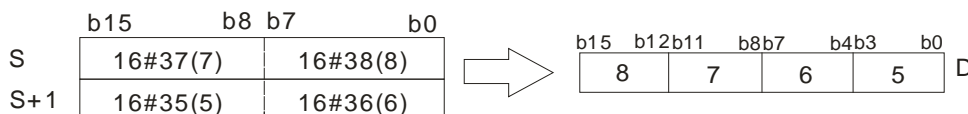
D : Device in which the conversion result is stored

Explanation:

- The ASCII code in **S** is converted into the binary-coded decimal number, and the conversion result is stored in **D**.
- The operand **S** used in the 16-bit instruction occupies two word devices, and the ASCII code in **S** should be within the range between 0000 and 9999. If **S** is a string, the string should be within the range between “0” and “9999”.

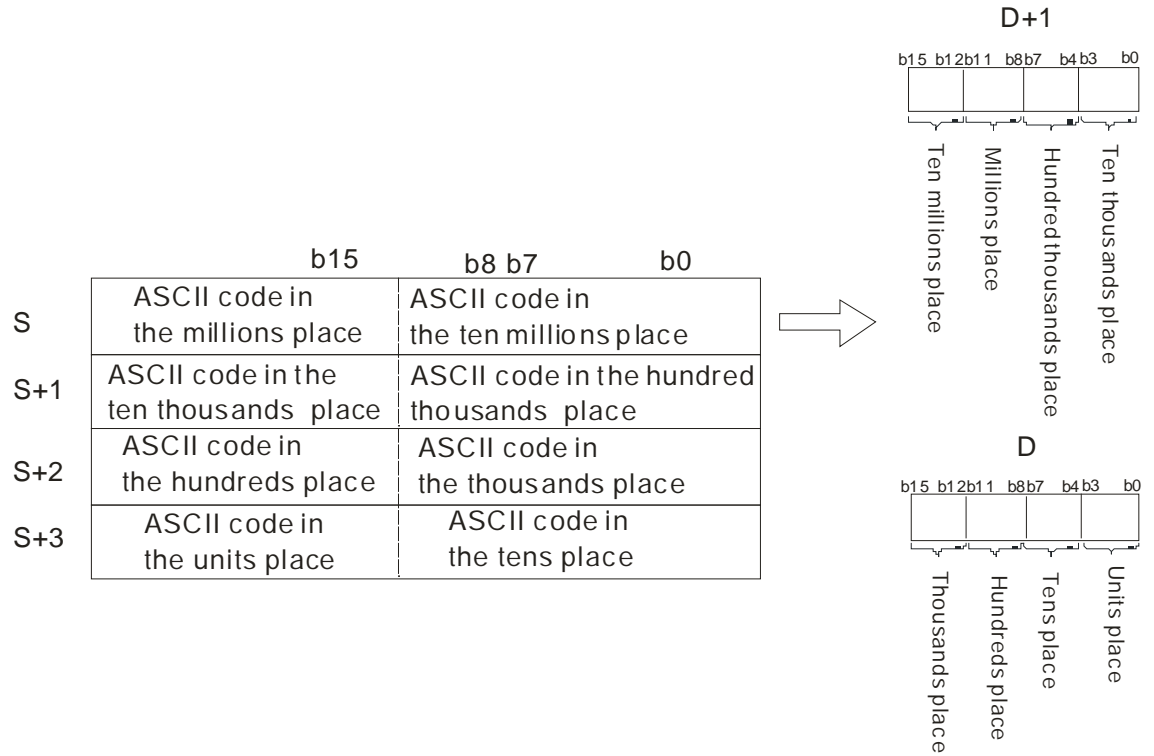


If the ASCII code in **S-S+1** is 8765, the conversion result is as follows.

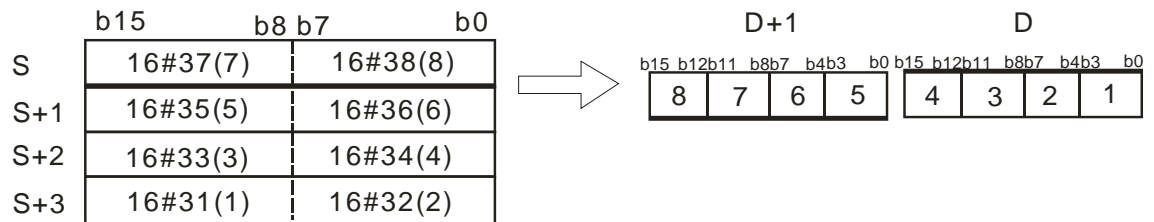


- The operand **S** used in the 32-bit instruction occupies four word devices, and the ASCII code in **S** should

be within the range between 0000000 and 99999999. If **S** is a string, the string should be within the range between "0" and "99999999".



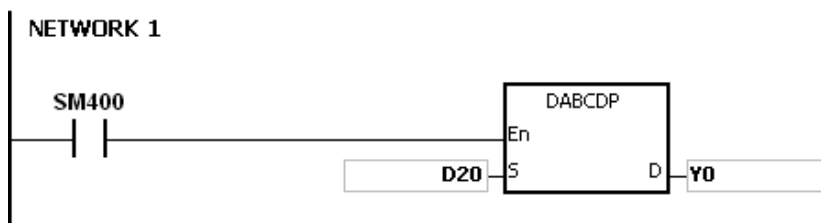
- If the ASCII code in **S~S+3** is 87654321, the conversion result is as follows.

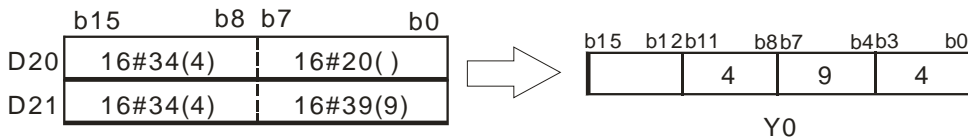


- If the value in **S** is 16#20 or 16#00, the value is processed as 16#30.
- If **S** used in the 16-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 4. If **S** used in the 32-bit instruction is a string, the number of characters contained in the string should be within the range between 1 and 8.

Example 1:

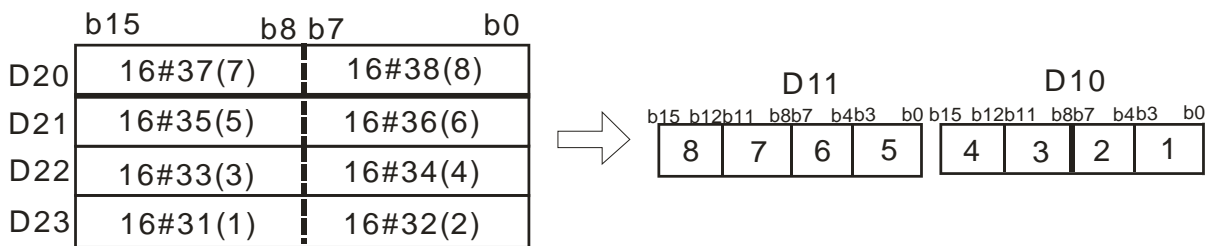
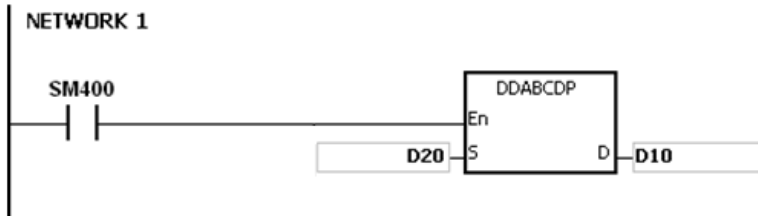
Suppose the values in D20 and D21 are 16#3420 and 16#3439 respectively. When the PLC runs, the value in Y0 is 16#494.





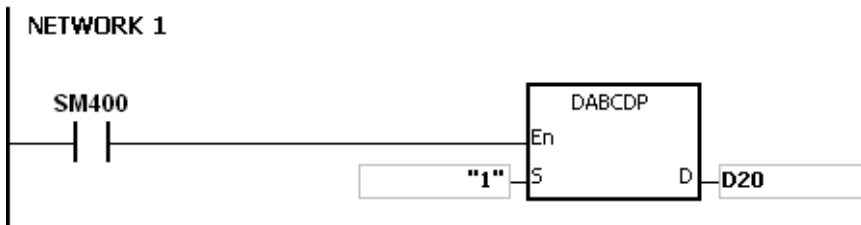
Example 2:

Suppose the values in D20, D21, D22, and D23 are 16#3738, 16#3536, 16#3334, and 16#3132 respectively. When the PLC runs, the value in (D11, D10) is 16#87654321.



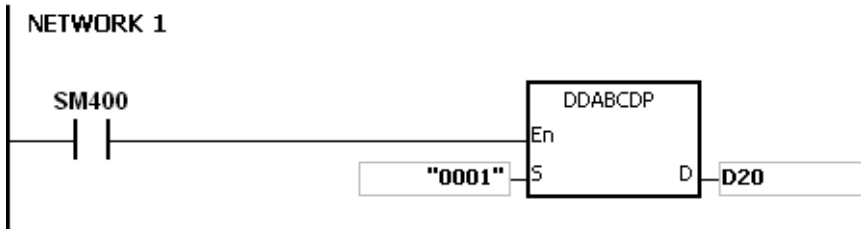
Example 3:

Suppose **S** is the string "1". Since the number of characters contained in the string is less than 4, the string is regarded as "1000". When the PLC runs, the value in D20 is 16#1000.



Example 4:

Suppose **S** is the string "0001". Since the number of characters contained in the string is less than 8, the string is regarded as "00010000". When the PLC runs, the value in (D21, D20) is 16#10000.



Additional remark:

1. If the ASCII code in S is not within the range between 16#30 and 16#39, 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. If the operand S used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will

be ARRAY [2] of WORD/INT.

4. If the operand S used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [4] of WORD/INT.

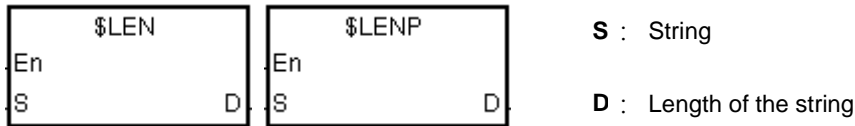
FB/FC	Instruction		Operand				Description					
FC	\$LEN	P	S, D				Calculating the length of the string					

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

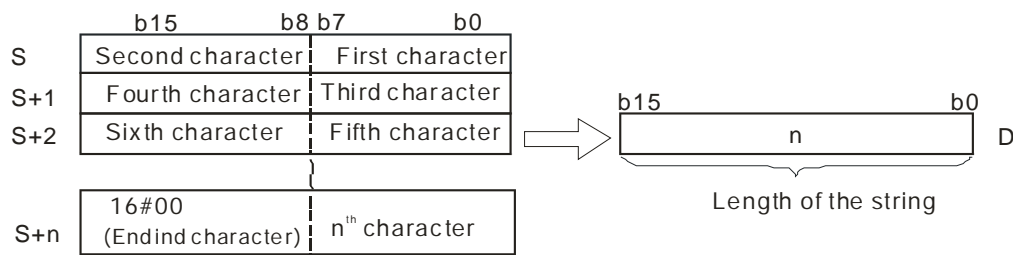


Explanation:

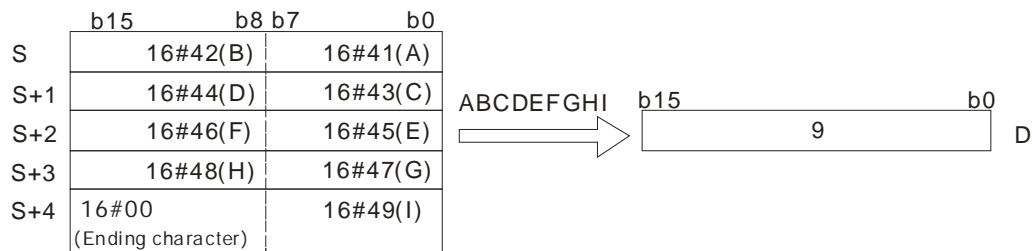
- The length of the string in **S** is calculated, exclusive of 16#00 with which the string ends. The length of the string is stored in **D**.
- The value stored in **D** should be within the range between 0 and 65535.

If the number of characters contained in the string is 65536, which is equal to 16#10000, the value in **D** is 0.

If the number of characters contained in the string is 65537, which is equal to 16#10001, the value in **D** is 0.

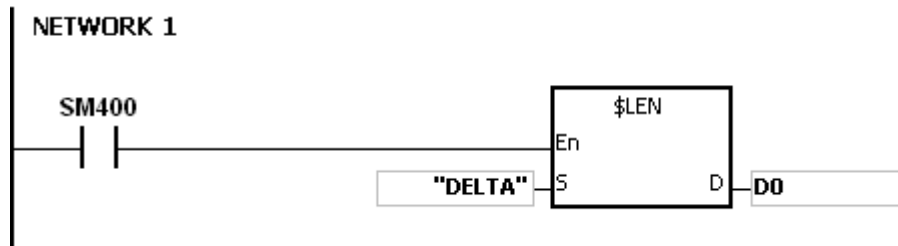


If the data in **S~S+4** is ABCDEFGHI, the calculation result is as follows.



Example 1:

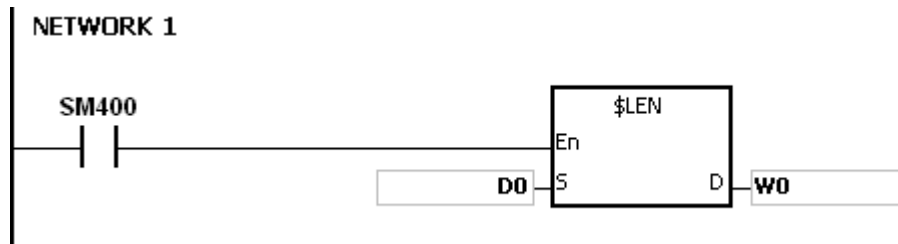
Suppose **S** is the string “DELTA”. When the PLC runs, the value in **D0** is 5.



Example 2:

Suppose the data in D0~D2 is as follows. When the PLC runs, 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#44 (A)



Additional remark:

If the string does not end with 16#00, the instruction is not executed, SM0 is ON, and the error code in SR0 is 6#200E.

FB/FC	Instruction			Operand	Description
FC	D*	\$STR	P	S ₁ , S ₂ , D	Converting the binary number into the string

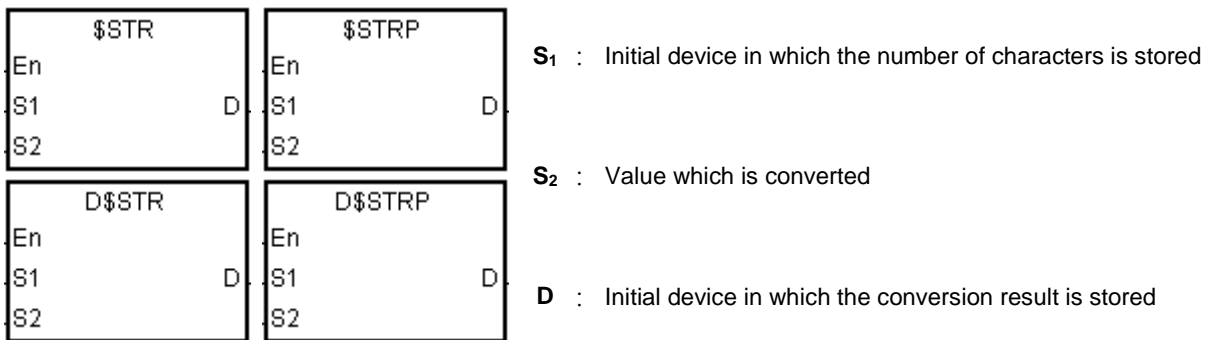
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●	●*				●	●*						
S ₂		●	●*				●	●*						
D		●/●*					●/●*							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●	●		●	○	●				
S ₂	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

3

Graphic expression:



Explanation:

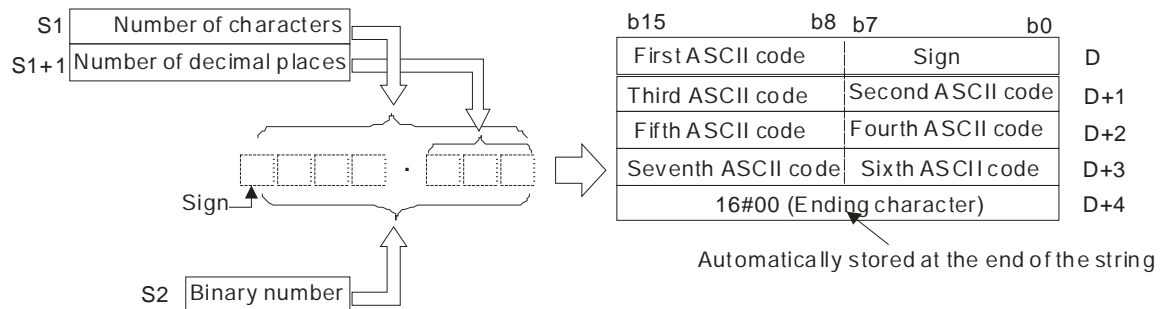
1. A decimal point is added to the value in S₂, the value in S₁+1 indicates the number of decimal places, and the value in S₁ indicates the number of characters. The conversion result is stored in D.

2. \$STR:

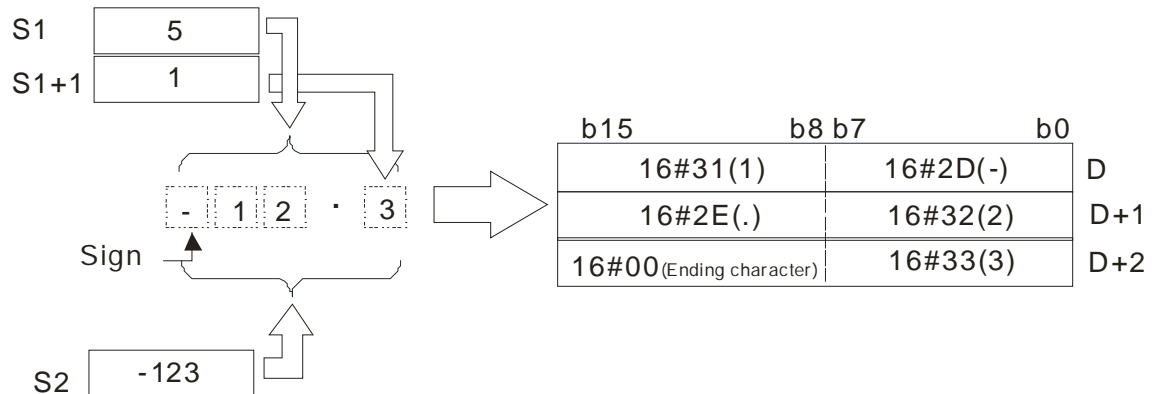
The value in S₁ should be within the range between 2 and 8.

The value in S₁+1 should be within the range between 0 and 5, and should be less than or equal to the value in S₁ minus 3.

The value in S₂ should be within the range between -32768 and 32767.



Suppose the number of characters is 5, the number of decimal places is 1, and the value is -123. The conversion result is as follows.

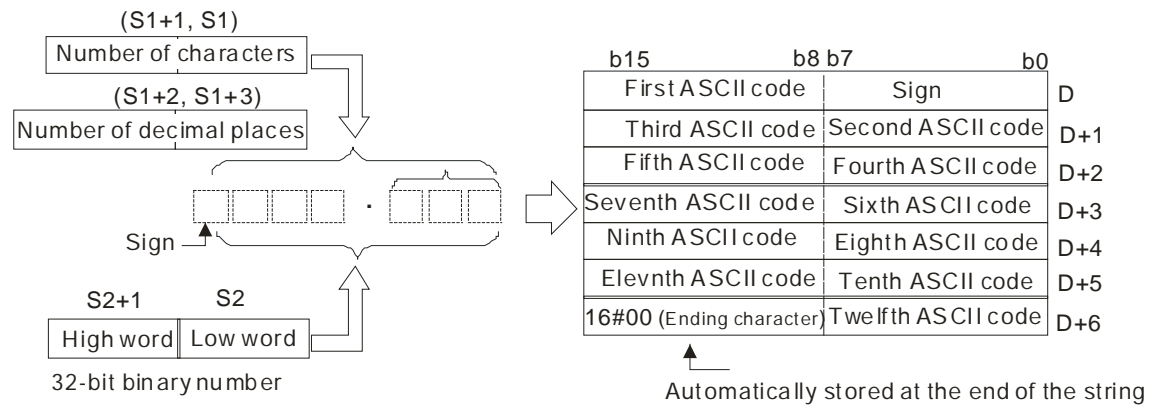


3. **D\$STR:**

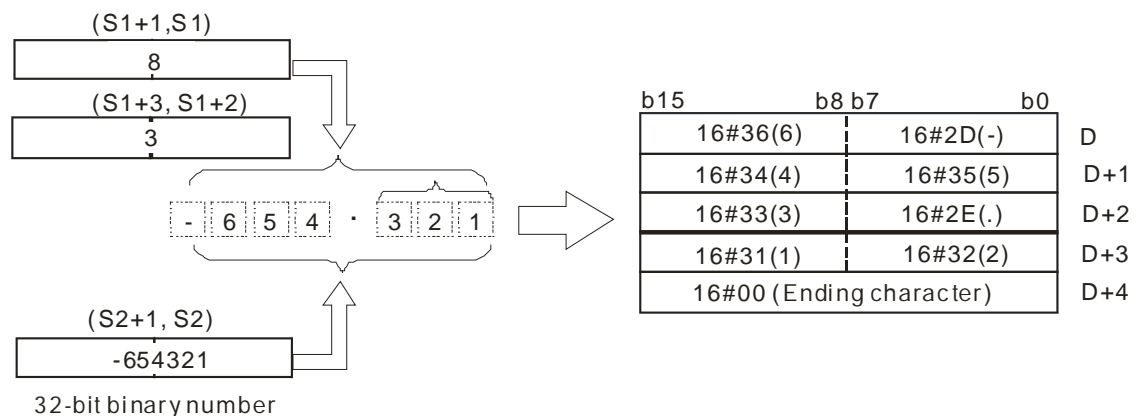
The value in **S₁** should be within the range between 2 and 13.

The value in **S₁+1** should be within the range between 0 and 10, and should be less than or equal to the value in **S₁** minus 3.

The value in **S₂** should be within the range between -2147483648 and 2147483647.



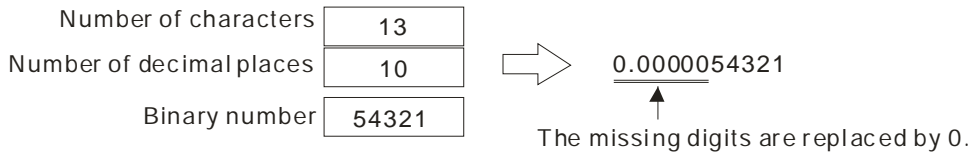
Suppose the number of characters is 8, the number of decimal places is 3, and the value is -654321. The conversion result is as follows.



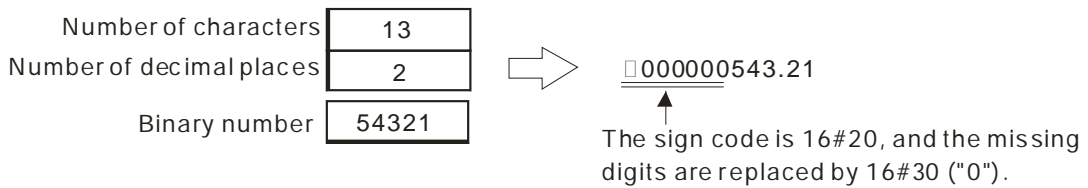
4. If the value in **S₂** is a positive value, the sign code in **D** is 16#20. If the value in **S₂** is a negative value, the

sign code in **D** is 16#2D.

- The code in **D** which represents the decimal point is 16#2E.
- If the value in **S₁+1** is larger than the number of digits in **S₂**, the missing digits are replaced by 0.

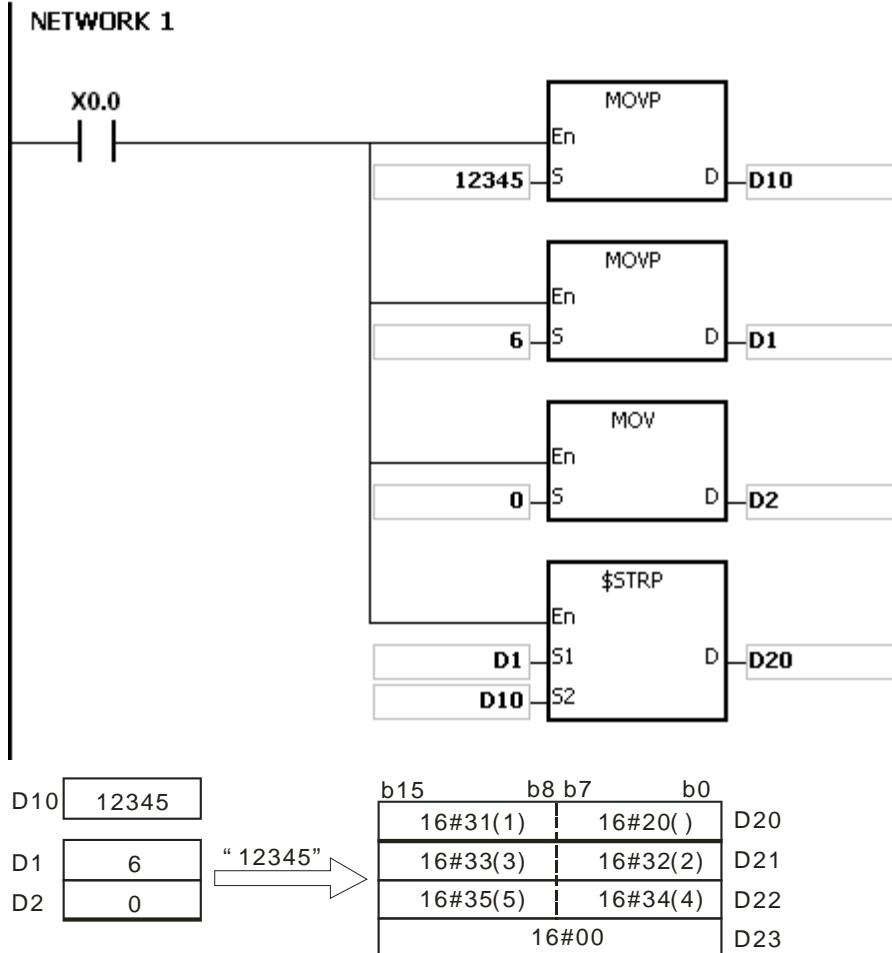


- If the value in **S₁** is larger than the number of digits in **S₂** plus the number of characters which include the decimal point and the sign, the missing digits are replaced by 0.

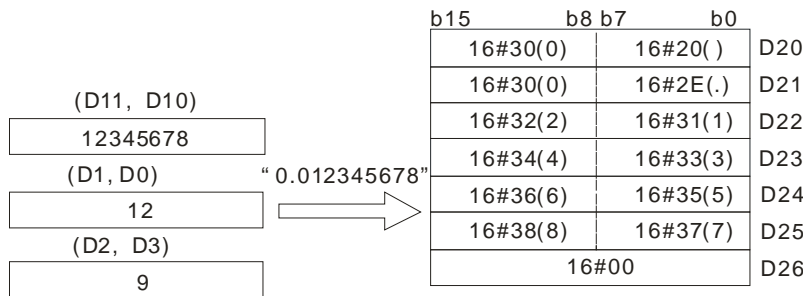
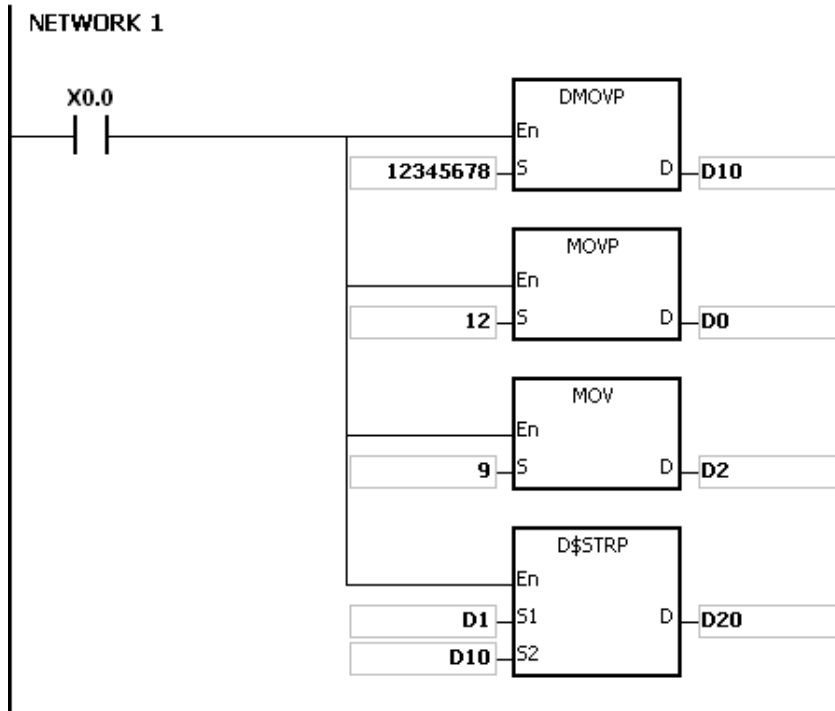


3

Example 1:



Example 2:



Additional remark:

1. If the value in S1 exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the value in S1+1 exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. The value in S1+1 should be less than or equal to the value in S1 minus 3. Otherwise, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the value in S1 is less than the number of digits in S2 plus the number of characters which include the decimal point and the sign, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the operand S1 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
6. If the operand S1 used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of DWORD/DINT.

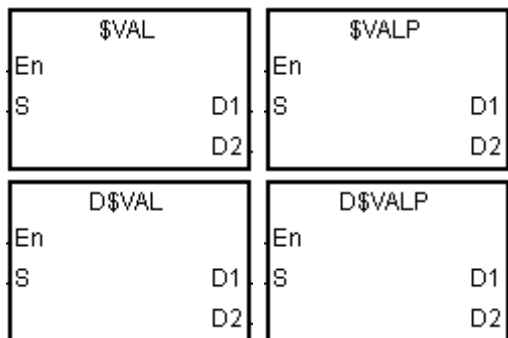
FB/FC	Instruction			Operand	Description
FC	D*	\$VAL	P	S, D ₁ , D ₂	Converting the string into the binary number

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S														●/●*
D ₁		●	●*				●	●*						
D ₂		●	●*				●	●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
D ₁	●	●			●	●		●	●				●				
D ₂	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	AH Motion CPU

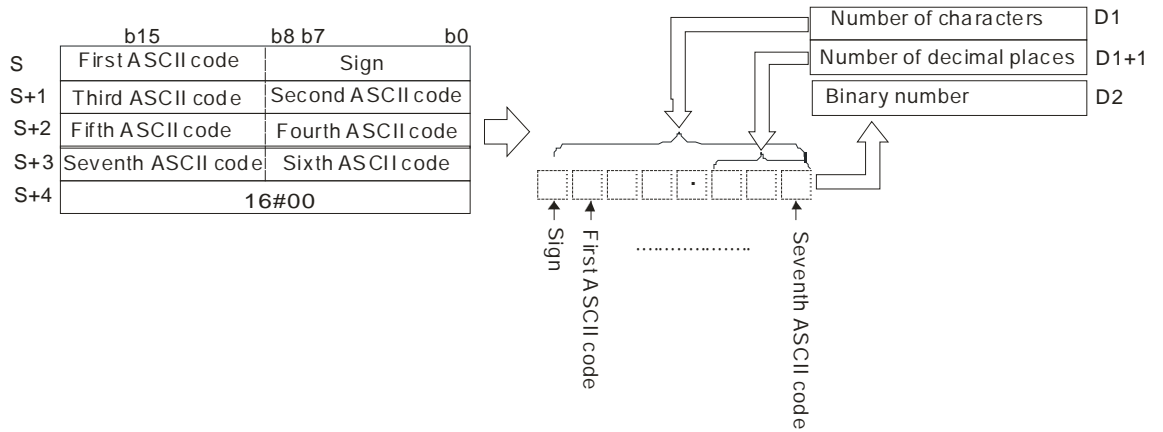
Graphic expression:



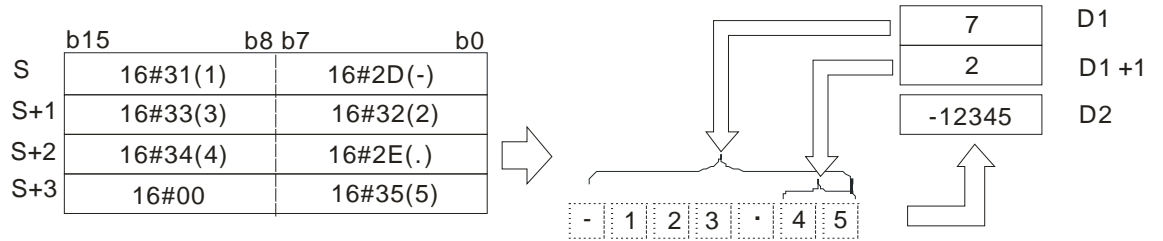
- S** : Source value
- D₁** : Device in which the number of characters is stored
- D₂** : Device in which the binary number is stored

Explanation:

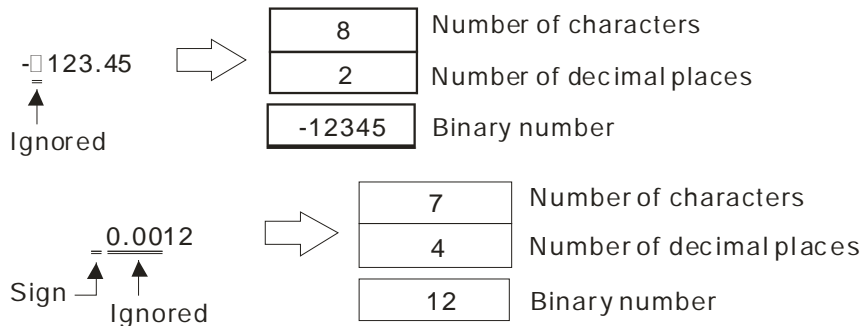
1. The string in **S** is converted into binary number. The number of characters is stored in **D₁**, the number of decimal places is stored in **D₁+1**, and the binary number is stored in **D₂**.
2. **\$VAL**:
 The operand **S** occupies five word devices at most.
 The number of characters contained in the string in **S** should be within the range between 2 and 8.
 If there is a decimal point in the string in **S**, 16#2E should be stored between the first character after the sign character and the last character.



If the data in **S-S+3** is -123.45, the calculation is as follows.



If there is 16#20 or 16#30 between the sign character and the first value which is not 0 in the string, 16#20 or 16#30 is ignored when the string is converted into the binary number.



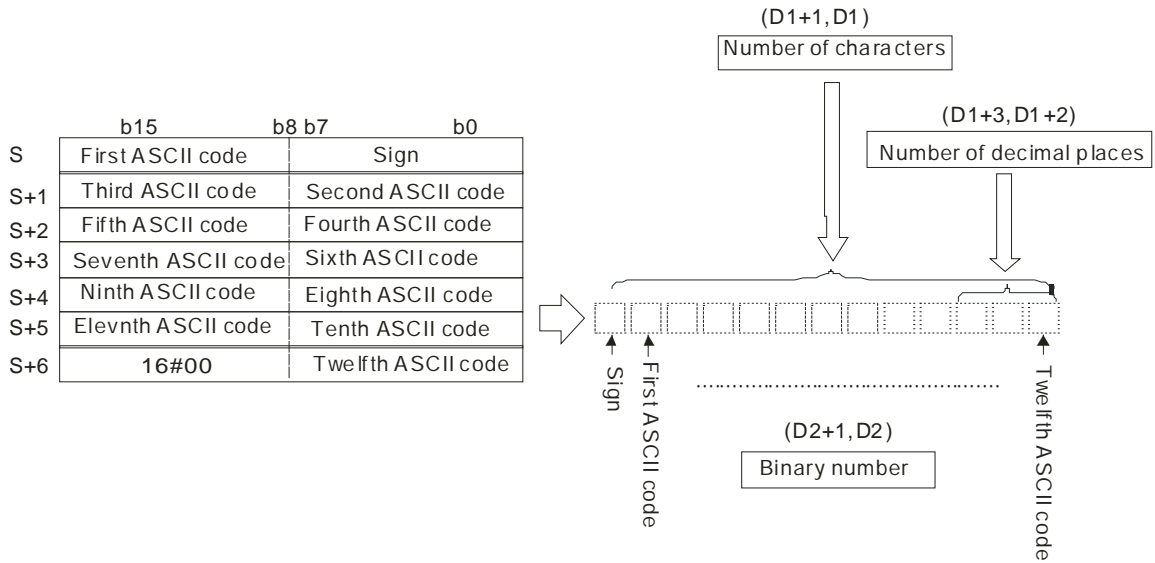
If 16#2E, which represents the decimal point, is ignored, the string in **S** should be within the range between -32768 and 32767. For example, if the string is "1235.3", users have to check whether "12353" is within the range.

3. **D\$VAL:**

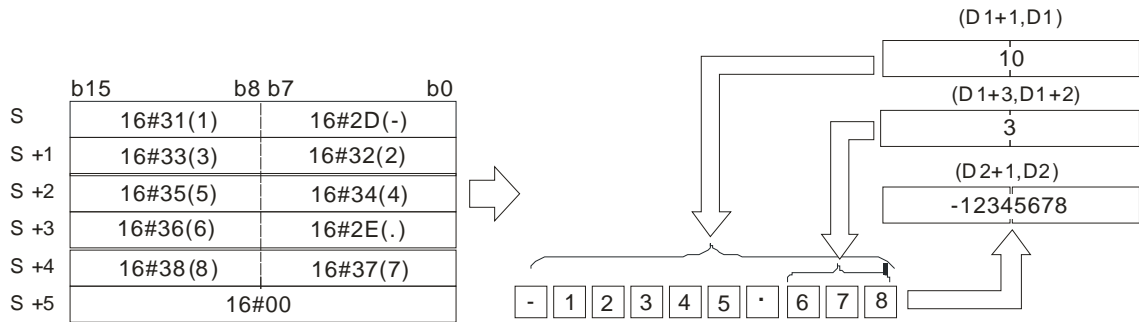
The operand **S** occupies seven word devices at most.

The number of characters contained in the string in **S** should be within the range between 2 and 13.

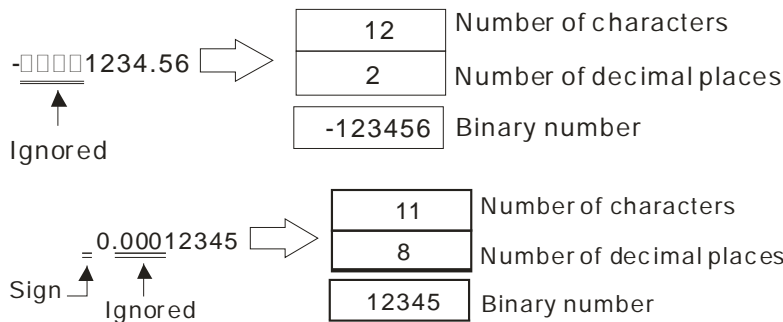
If there is a decimal point in the string in **S**, 16#2E should be stored between the first character after the sign character and the last character.



If the data in **S**~**S**+5 is -12345.678, the calculation is as follows.



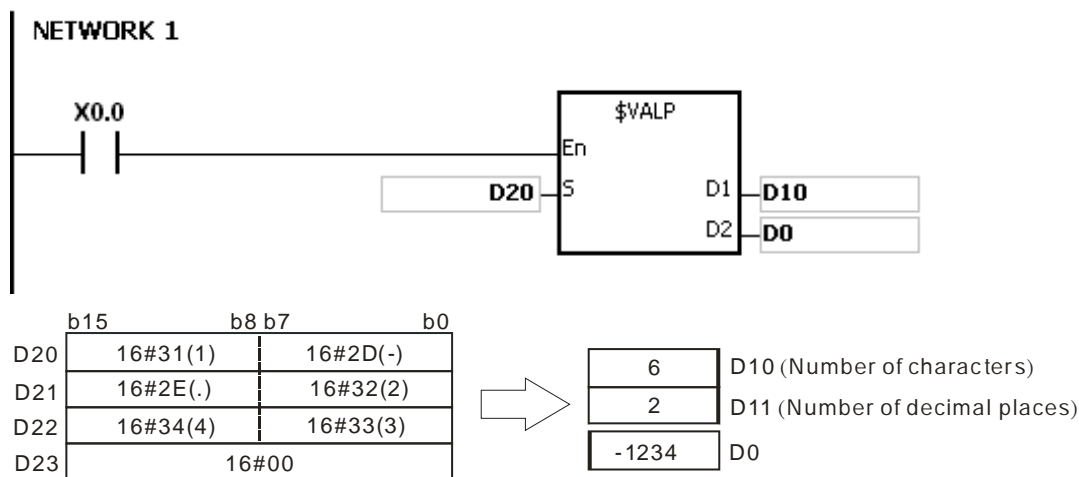
If there is 16#20 or 16#30 between the sign character and the first value which is not 0 in the string in **S**, 16#20 or 16#30 is ignored when the string is converted into the binary number.



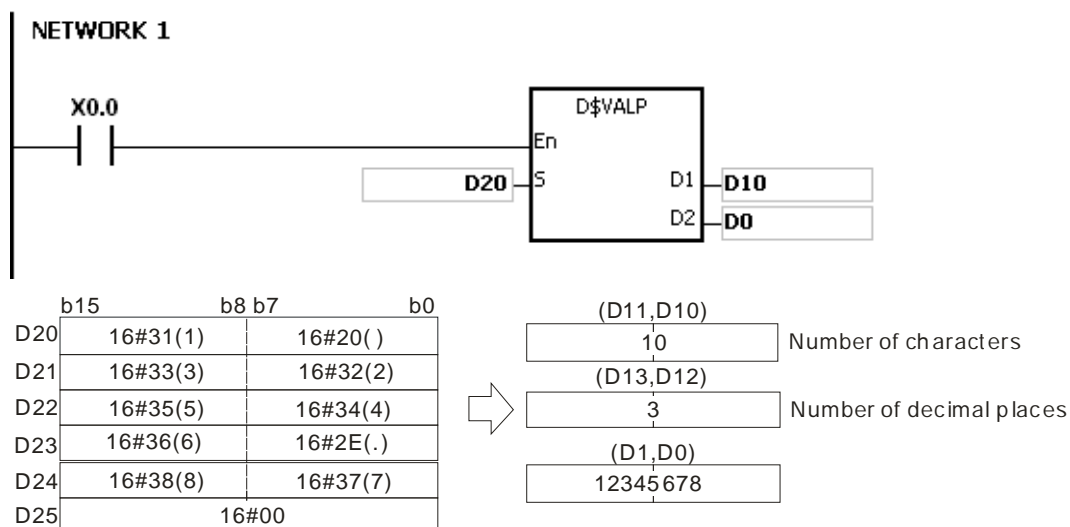
If 16#2E, which represents the decimal point, is ignored, the string in **S** should be within the range between -2147483648 and 2147483647. For example, if the string is "1234567.8", users have to check whether "12345678" is within the range.

4. If the sign code in **S** is 16#20, 16#2B, 16#30, the conversion result is a positive value. If the sign code in **S** is 16#2D, the conversion result is a negative value.
5. In the string in **S**, except for the sign code, the code representing the decimal point, and the code which can be ignored, i.e. 16#20 or 16#30, the other codes have to be within the range between 16#30 and 16#39.

Example 1:



Example 2:



Additional remark:

1. If the number of characters contained in the string in **S** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the sign code in **S** is neither 16#20, 16#30, 16#2B nor 16#2D, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the decimal point in the string in **S** is not stored between the first character after the sign character and the last character, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the binary number converted from the string in **S** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. In the string in **S**, except for the sign code, the code representing the decimal point, and the code which can be ignored, i.e. 16#20 or 16#30, the other codes have to be within the range between 16#30 and 16#39. If the other codes are not within the range between 16#30 and 16#39, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. If the operand D1 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.
7. If the operand D1 used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of DWORD/DINT.

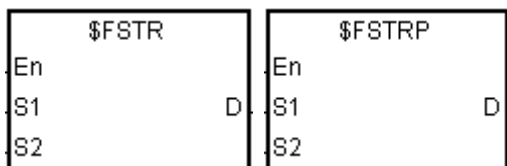
FB/FC	Instruction		Operand	Description
FC	\$FSTR	P	S₁, S₂, D	Converting the floating-point number into the string

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁														●
S₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁	●	●			●	●		●	●		●		●				○
S₂	●	●			●	●		●	●		●		●				
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
-	AH Motion CPU	-

Graphic expression:



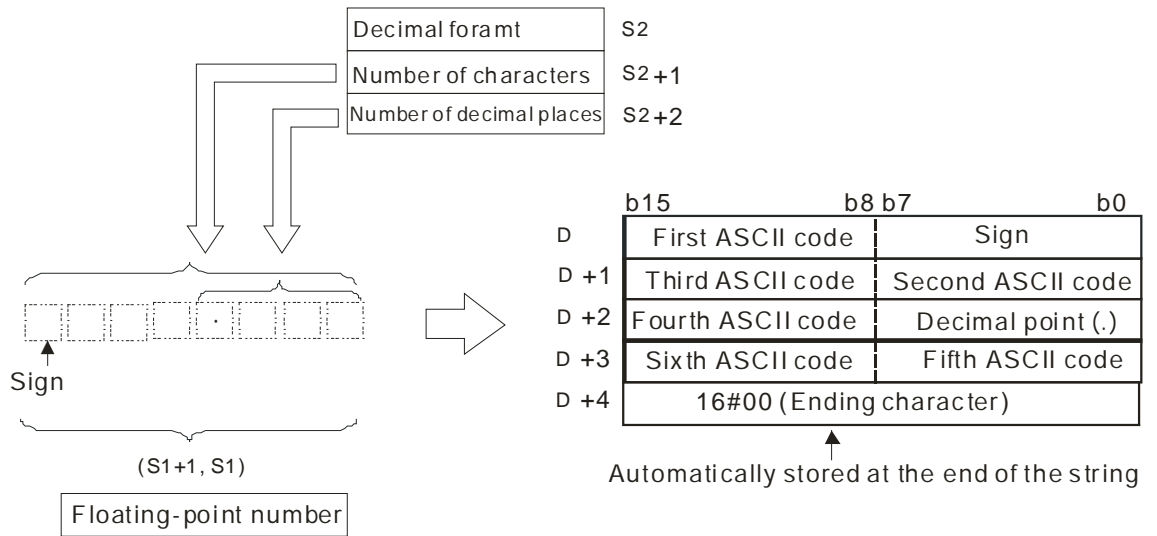
- S₁** : Source value
- S₂** : Initial device in which the format is stored
- D** : Initial device in which the conversion result is stored

Explanation:

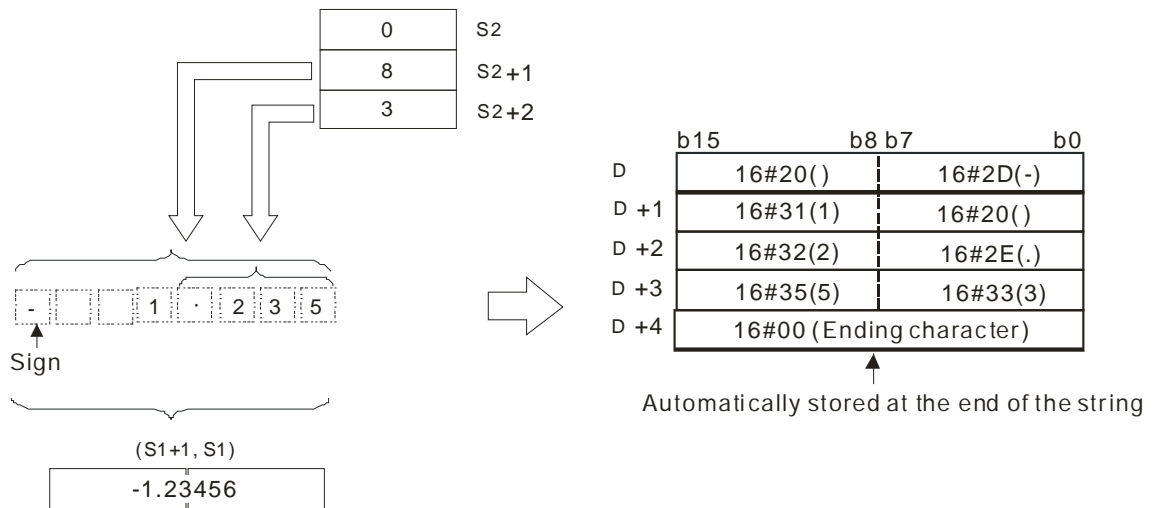
1. The floating-point number in **S₁** is converted into the string in accordance with the setting of **S₂**, and the conversion result is stored in **D**.
2. The conversion result varies with the setting of **S₂**.
3. The value in **S₂+1** should be within the range between 2 and 24.

S2	0: Decimal format 1: Exponential
S2+1	Number of characters
S2+2	Number of decimal places

4. Decimal format

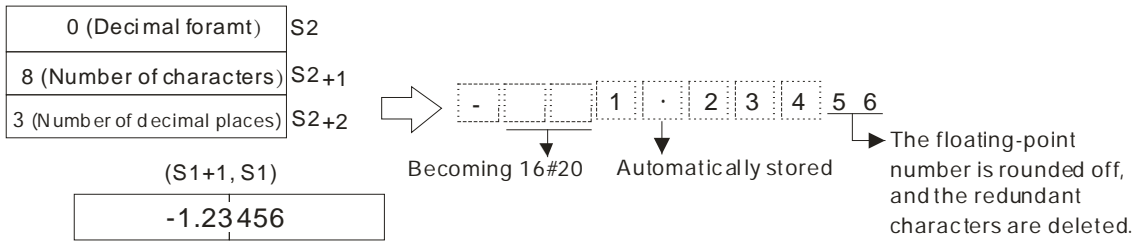


Suppose the number of characters is 8, the number of decimal places is 2, and the value is -1.23456. The calculation is as follows.

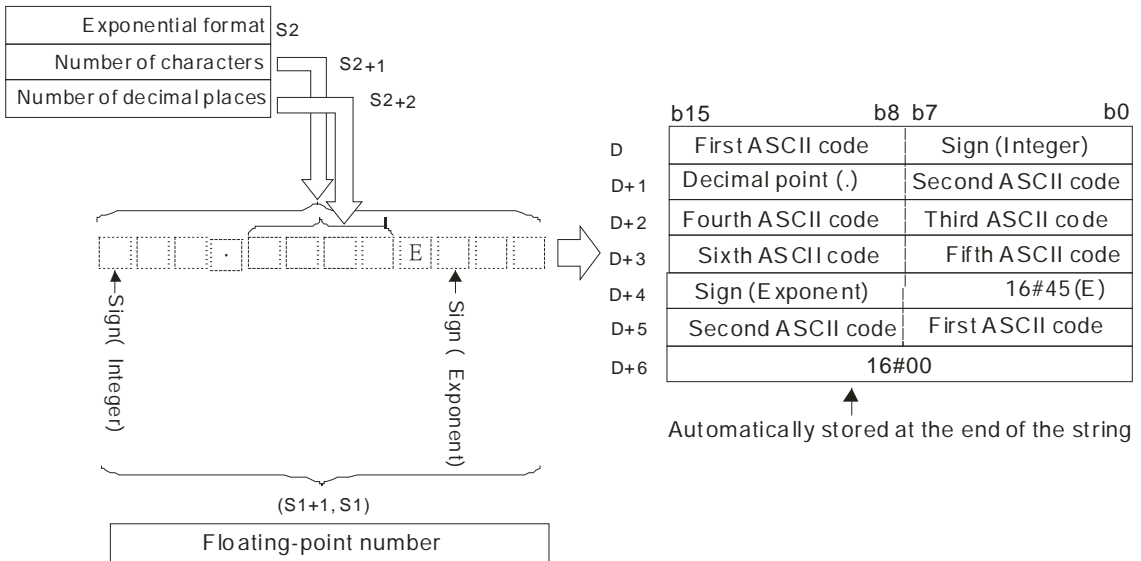


- The value in S_2+1 :
If the value in S_2+2 is 0, the value in S_2+1 should be within the range between 2 and 24, and the number of characters which the integer part contains should be less than or equal to 23.
If the value in S_2+2 is not 0, the value in S_2+1 should be within the range between the value in S_2+2 plus 3 and 24, and the number of characters which the integer part contains should be less than or equal to 22 minus the value in S_2+2 .
- The value in S_2+2 should be within the range between 0 and 7. If the value in S_2+2 is not 0, it should be less than or equal to the value in S_2+1 minus 3.
- If the floating-point number in S_1 is a positive number, the sign code in D is 16#20. If the floating-point number in S_1 is a negative number, the sign code in D is 16#2D.
- If the length of the floating-point number is larger than the value in S_2+1 , the floating-point number is rounded off, and the redundant characters are deleted.
- If the value in S_2+2 is larger than 0, 16#2E (“.”) is stored in front of the specified character automatically.
- If the length of the conversion result is less than the value in S_2+1 , the codes between the sign character and the real number are 16#20.

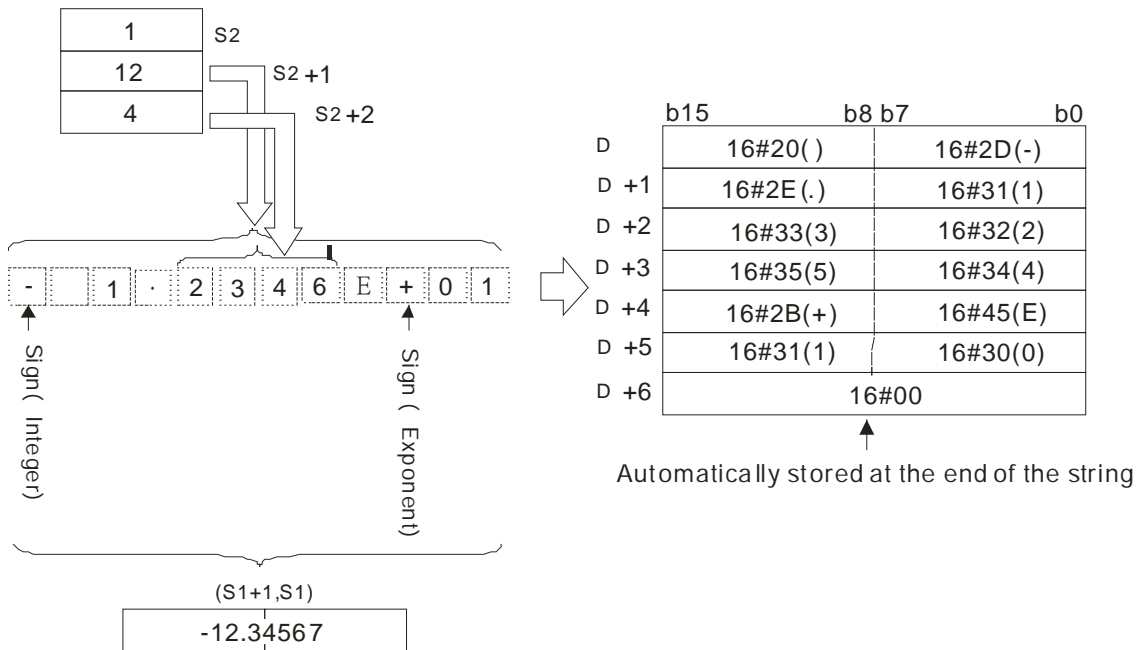
- The conversion result ends with 16#00.



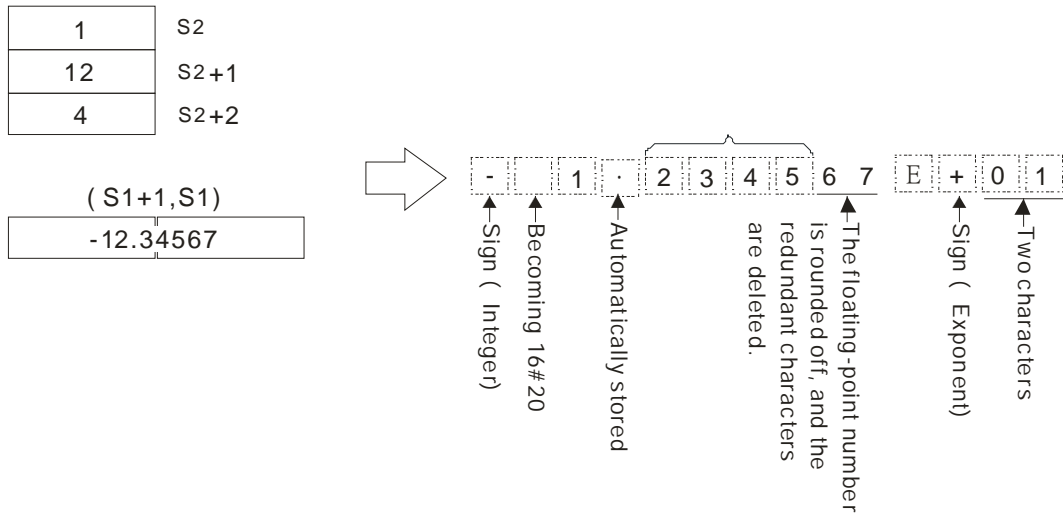
5. Exponential format



Suppose the number of characters is 12, the number of decimal places is 4, and the value is -12.34567. The calculation is as follows.

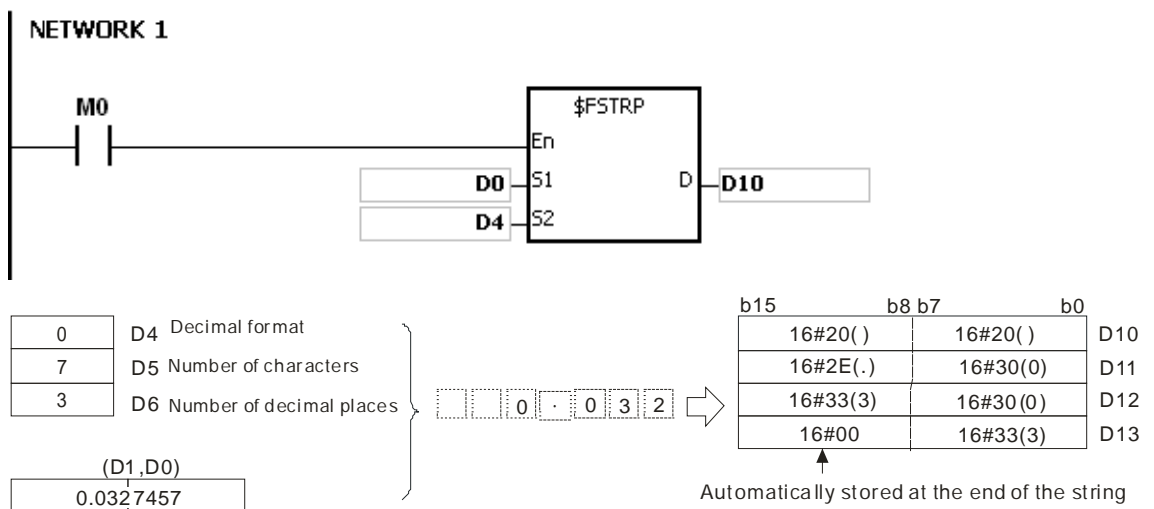


- The value in S2+1:
- If the value in S2+2 is 0, the value in S2+1 should be within the range between 6 and 24.
- If the value in S2+2 is not 0, the value in S2+1 should be within the range between the value in S2+2 plus 7 and 24.
- The value in S2+2 should be within the range between 0 and 7. If the value in S2+2 is not 0, it should be less than or equal to the value in S2+1 minus 7.
- If the floating-point number in S1 is a positive number, the sign code in D is 16#20. If the floating-point number in S1 is a negative number, the sign code in D is 16#2D.
- The integer part contains one character. To fulfill the number of characters, the codes between the sign code and the integer part are 16#20.
- If the value in S2+2 is larger than 0, 16#2E (“.”) is stored in front of the specified character automatically.
- If the exponent is a positive number, the sign code in D is 16#2B. If the exponent is a negative number, the sign code in D is 16#2D.
- The exponent part contains two characters. If there is only one character, the other character is “0” (16#30).
- The conversion result ends with 16#00.



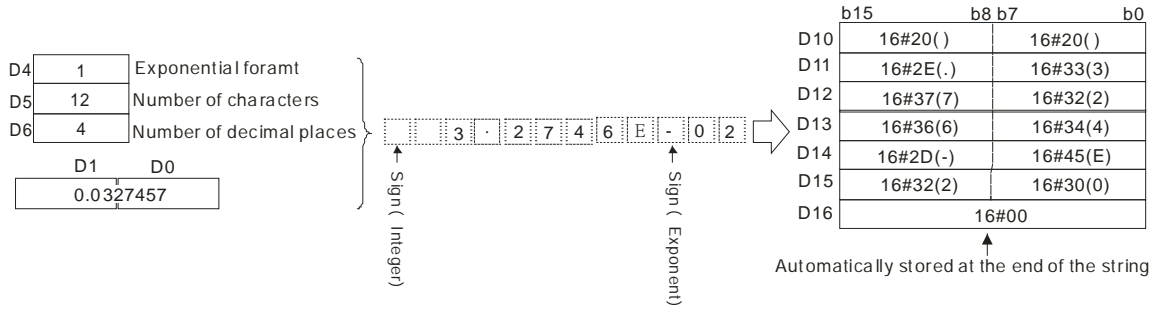
Example 1:

Suppose the value in D4 is 0. The floating-point number in (D1, D0) is converted into the decimal format of the string.



Example 2:

Suppose the value in D4 is 1. The floating-point number in (D1, D0) is converted into the exponential format of the string.



Additional remark:

1. If the value in **S₁** exceeds the range of values which 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.
2. If the value in S2 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 S2+1 exceeds the range below, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
 - The decimal format:
If the value in S2+2 is 0, the value in S2+1 should be within the range between 2 and 24, and the number of characters which the integer part contains should be less than or equal to 23.
If the value in S2+2 is not 0, the value in S2+1 should be within the range between the value in S2+2 plus 3 and 24, and the number of characters which the integer part contains should be less than or equal to 22 minus the value in S2+2
 - The exponential format:
If the value in S2+2 is 0, the value in S2+1 should be within the range between 6 and 24.
If the value in S2+2 is not 0, the value in S2+1 should be within the range between the value in S2+2 plus 7 and 24.
4. If the value in **S₂+2** exceeds the range below, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
 - The decimal format:
The value in **S₂+2** should be within the range between 0 and 7. Besides, it should be less than or equal to the value in **S₂+1** minus 3.
 - The exponential format:
The value in **S₂+2** should be within the range between 0 and 7. Besides, it should be less than or equal to the value in **S₂+1** minus 7.
5. If users declare the operand **S₂** in ISPSOft, the data type will be ARRAY [3] of WORD/INT.

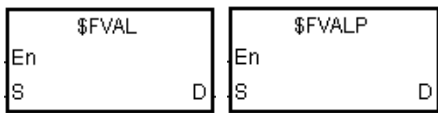
FB/FC	Instruction		Operand		Description
FC	\$FVAL	P	S, D		Converting the string into the floating-point number

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●		●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

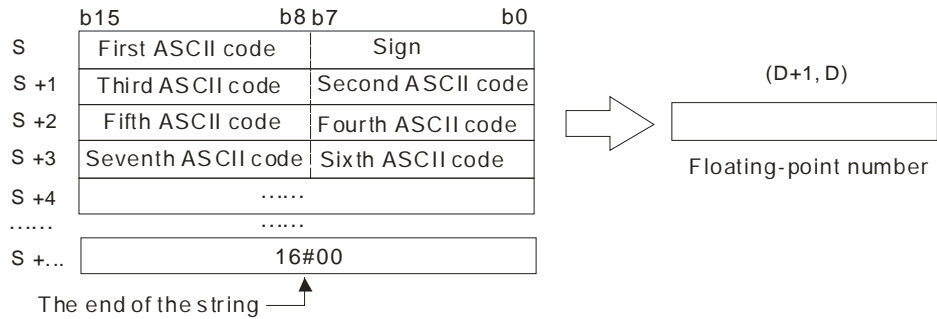


S : Source value

D : Device in which the conversion result is stored

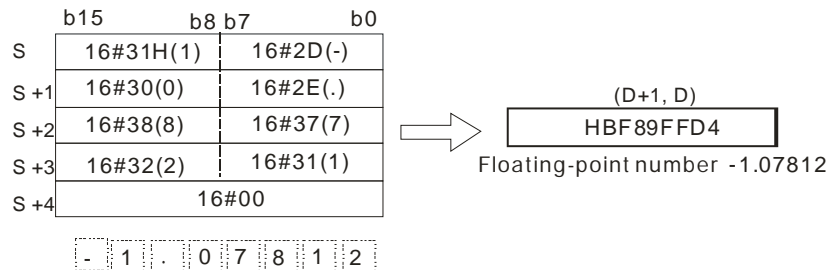
Explanation:

- The string in **S** is converted into the floating-point number, and the conversion result is stored in **D**.

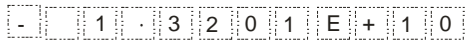
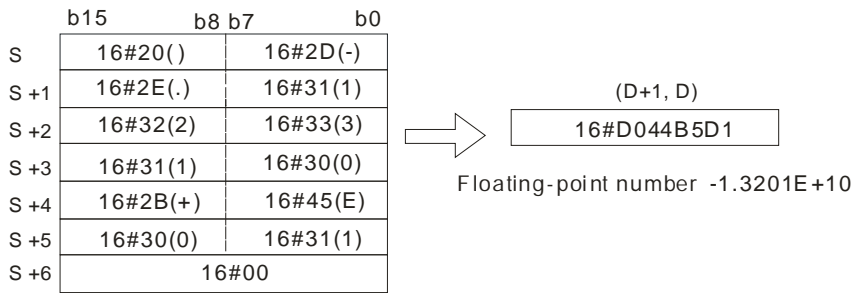


- The string in **S** can be the decimal format of the string or the exponential format of the string.

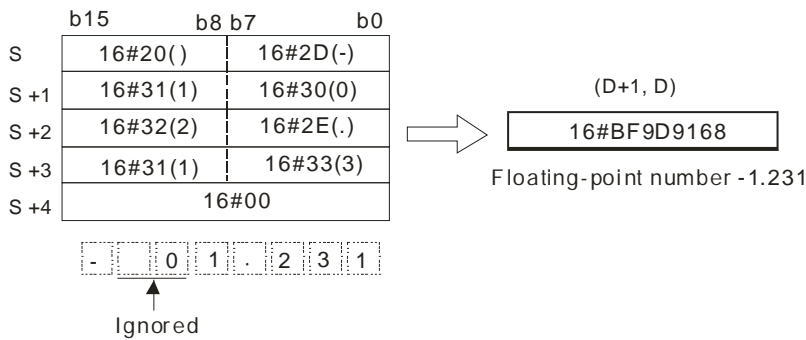
- The decimal format:



- The exponential format:

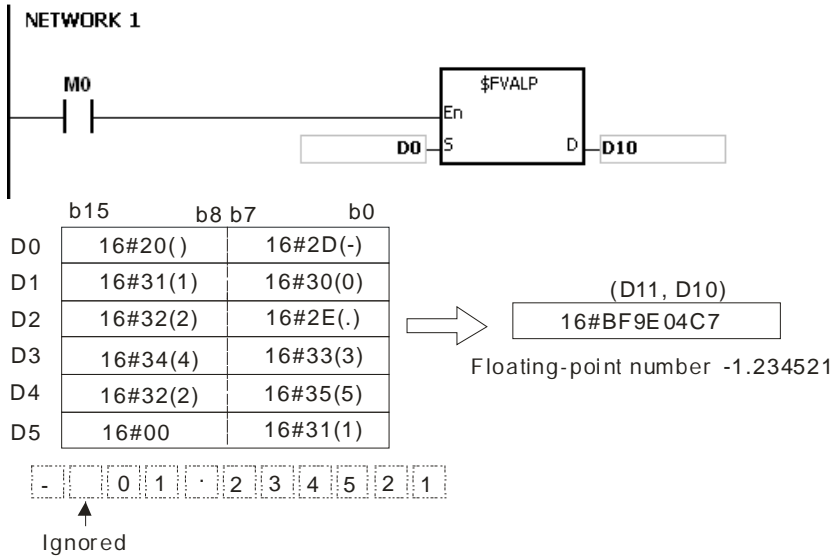


3. If the sign code in **S** is 16#20, 16#30, 16#2B, the conversion result is a positive value. If the sign code in **S**₁ is 16#2D, the conversion result is a negative value.
4. 16#20 or 16#30 is ignored during the conversion, as the example below shows.

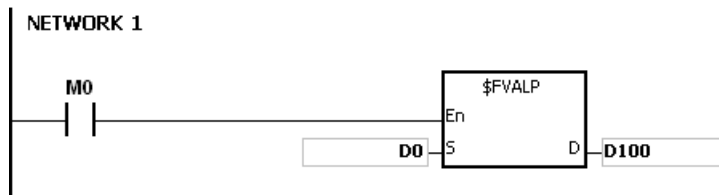


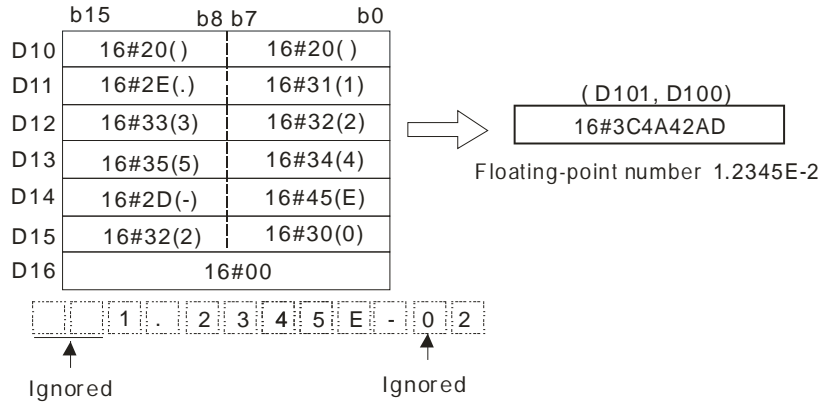
5. **S** : 24 characters at most can be contained in the string.

Example 1:



Example 2:





Additional remark:

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. If the length of the string in **S** exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the sign code in **S** is neither 16#20, 16#30, 16#2B nor 16#2D, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If there is more than one 16#2E ("."), 16#2B ("+"), or 16#2D ("-") in the string in **S**, exclusive of 16#2D ("-") with which the string starts, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the characters which constitute the integer part and the characters which constitute the fractional part in the string in **S** are not within the range between 16#30 ("0") and 16#39 ("9"), the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. The character in the exponent part in the string in **S** only can be "E" (16#45), "+" (16#2B), "-" (16#2D), or the number between "0" (16#30) and "9" (16#39). Otherwise, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
7. If the conversion result exceeds the range of values which 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.

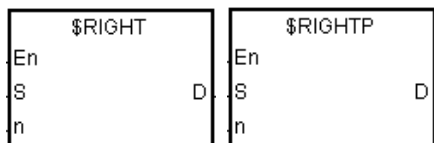
FB/FC	Instruction		Operand				Description			
FC	\$RIGHT	P	S, n, D				The retrieve of the characters in the string begins from the right.			

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

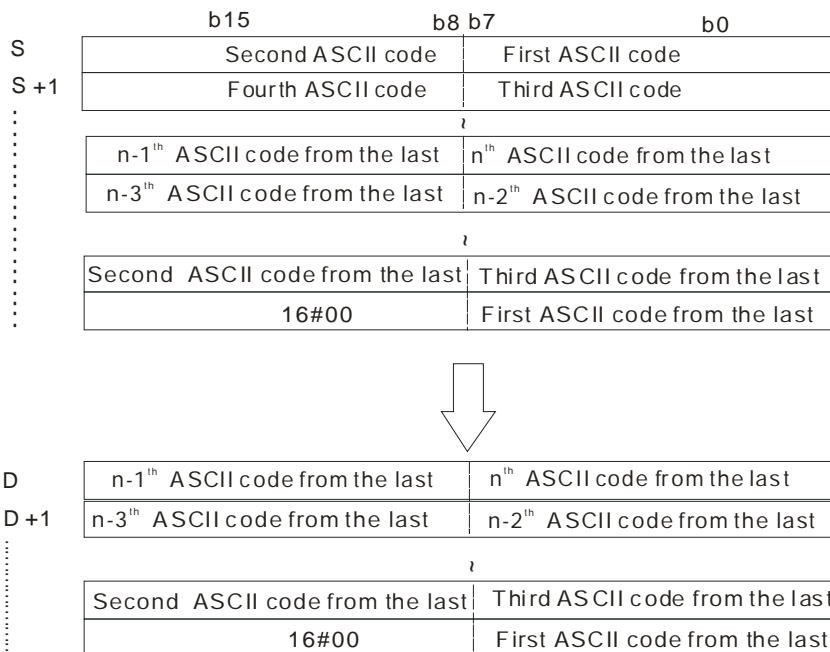
Graphic expression:



- S** : String
- n** : Number of characters which are retrieved
- D** : Device in which the characters retrieved are stored

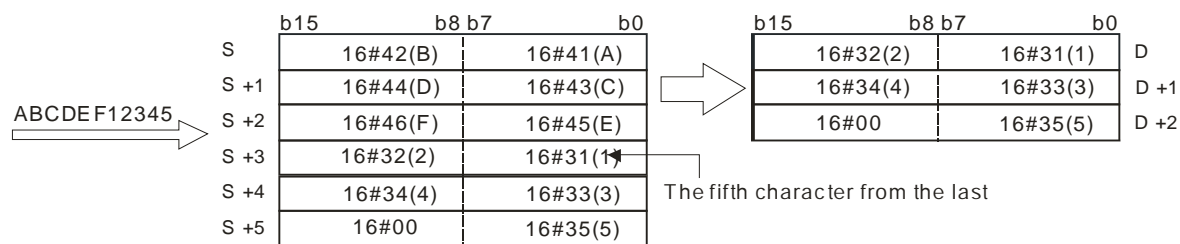
Explanation:

- The instruction is used to retrieve **n** characters in the string in **S** from the right, and the characters which are retrieved are stored in **D**.
- If **n** is 0, the value in **D** is 0.

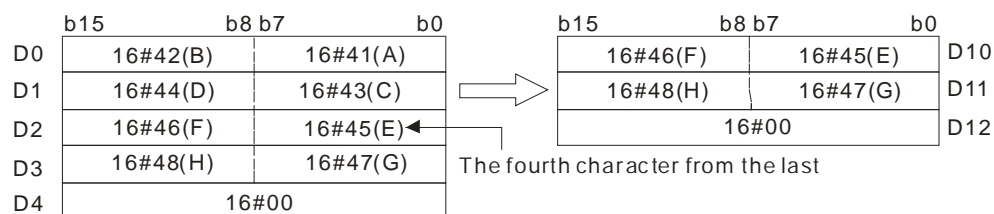
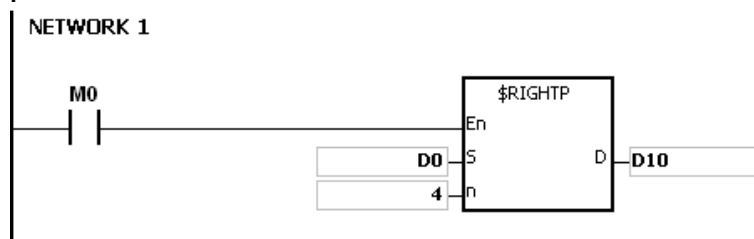


If the data in **S** is ABCDEF12345 and **n** is 5, five characters in the string in **S** are retrieved from the right. The

conversion result is as follows.



Example:



Additional remark:

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. If **n** is less than 0, or if **n** is larger than the length of the string in **S**, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
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.

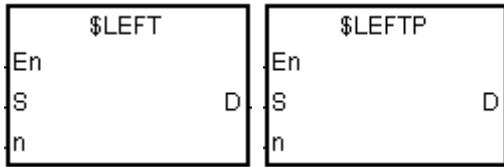
FB/FC	Instruction		Operand				Description						
FC	\$LEFT	P	S, n, D				The retrieve of the characters in the string begins from the left.						

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●			●	●		●	●		●		●			○	
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

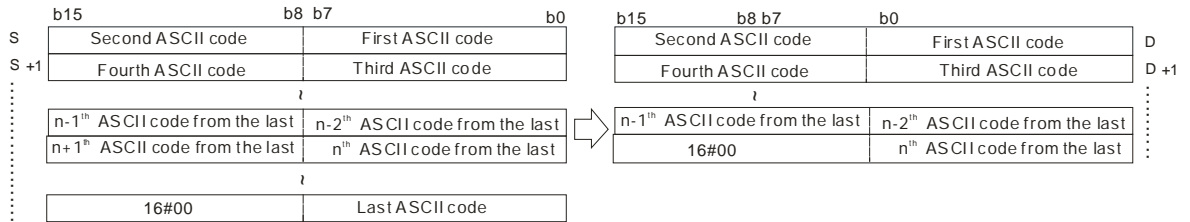
Graphic expression:



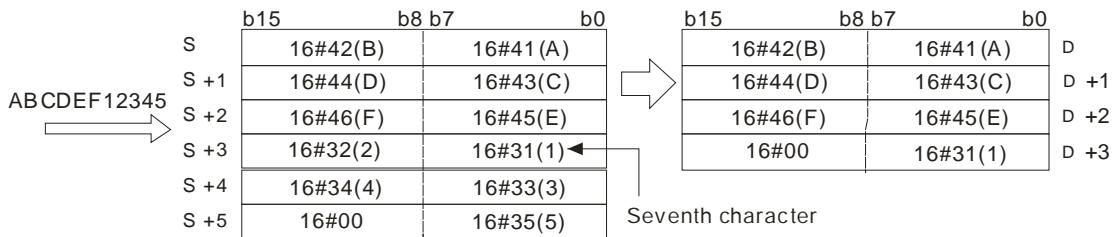
- S : String
- n : Number of characters which are retrieved
- D : Device in which the characters retrieved are stored

Explanation:

- The instruction is used to retrieve n characters in the string in S from the left, and the characters which are retrieved are stored in D.
- If n is 0, the value in D is 0.

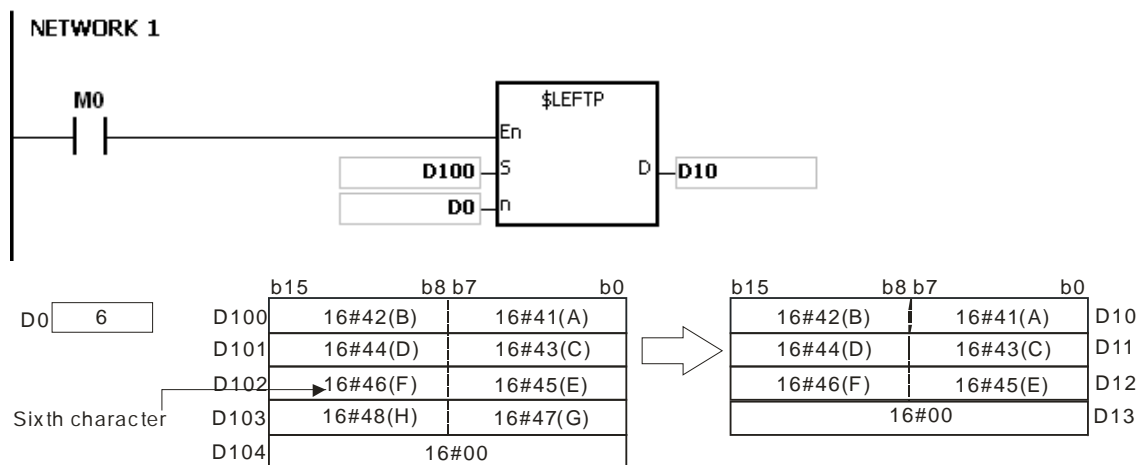


If the data in S is ABCDEF12345 and n is 7, seven characters in the string in S are retrieved from the left. The conversion result is as follows.



Example:

When M0 is ON, the instruction \$LEFT is executed. The six characters starting from the character in D100 are retrieved, and stored in D10~D12.



Additional remark:

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. If n is less than 0, or if n is larger than the length of the string in S, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
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.

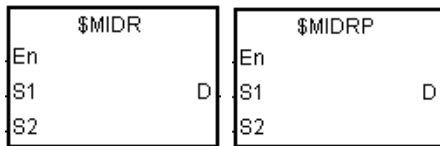
FB/FC	Instruction		Operand	Description
FC	\$MIDR	P	S ₁ , S ₂ , D	Retrieving a part of the string

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁														●
S ₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●			●		●			○	
S ₂	●	●			●	●		●			●	○	●				
D	●	●			●	●		●					●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

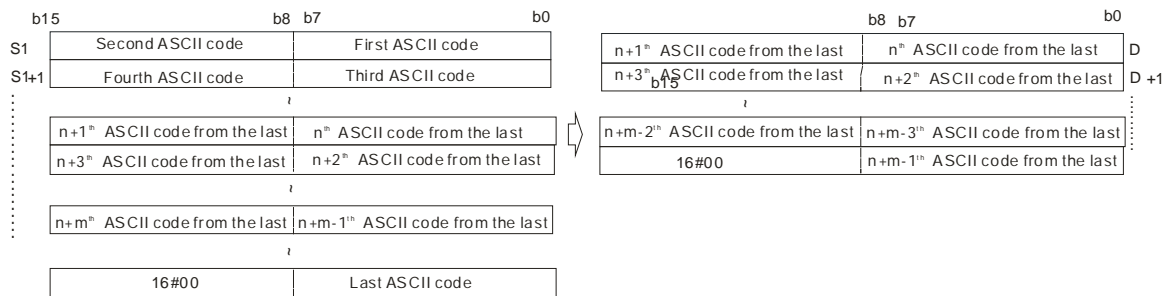
Graphic expression:



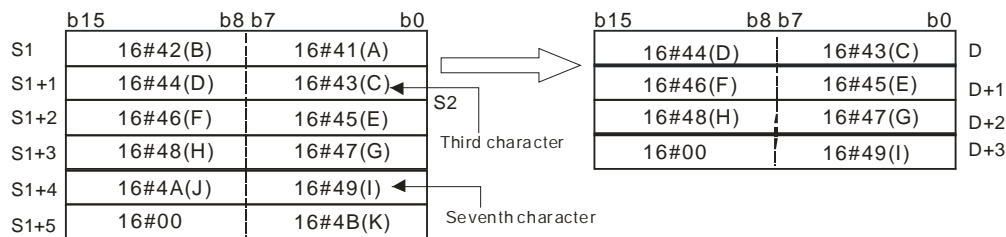
- S₁ : String
- S₂ : Part of the string which is retrieved
- D : Device in which the characters retrieved are stored

Explanation:

- Suppose the values in S₂ and S₂+1 are n and m respectively. The m characters starting from the nth character in the string in S₁ are retrieved, and stored in D.



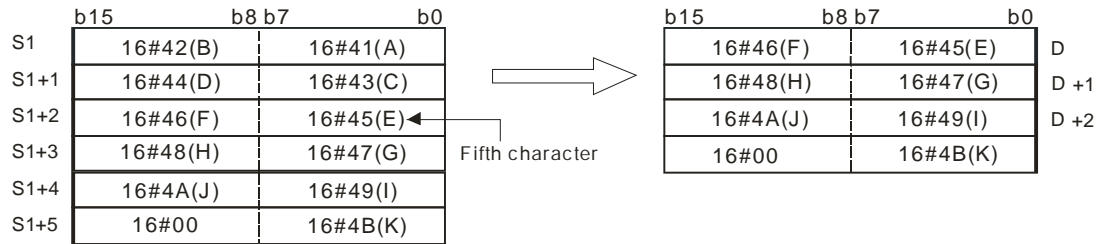
If the data in S₁ is ABCDEFGHIJK, the value in S₂ is 3, and the value in S₂+1 is 7, the seven characters starting from the third characters in the string are retrieved from the left. The conversion result is as follows.



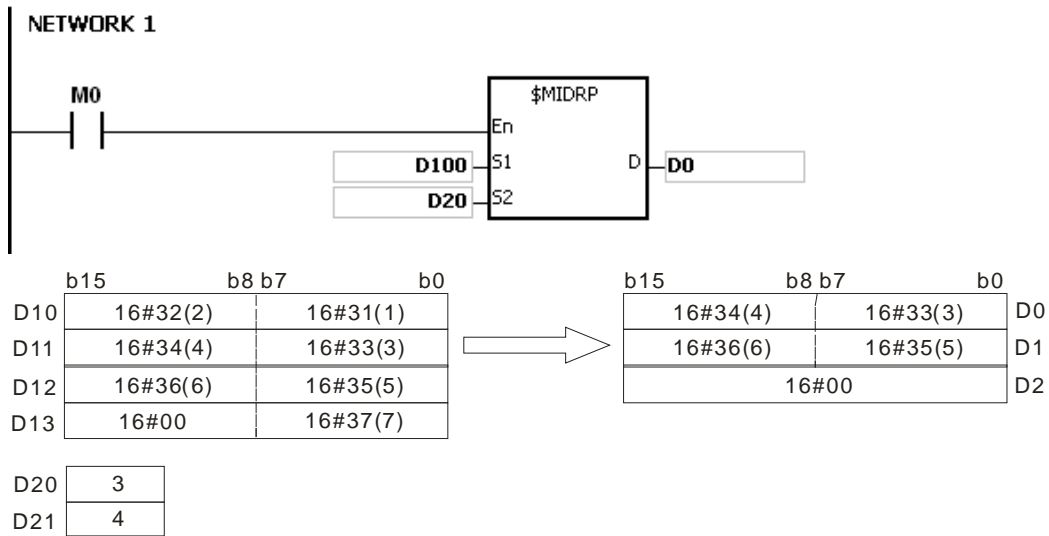
- If the value in S₂+1 is 0, the instruction is not executed.
- If the value in S₂+1 is -1, the characters in S₁ starting from the character indicated by the value in S₂ to the

last character in S_1 are retrieved.

- 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:



Additional remark:

- If the string in S_1 does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
- If the value in S_2 is less than or equal to 0, or if the value in S_2+1 is less than -1, SM0 is ON, and the error code in SR0 is 16#2003.
- If the value in S_2 is larger than the length of the string in S_1 , SM0 is ON, and the error code in SR0 is 16#2003.
- If the value in S_2+1 is larger than the number of characters which can be retrieved from the string in S_1 , SM0 is ON, and the error code in SR0 is 16#2003.
- If the operand S_2 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.

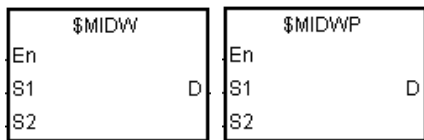
FB/FC	Instruction		Operand	Description
FC	\$MIDW	P	S ₁ , S ₂ , D	Replacing a part of the string

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁														●
S ₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●	●		●		●			○	
S ₂	●	●			●	●		●	●		●	○	●				
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

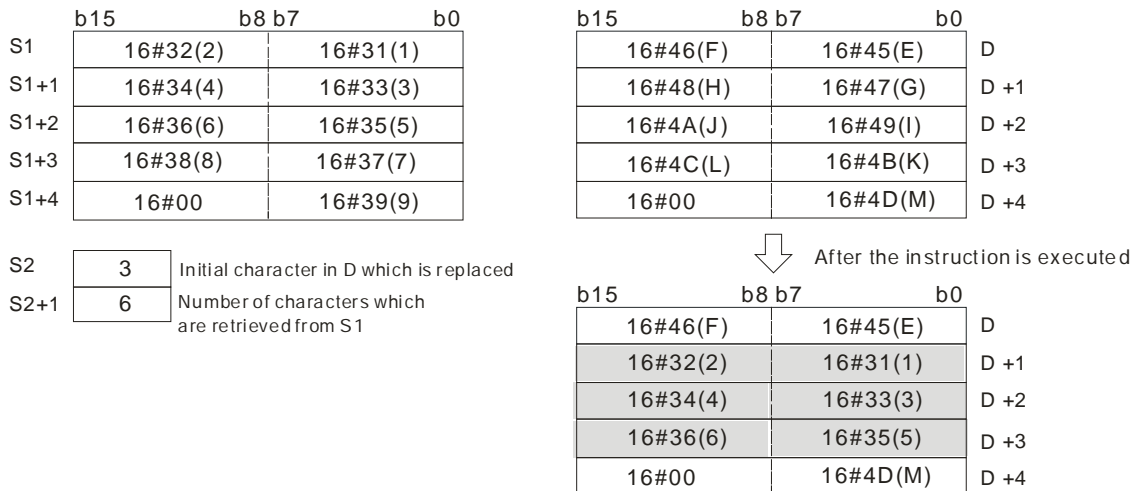
Graphic expression:



- S₁ : String
- S₂ : Part of the string which is replaced
- D : String which is replaced

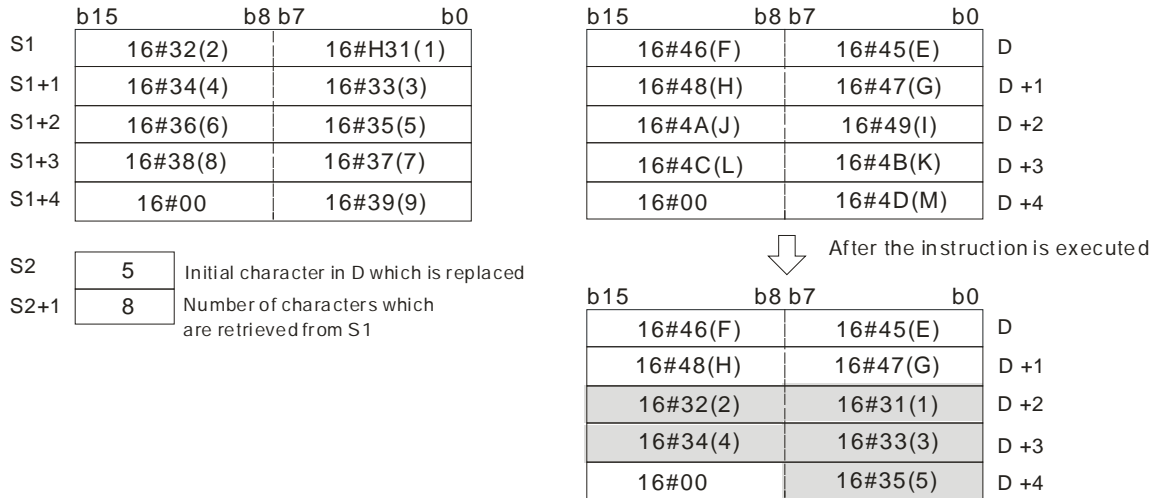
Explanation:

- S₂: The initial character in D which is replaced
 S₂+1: The number of characters which are retrieved from S₁
- The retrieve of the characters in the string in S₁ begins from the first character, and the value in S₂+1 indicates the number of characters which are retrieved from the string in S₁. The characters which are retrieved from the string in S₁ replace the characters in D starting from the character indicated by the value in S₂.

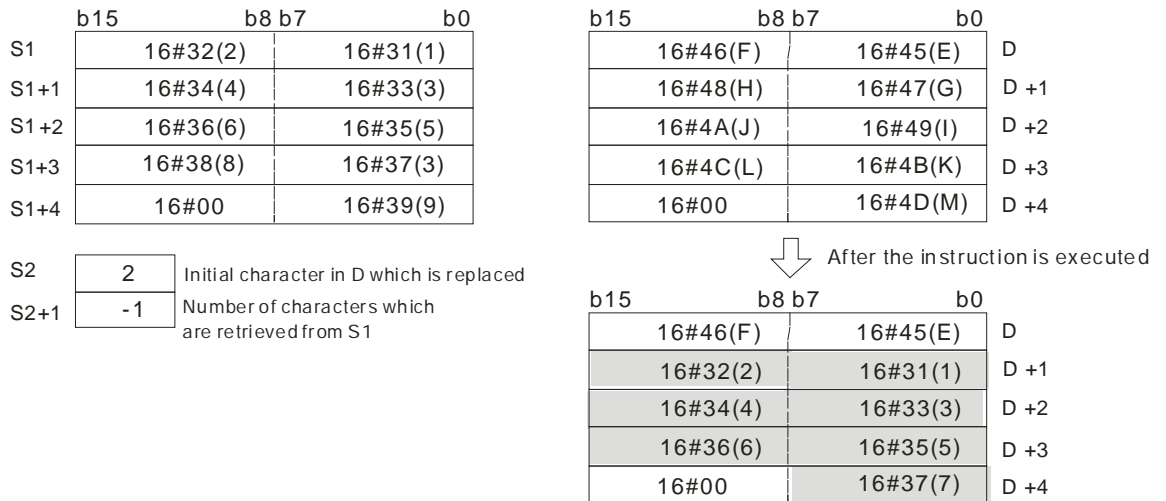


- If the value in S₂+1 is 0, the instruction is not executed.

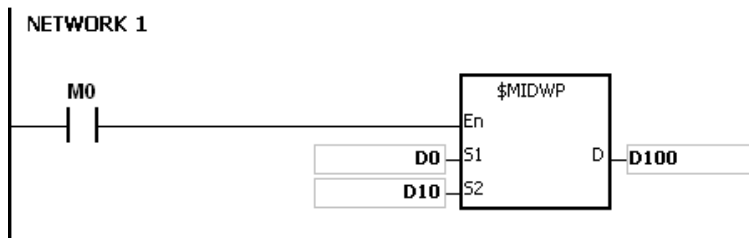
4. If the value in S_2+1 is larger than the length of the string in D , the characters in D which are replaced start from the character indicated by the value in S_2 to the last character in D .



5. If the value in S_2+1 is -1, all characters in S_1 are retrieved.



Example:



	b15	b8 b7	b0
D0	16#42(B)	16#41(A)	
D1	16#44(D)	16#43(C)	
D2	16#46(F)	16#45(E)	
D3	16#00		

	b15	b8 b7	b0
D100	16#32(2)	16#31(1)	
D101	16#34(4)	16#33(3)	
D102	16#36(6)	16#35(5)	
D103	16#38(8)	16#37(7)	
D104	16#00		

D10	3	Initial character in D which is replaced Number of characters which are retrieved from S1
D11	4	

↓ After the instruction is executed

	b15	b8 b7	b0
D100	16#32(2)	16#31(1)	
D101	16#42(B)	16#41(A)	
D102	16#44(D)	16#43(C)	
D103	16#38(8)	16#37(7)	
D104	16#00		

Additional remark:

1. If the string in **S₁** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in D does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the value in S2 is less than or equal to 0, or if the value in S2 is larger than the length of the string in D, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the value in S2+1 is less than -1, or if the value in S2+1 is larger than the number of characters which can be retrieved from the string in S1, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the operand S2 used during the execution of the 16-bit instruction is declared in ISPSOft, the data type will be ARRAY [2] of WORD/INT.

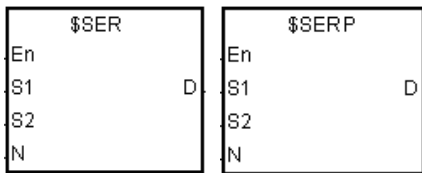
FB/FC	Instruction			Operand	Description
FC	\$SER	P		S ₁ , S ₂ , n, D	Searching the string

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁														●
S ₂														●
n		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●		●		●			○	
S ₂	●	●			●	●		●	●		●		●			○	
n	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●		●	○	●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

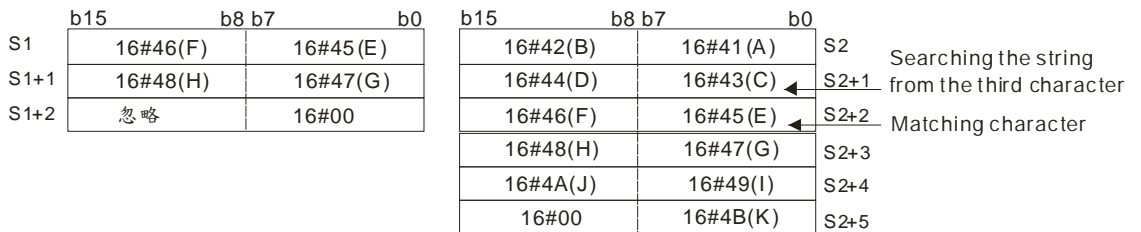
Graphic expression:



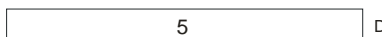
- S₁ : String which is searched
- S₂ : String which is searched for
- n : nth character in S₂ from which the search begins
- D : Search result

Explanation:

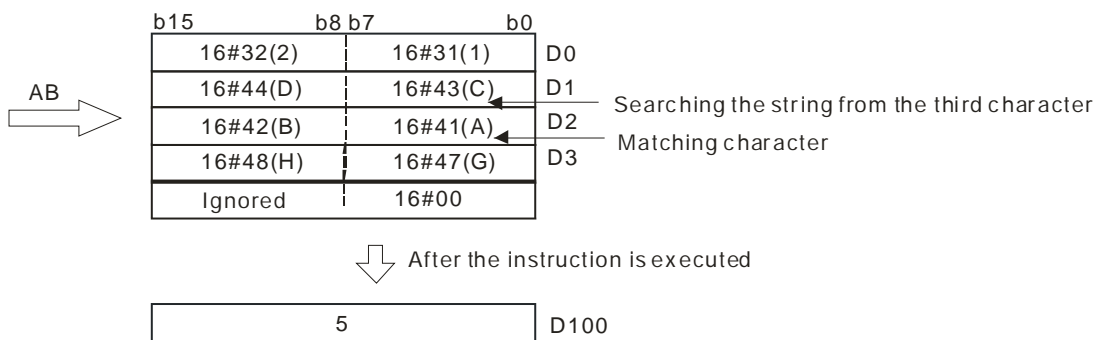
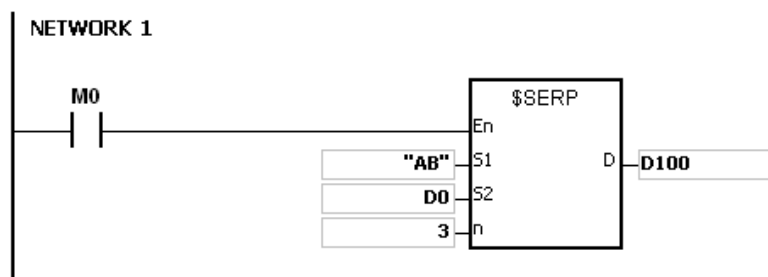
- The instruction is used to search the string from the nth character in S₂ for the string which is the same as the string in S₁, and the search result is stored in D.
- Suppose the string in S₂ is "ABCDEFGHJK", the string in S₁ is "EFGH", and n is 3. The search begins from the third character in S₂, and the value in D is 5.



After the instruction is executed



Example:



Additional remark:

1. If the string in **S₁** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in **S₂** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#2003.
3. If **n** is less than or equal to 0, or if **n** is larger than the length of the string in **S₂**, SM0 is ON, and the error code in SR0 is 16#2003

FB/FC	Instruction			Operand	Description
FC	\$RPLC	P		S ₁ , S ₂ , S ₃ , S ₄ , D	Replacing the characters in the string

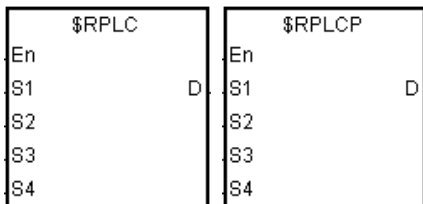
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂														●
S ₃		●					●							
S ₄		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●	●	●	●		●				
S ₂	●	●			●	●		●	●	●	●		●			○	
S ₃	●	●			●	●		●	●	●	●	○	●	○	○		
S ₄	●	●			●	●		●	●	●	●	○	●	○	○		
D	●	●			●	●		●	●	●			●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

S₁ : String which is replaced



S₂ : New string

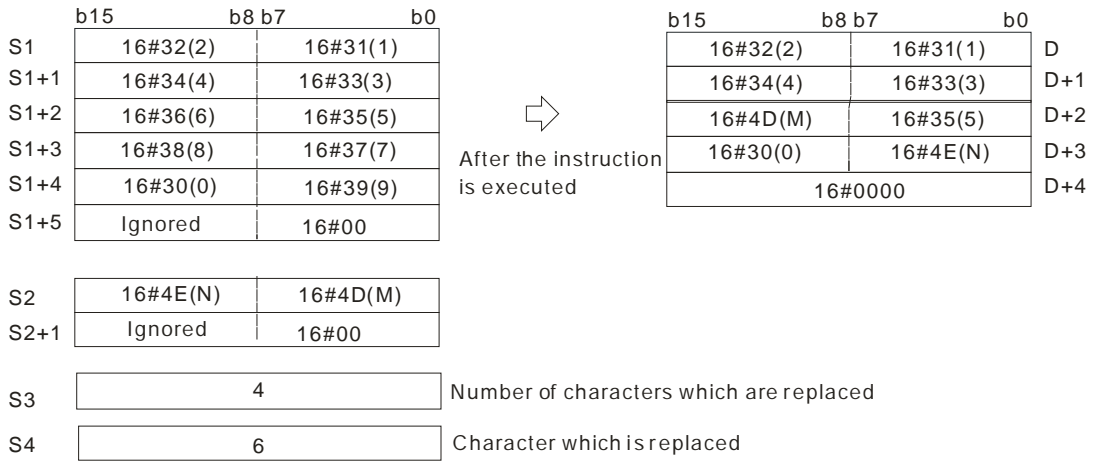
S₃ : Number of characters in S₁ which are replaced

S₄ : The characters in S₁ starting from the character indicated by the value in S₄ are replaced.

D : Device in which the execution result is stored

Explanation:

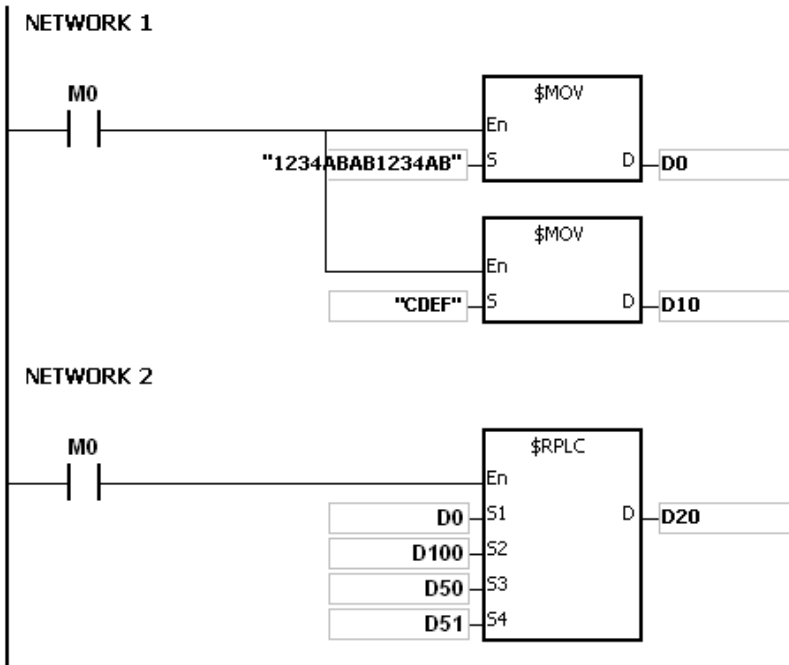
- The characters in S₁ starting from the character indicated by the value in S₄ are replaced by the characters in S₂, the number of characters which are replaced is indicated by the value in S₃, and the result is stored in D.
- The four characters starting from the sixth character in the string “1234567890” are replaced by “MN”, and the result is “12345MN0”.




3. If the string in S2 is 16#00, the instruction has the function of deleting the characters.
4. If the value in S3 is larger than the number of characters which can be replaced in the string in S1, the characters in S1 starting from the character indicated by the value in S4 to the last character in S1 are replaced.
5. If the value in S₃ 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 instruction \$RPLC is executed, the characters in D0~D7 starting from the character indicated by the value in D51 are replaced by the characters in D10~D11. The number of characters which are replaced is indicated by the value in D50, and the result is stored in D20~D27.



If the values in D50 and D51 are 3 and 4 respectively, the execution result is as follows.


	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		 After the instruction is executed	16#32(2)	16#31(1)		D20
D1	16#34(4)	16#33(3)			16#43(C)	16#33(3)		D21
D2	16#42(B)	16#41(A)			16#45(E)	16#44(D)		D22
D3	16#42(B)	16#41(A)			16#42(B)	16#46(F)		D23
D4	16#32(2)	16#31(1)			16#32(2)	16#31(1)		D24
D5	16#34(4)	16#33(3)			16#34(4)	16#33(3)		D25
D6	16#42(B)	16#41(A)			16#42(B)	16#41(A)		D26
D7	Ignored	16#00			16#0000			D27

D10	16#44(D)	16#43(C)
D11	16#45(F)	16#45(E)
D12	Ignored	16#00

D50 Number of characters which are replaced

D51 Character which is replaced

If the values in D50 and D51 are 4 and 4 respectively, the execution result is as follows.


	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		 After the instruction is executed	16#32(2)	16#31(1)		D20
D1	16#34(4)	16#33(3)			16#43(C)	16#33(3)		D21
D2	16#42(B)	16#41(A)			16#45(E)	16#44(D)		D22
D3	16#42(B)	16#41(A)			16#42(B)	16#46(F)		D23
D4	16#32(2)	16#31(1)			16#32(2)	16#31(1)		D24
D5	16#34(4)	16#33(3)			16#34(4)	16#33(3)		D25
D6	16#42(B)	16#41(A)			16#42(B)	16#41(A)		D26
D7	Ignored	16#00			16#0000			D27

D10	16#44(D)	16#43(C)
D11	16#45(F)	16#45(E)
D12	Ignored	16#00

D50 Number of characters which are replaced

D51 Character which is replaced


If the values in D50 and D51 are 20 and 4 respectively, the execution result is as follows.

	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		 After the instruction is executed	16#32(2)	16#31(1)		D20
D1	16#34(4)	16#33(3)			16#43(C)	16#33(3)		D21
D2	16#42(B)	16#41(A)			16#45(E)	16#44(D)		D22
D3	16#42(B)	16#41(A)			16#00	16#46(F)		D23
D4	16#32(2)	16#31(1)						
D5	16#34(4)	16#33(3)						
D6	16#42(B)	16#41(A)						
D7	Ignored	16#00						

D10	16#44(D)	16#43(C)
D11	16#45(F)	16#45(E)
D12	Ignored	16#00

D50	20	Number of characters which are replaced
D51	4	Character which is replaced

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.

	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		 After the instruction is executed	16#32(2)	16#31(1)		D20
D1	16#34(4)	16#33(3)			16#41(A)	16#33(3)		D21
D2	16#42(B)	16#41(A)			16#31(1)	16#42(B)		D22
D3	16#42(B)	16#41(A)			16#33(3)	16#32(2)		D23
D4	16#32(2)	16#31(1)			16#41(A)	16#34(4)		D24
D5	16#34(4)	16#33(3)			16#00	16#42(B)		D25
D6	16#42(B)	16#41(A)						
D7	16#00							

D10	16#00
-----	-------

D50	3	Number of characters which are replaced
D51	4	Character which is replaced

Additional remark:

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 **S**₃<0, **S**₄<=0, or the length designated by **S**₄ exceeds **S**₁, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction		Operand				Description				
FC	\$DEL	P	S ₁ , S ₂ , S ₃ , D				Deleting the characters in the string				

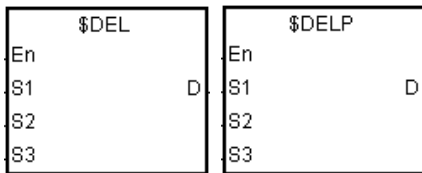
Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁														●
S ₂		●					●							
S ₃		●					●							
D														●

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S ₁	●	●			●	●		●	●		●		●				
S ₂	●	●			●	●		●	●		●	○	●	○	○		
S ₃	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

S₁ : String



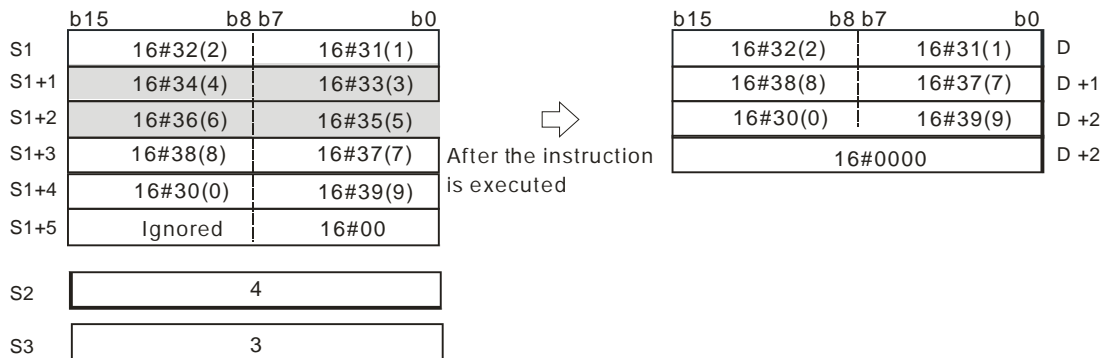
S₂ : Number of characters which are deleted

S₃ : The characters in S₁ starting from the character indicated by the value in S₃ are deleted.

D : Device in which the execution result is stored

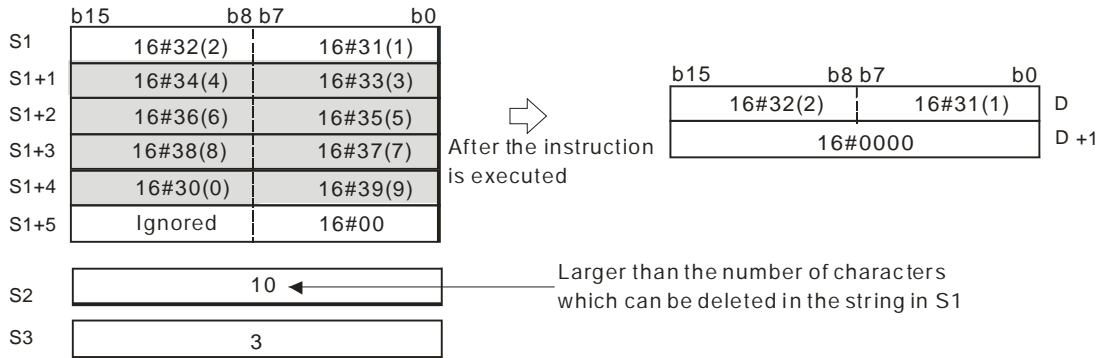
Explanation:

- The characters in S₁ starting from the character indicated by the value in S₃ are deleted, the number of characters which are deleted is indicated by the value in S₂, and the result is stored in D.
- The four characters starting from the third character in the string “1234567890” in S₁ are deleted, and the result “127890” is stored in D.



- If the value in S₂ is larger than the number of characters which can be deleted in the string in S₁, the

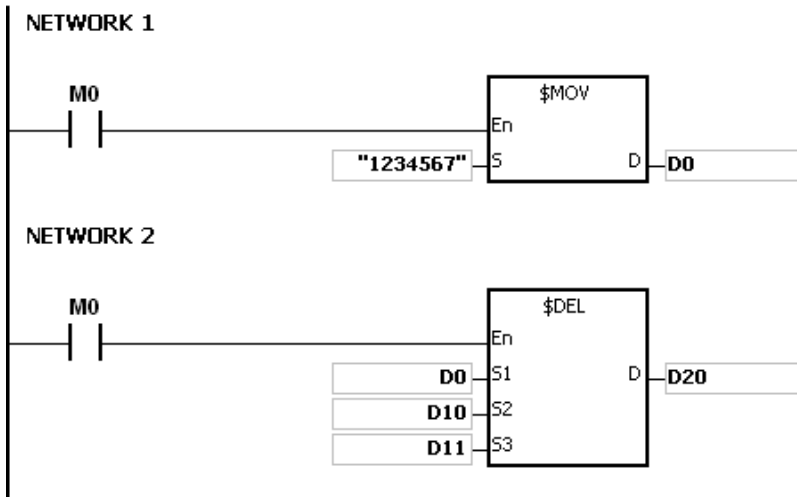
characters in **S₁** starting from the character indicated by the value in **S₃** to the last character in **S₁** are deleted, and 16#00 is stored in **D**.



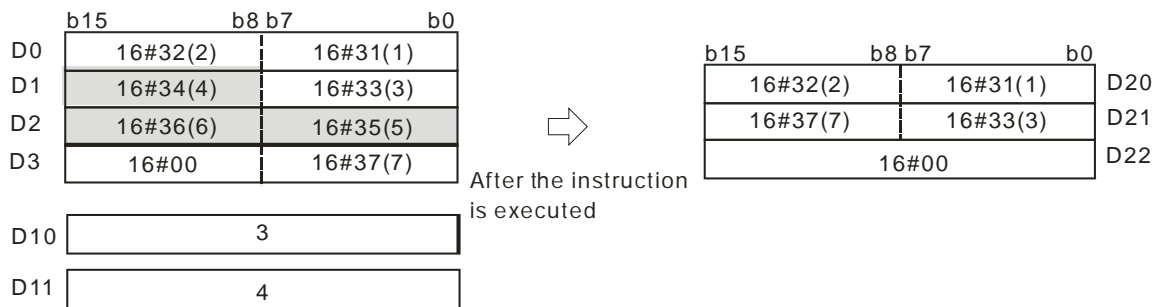
- If the value in **S₂** is equal to 0, the instruction is not executed.

Example:

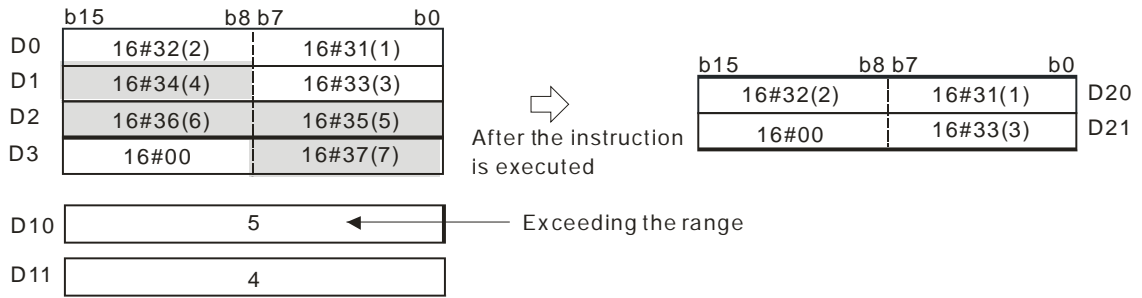
When M0 is ON, the data in D0~D3 is "1234567". When the instruction \$DEL is executed, the characters in D0~D3 starting from the character indicated by the value in D11 are deleted. The number of characters which are deleted is indicated by the value in D10, and the result is stored in D20~D22.



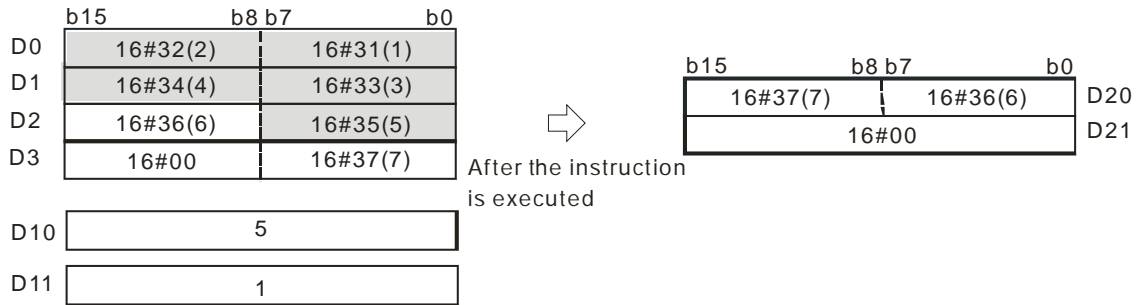
If the values in D10 and D11 are 3 and 4 respectively, the execution result is as follows.



If the values in D10 and D11 are 5 and 4 respectively, the execution result is as follows. Owing to the fact that the number of characters which are deleted exceeds the range, the characters in D0~D3 starting from the fourth character to the last character are deleted.



If the values in D10 and D11 are 5 and 1 respectively, the execution result is as follows.



Additional remark:

1. If the string in **S₁** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the value in **S₂** is less than 0, the value in **S₃** is less than or equal to 0, or the value in **S₃** is larger than the length of the string in **S₁**, SM0 is ON, and the error code in SR0 is 16#2003.

FB/FC	Instruction		Operand				Description			
FC	\$CLR	P	S				Clearing the string			

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

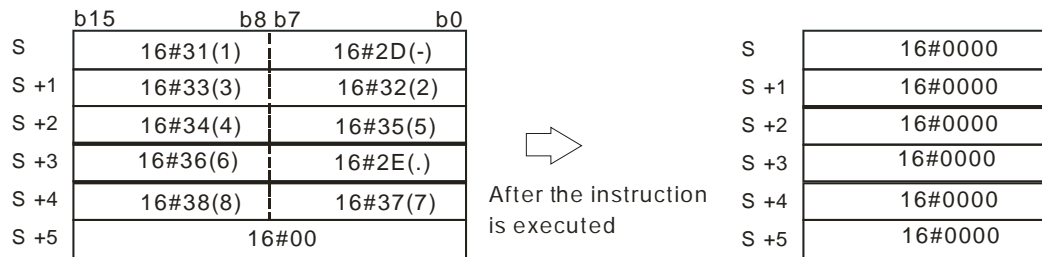
Graphic expression:



S : String which is cleared

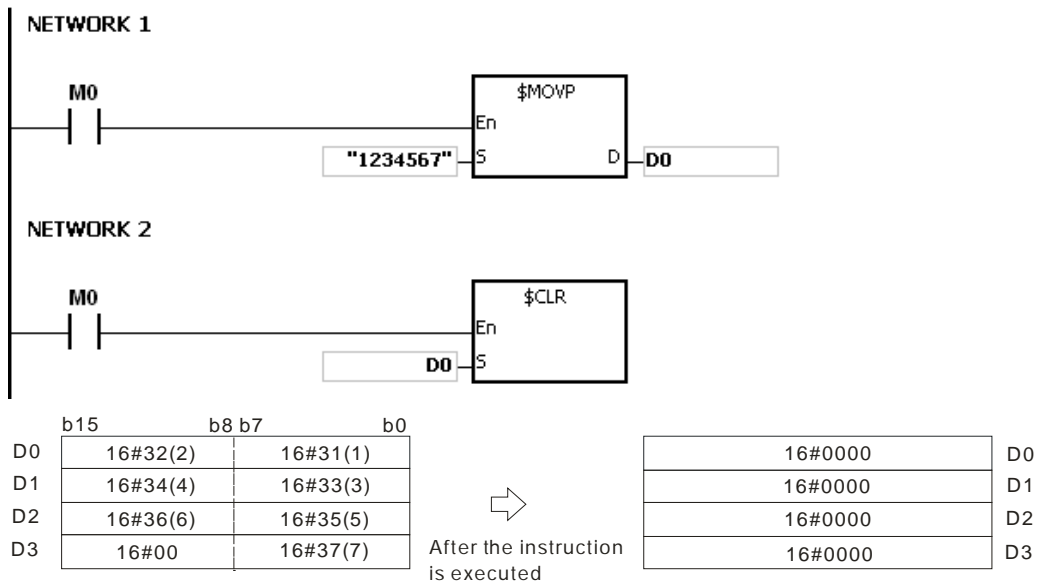
Explanation:

The string in **S** is cleared.



Example:

The string in D0 is cleared, as illustrated below.



Additional remark:

If the string in **S** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.

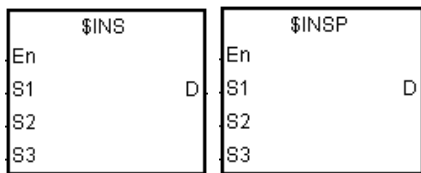
FB/FC	Instruction		Operand	Description
FC	\$INS	P	S ₁ , S ₂ , S ₃ , D	Inserting the string

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂														●
S ₃		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●		●		●				
S ₂	●	●			●	●		●	●		●		●			○	
S ₃	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : String

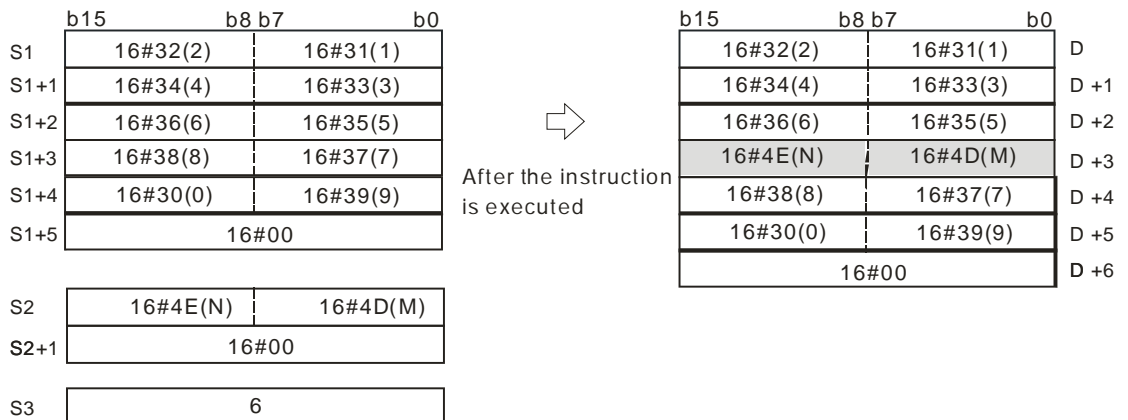
S₂ : String which is inserted

S₃ : The string is inserted into S₁ after the character indicated by the value in S₃.

D : Device in which the execution result is stored

Explanation:

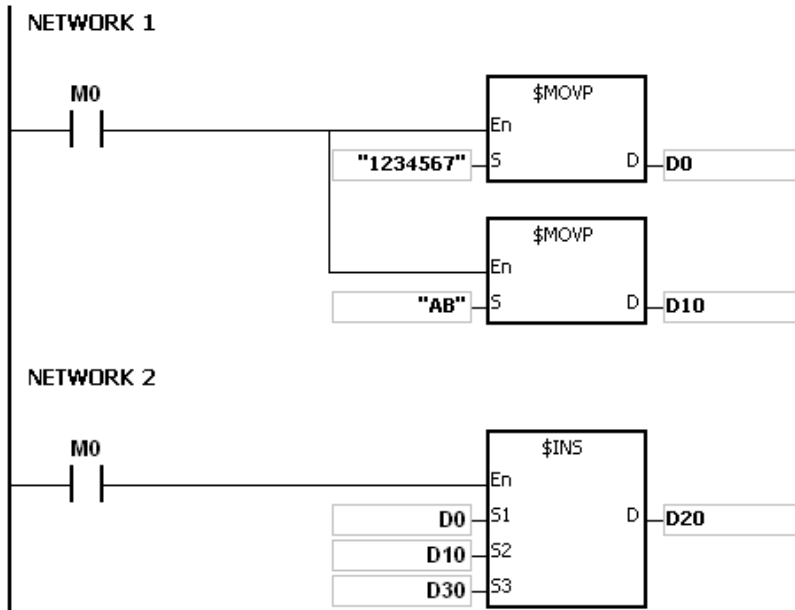
- The string in S₂ is inserted into the string in S₁ after the character indicated by the value in S₃, and the result is stored in D.
- If the string in either S₁ or S₂ is a null string, the other string which is not a null string is stored in D.
- If the strings in S₁ and S₂ are null strings, 16#0000 is stored in D.



Example:

When M0 is ON, the data in D0-D3 is "1234567", and the data in D10 is "AB". When the instruction \$INS is

executed, "AB" is inserted into the string in D0~D3 after the character indicated by the value in D30. The result is stored in D20~D24.



If the value in D30 is 1, the execution result is as follows.

	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		⇒ After the instruction is executed	16#41(A)	16#31(1)		D20
D1	16#34(4)	16#33(3)			16#32(2)	16#42(B)		D21
D2	16#36(6)	16#35(5)			16#34(4)	16#33(3)		D22
D3	16#00	16#37(7)			16#36(6)	16#35(5)		D23
					16#00	16#37(7)		D24
D10	16#42(B)	16#41(A)						
D11	Ignored	16#00						
D30	1							

If the value in D30 is 0, the execution result is as follows.

	b15	b8 b7	b0		b15	b8 b7	b0	
D0	16#32(2)	16#31(1)		⇒ After the instruction is executed	16#42(B)	16#41(A)		D20
D1	16#34(4)	16#33(3)			16#32(2)	16#31(1)		D21
D2	16#36(6)	16#35(5)			16#34(4)	16#33(3)		D22
D3	16#00	16#37(7)			16#36(6)	16#35(5)		D23
					16#00	16#37(7)		D24
D10	16#42(B)	16#41(A)						
D11	Ignored	16#00						
W0	0							

Additional remark:

1. If the string in **S₁** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
2. If the string in **S₂** does not end with 16#00, SM0 is ON, and the error code in SR0 is 16#200E.
3. If the value in **S₃** is less than 0, or if the value in **S₃** is larger than the length of the string in **S₁**, SM0 is ON, and the error code in SR0 is 16#2003.

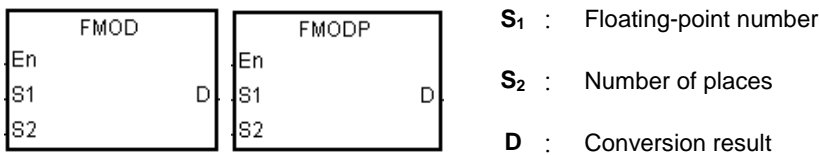
FB/FC	Instruction		Operand	Description
FC	FMOD	P	S₁, S₂, D	Converting the floating-point number into the binary-coded decimal floating-point number

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁			●					●						
S₂		●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁	●	●			●	●		●	●		●		●				○
S₂	●	●			●	●		●	●		●	○	●	○	○		
D	●	●			●	●		●	●				●				

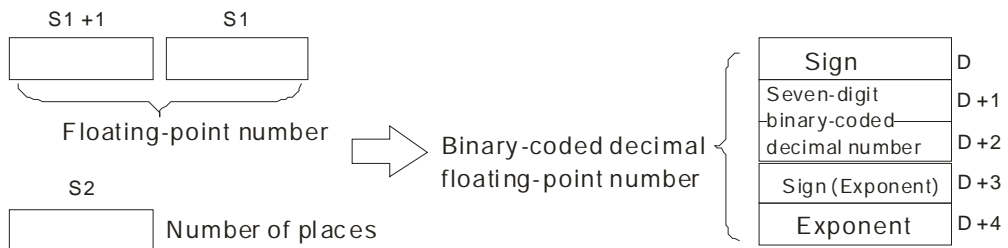
Pulse instruction	32-bit instruction	64-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

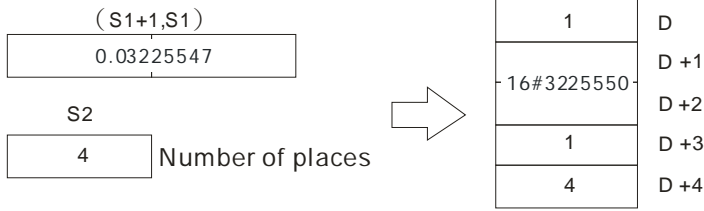


Explanation:

- The decimal point in the floating-point number in **S₁** is moved to the right in accordance with the setting of **S₂** first, and then the result is converted into the binary-coded decimal floating-point number. The final conversion result is stored in **D**.

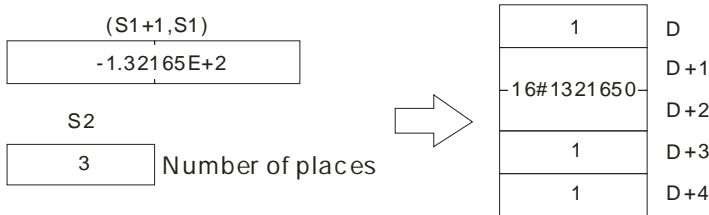


- The binary-coded decimal floating-point number format:
 - S₂**: The number of places. The value in **S₂** should be within the range between 0 and 7.
 - D**: If the floating-point number in **S₁** is a positive number, the value in **D** is 0. If the floating-point number in **S₁** is a negative number, the value in **D** is 1.
 - (D+2, D+1)**: The seven-digit binary-coded decimal number converted from the floating-point number
 - D+3**: If the exponent is a positive number, the value in **D+3** is 0. If the exponent is a negative number, the value in **D+3** is 1.
 - D+4**: The exponent. If the floating-point number in **S₁** is -0.03225547 and the value in **S₂** is 4, the conversion result is as follows.



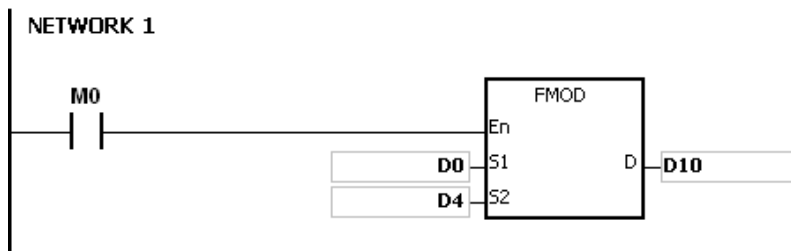
- Since the value in **S₂** is 4, the decimal point in the floating-point number in **S₁** is moved to the right by four decimal places. The floating-point number in **S₁** becomes -322.5547.
-322.5547 is equal to -3225547E-4. The binary-coded decimal floating-point number format is as follows.
- The value in **D** is 1 because the floating-point number in **S₁** is a negative number.
- The value stored in (**D+2, D+1**) is 16#3225550. (The floating-point number is converted into the seven-digit binary-coded decimal number, and the seven-digit binary-coded decimal number is rounded off).
- The value in **D+3** is 1 because the exponent is a negative number.
- The value in **D+4** is 4.

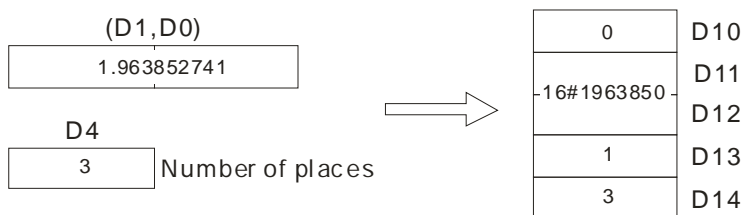
If the floating-point number in **S₁** is -1.32165E+2 and the value in **S₂** is 3, the conversion result is as follows.



- -1.32165E+2 is equal to 132.165. Since the value in **S₂** is 3, the decimal point in the floating-point number in **S₁** is moved to the right by three decimal places. The floating-point number in **S₁** becomes -132165.
-132165 is equal to -1321650E-1. The binary-coded decimal floating-point format is as follows.
- The value in **D** is 1 because the floating-point number in **S₁** is a negative number.
- The value stored in (**D+2, D+1**) is 16#1321650. (The floating-point number is converted into the seven-digit binary-coded decimal number, and the seven-digit binary-coded decimal number is rounded off).
- The value in **D+3** is 1 because the exponent is a negative number.
- The value in **D+4** is 4.

Example:





- Since the value in D4 is 3, the decimal point in 1.963852741 in (D1, D0) is moved to the right by three decimal places. The floating-point number in (D1, D0) becomes 1963.852741.
- The value in D10 is 0 because the floating-point number in S1 is a positive number.
1963.852741 is equal to 1963852E-3. The binary-coded decimal floating-point number format is as follows.
- The value stored in (D12, D11) is 16#1963850. (The floating-point number is converted into the seven-digit binary-coded decimal number, and the seven-digit binary-coded decimal number is rounded off).
- The value in D13 is 1 because the exponent is a negative number.
- The value in D14 is 3.

Additional remark:

1. If the value in S1 exceeds the range of values which 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.
2. If the value in S2 exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the operand D used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of WORD/INT.

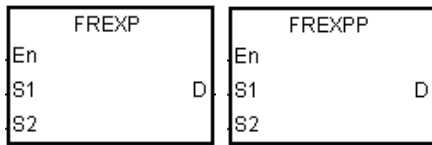
FB/FC	Instruction		Operand				Description			
FC	FREXP	P	S₁, S₂, D				Converting the Binary-coded decimal floating-point number into the floating-point number			

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁		●					●							
S₂		●					●							
D			●					●						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁	●	●			●	●		●			●		●				
S₂	●	●			●	●		●			●	○	●	○	○		
D	●	●			●	●		●			●		●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

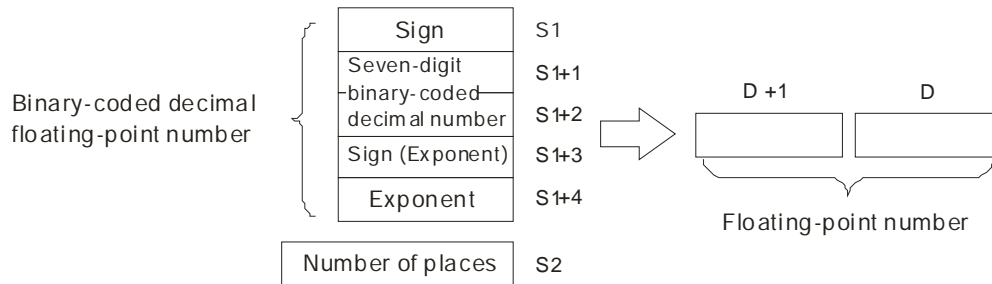
Graphic expression:



- S₁** : Binary-coded decimal floating-point number
- S₂** : Number of places
- D** : Conversion result

Explanation:

The binary-coded decimal floating-point number in **S₁** is converted into the floating-point number first, and then the decimal point in the floating-point number is moved to the left in accordance with the setting of **S₂**. The final result is stored in **D**.



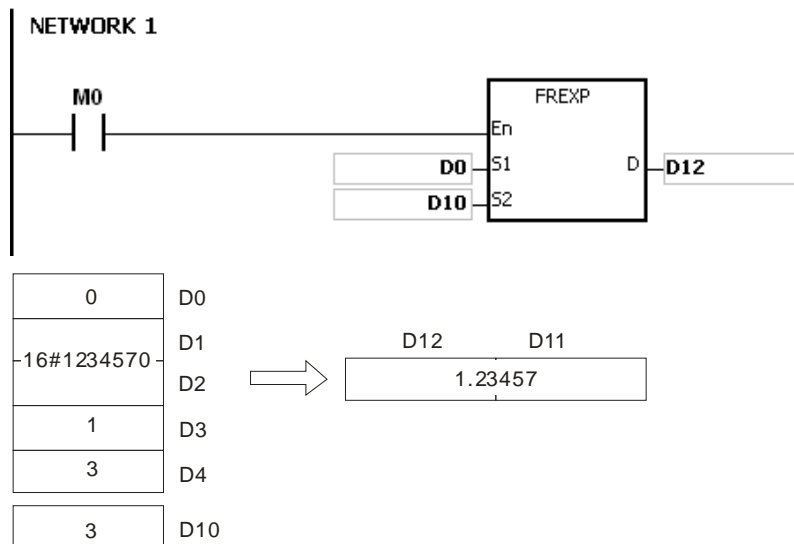
The binary-coded decimal floating-point number format:

1. If the binary-coded decimal floating-point number is a positive number, the value in **S₁** is 0. If the binary-coded decimal floating-point number is a negative number, the value in **S₁** is 1.
2. The seven-digit binary-coded decimal number is stored in (**S₁+2, S₁+1**).
3. If the exponent is positive, the value in **S₁+3** is 0. If the exponent is negative, the value in **S₁+3** is 1.
4. **S₁+4**: The exponent
5. The value in **S₁+4** should be within the range between 0 and 38.
6. **S₂**: The number of places

7. The value in S_2 should be within the range between 0 and 7.

Example:

When the conditional contact MO is ON, the binary-coded decimal floating-point number is converted into the floating-point number.



- The value in D0 is 0 because the binary-coded decimal floating-point number is a positive number.
- 16#1234570 is stored in (D2, D1).
- The value in D3 is 1 because the exponent is a negative number.
- The value in D4 is 3.
- 1234570E-3, the binary-coded decimal floating-point number in D0~D4, is converted into the 1234.57.
- Since the value in D10 is 3, the decimal point in 1234.57 in is moved to the left by three places. The result is 1.23457, and is stored in (D12, D11).

Additional remark:

1. If the value in S_1 is neither 0 nor 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If the number of digits in (S_1+2 , S_1+1) is larger than 7, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the value in (S_1+2 , S_1+1) is not a binary-coded decimal value (The value is represented by the hexadecimal number, but one of digits is not within the range between 0 and 9.), the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200D.
4. If the value in S_1+3 is neither 0 nor 1, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the value in S_1+4 is less than 0 or larger than 38, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. If the value in S_2 is less than 0 or larger than 7, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
7. If the operand S_1 used during the execution of the 32-bit instruction is declared in ISPSOft, the data type will be ARRAY [5] of WORD/INTT.

3.24 Ethernet Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>EMDRW</u>	–	✓	Reading/Writing the Modbus TCP data	11
FC	–	<u>DINTOA</u>	✓	Converting the IP address of the integer type into the IP address of the string type	5
FC	–	<u>DIATON</u>	✓	Converting the IP address of the string type into the IP address of the integer type	5-11

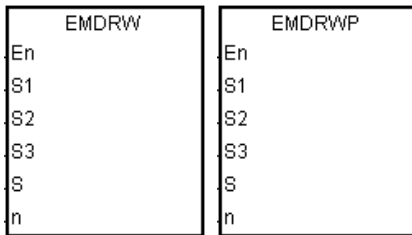
FB/FC	Instruction		Operand		Description
FC	EMDRW	P	S₁, S₂, S₃, S, n		Reading/Writing the Modbus TCP data

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁		●					●							
S ₂ , S ₃		●					●							
S	●	●					●							
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S ₁	●	●			●	●		●	●			○	●				
S ₂ , S ₃	●	●			●	●		●	●			○	●	○	○		
S	●	●	●		●	●		●	●			○	●				
D	●	●			●	●		●	●			○	●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



- S₁ : Unit address
- S₂ : Function code
- S₃ : Device address
- S : Register involved in the reading/writing of the data
- n : Data length

Explanation:

1. Before using the instruction, users have to accomplish the following setting in ISPSOft.
 PLC Parameter Setting→Ethernet-Basic→Setting the IP address and the netmask address
2. Setting S₁ :

Operand	Description	Setting range
S ₁	Station address	The station address should be within the range between 0 and 255.
S ₁ +1	Remote IP address (high word)	Example: The remote IP address is 172.16.144.230. S ₁ +1=16#AC10 S ₁ +2=16#90E6
S ₁ +2	Remote IP address (low word)	
S ₁ +3	Whether to close the connection	0: The connection is closed after the execution of the instruction is complete. 1: The connection is persistent. (The closing of the connection depends on the setting of the TCP keepalive timer.)

3. S₂: Function code

For example:

1 (16#01): The AH Motion CPU reads the data from several bit devices which are not discrete input devices.

2 (16#02): The AH Motion CPU reads the data from several bit devices which are discrete input devices.

3 (16#03): The AH Motion CPU reads the data from several word devices which are not input registers.

4 (16#04): The AH Motion CPU reads the data from several word devices which are input registers.

5 (16#05): The AH Motion CPU writes the state into a bit device.

6 (16#06): The AH Motion CPU writes the data into a word device.

15 (16#0F): The AH Motion CPU writes the states into several bit devices.

16 (16#10): The AH Motion CPU writes the data into several word devices.

Only the function codes mentioned above are supported, and other function codes can not be executed.

Please refer to the examples below.

- 4. **S₃**: The device address
- 5. **S**: The register involved in the reading/writing of the data

The data which will be written into the external equipment is stored in the register in advance.

The data which is read from the external equipment is stored in the register.

- 6. **n**: The length of the data

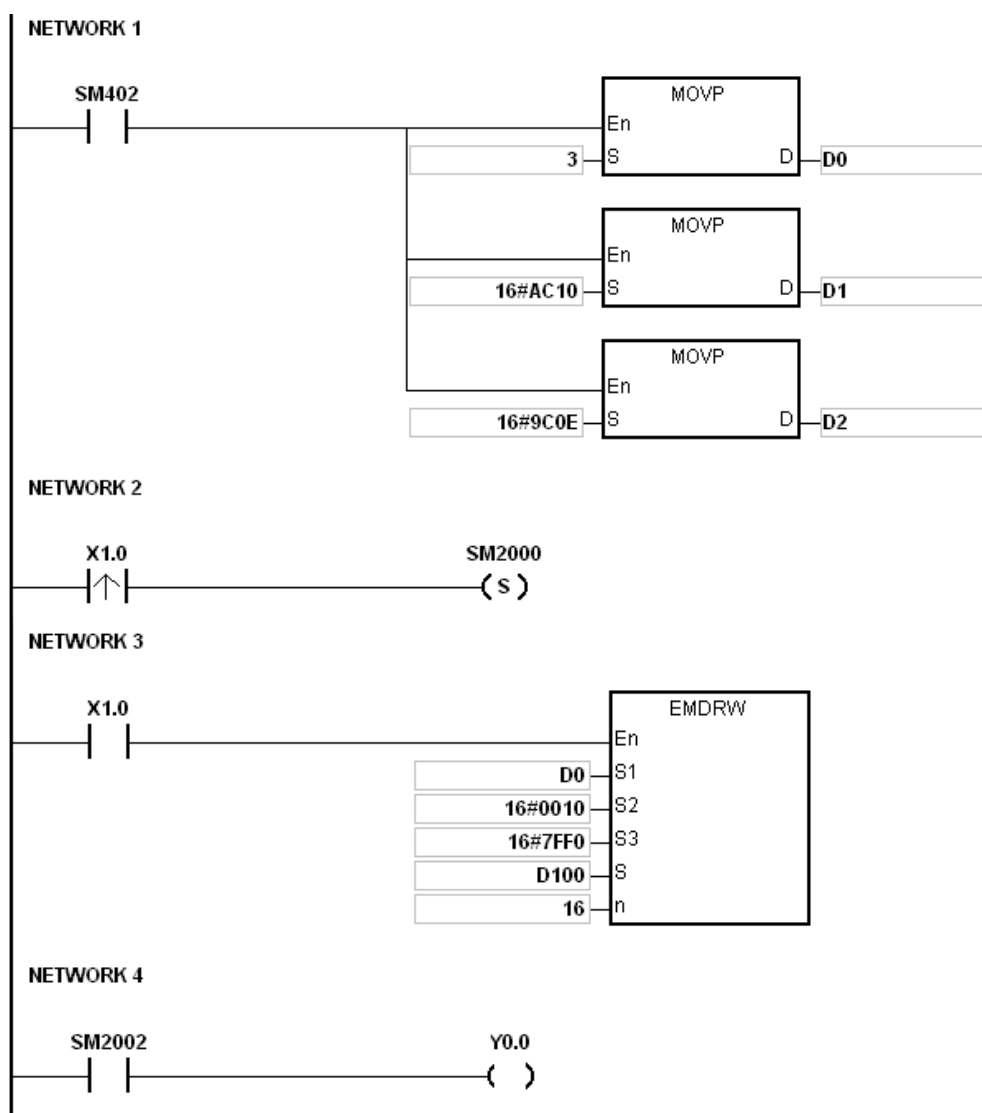
The size of the data can not be larger than 240 bytes. For the communication commands related to the coils, the unit of the data is the bit, and n should be within the range between 1 and 1920. For the communication commands related to the registers, the unit of the data is the word, and n should be within the range between 1 and 120.

Flag EMDRW	Sending the data	Waiting for the data	Having received the data	Error flag	Timeout flag	Having closed the connection
1	SM2000	SM2001	SM2002	SM2003	SM2004	SM2005
2	SM2006	SM2007	SM2008	SM2009	SM2010	SM2011
3	SM2012	SM2013	SM2014	SM2015	SM2016	SM2017
4	SM2018	SM2019	SM2020	SM2021	SM2022	SM2023
5	SM2024	SM2025	SM2026	SM2027	SM2028	SM2029
6	SM2030	SM2031	SM2032	SM2033	SM2034	SM2035
7	SM2036	SM2037	SM2038	SM2039	SM2040	SM2041
8	SM2042	SM2043	SM2044	SM2045	SM2046	SM2047

- 7. The instruction can be used several times in the program, but only eight instructions are executed at a time.
- 8. If several flags which are related to the sending of the data are ON simultaneously, the data indicated by the flag whose number is the smallest is sent first.
- 9. Generally, the pulse instruction EMDRWP is used.

Example:

- 1. The remote station address is set to 3.

**Additional remark:**

1. If the value in S1, S2, or S exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
2. If n exceeds the range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#200B.
3. If the device specified by S1+3 exceeds the device range, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. 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.
5. If the function code specified by S2 is related to the bit device, the device specified by S has to be the bit device. Otherwise, the operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.
6. If the function code specified by S2 is related to the word device, the device specified by S has to be the word device. Otherwise, the operation error occurs, the instruction is not executed, and the error code in SR0 is 16#2003.
7. If the communication command is 0x05 or 0x06, n does not work. The state or the data is written into one bit device or one word device.

8. If a flag related to the sending of the data is ON, and the corresponding flag related to the connection's having been closed is not ON, the system will search for the flags which both are ON to execute the instruction. If there are no flags which both are ON, the instruction is not executed.
9. During the connection, if the value in the high word of the remote IP address is 0, 127, or the value larger than 225, the error occurs, and the corresponding error flag is ON, and the error code is 16#6403.
10. During the connection, if the remote machine is disconnected, the error code is 16#6401. If there is a connection timeout, the error code is 16#6402.
11. If the remote machine is disconnected when the command is sent, the error code is 16#6401. If there is an ACK timeout, the error code is 16#6402.
12. If the function code is incorrect when the command is received, the error code is 16#6404. If the length of the data is incorrect, the error code is 16#6405. If there is a response timeout, the error code is 16#6402.
13. If the remote machine is disconnected when the connection is closed, the error code is 16#6401. If there is a connection timeout, the error code is 16#6402.
14. If users declare the operand **S₁** in ISPSOft, the data type will be ARRAY [4] of WORD/INT.
15. If the instruction is executed while the Ethernet cable is not connected, SM1100 will be ON with the error code 16#600D.

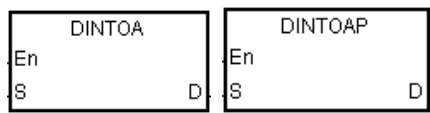
FB/FC	Instruction			Operand	Description
FC	D*	INTOA	P	S, D	Converting the IP address of the integer type into the IP address of the string type

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●	○	○		
D	●	●			●	●		●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	-	AH Motion CPU

Graphic expression:



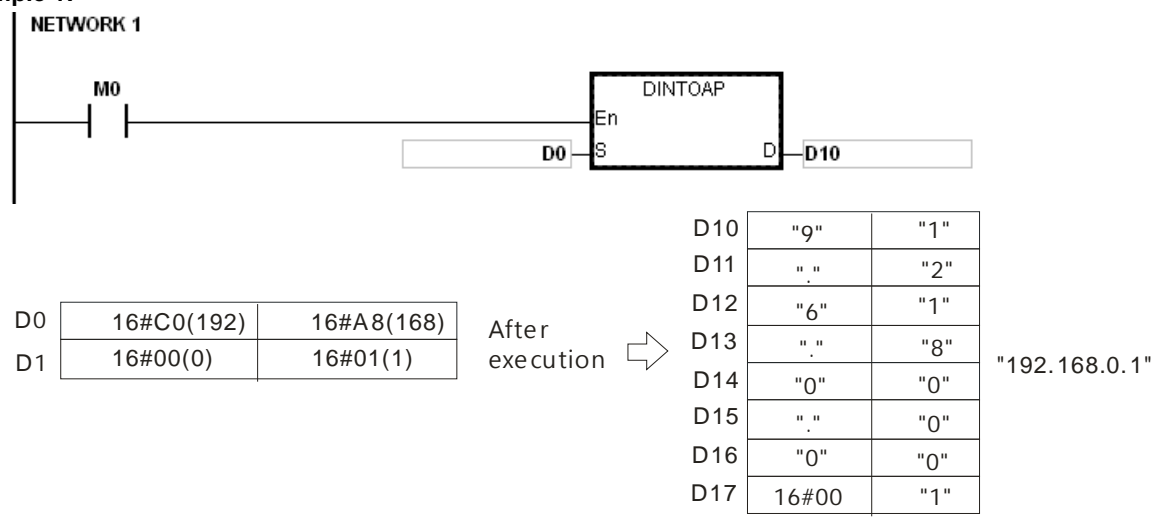
S : Source value

D : Conversion result

Explanation:

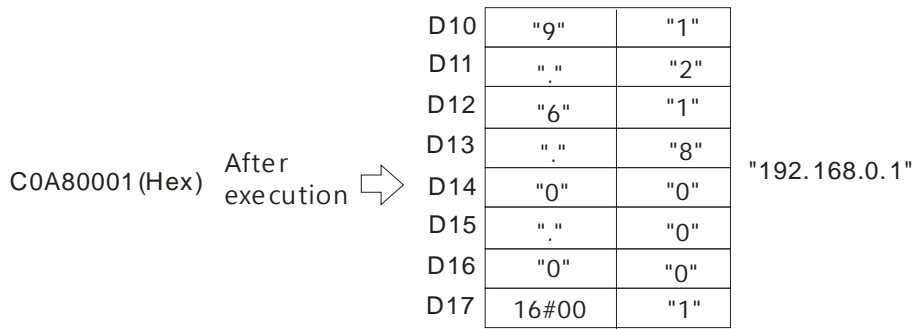
- The IP address of the integer type in S is converted into the IP address of the string type, and the conversion result is stored in D.
- The operand D occupies eight devices.

Example 1:



Example 2:





Additional remark:

If users declare the operand D in ISPSOft, the data type will be ARRAY [8] of WORD/INT.

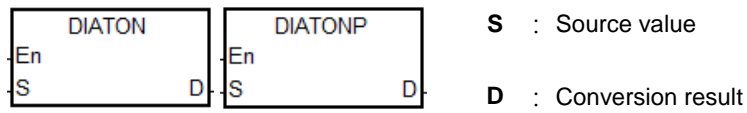
FB/FC	Instruction			Operand	Description
FC	D*	IATON	P	S, D	Converting the IP address of the string type into the IP address of the integer type

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S ₁														●
D			●*					●*						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●			●	●		●	●		●		●			○	
D	●	●			●	●		●	●				●				

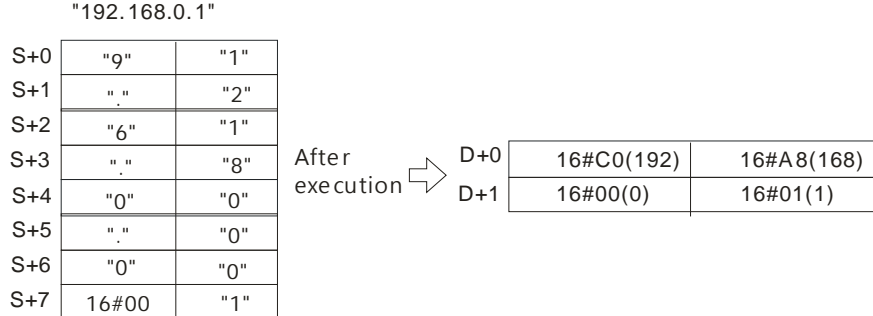
Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	-	AH Motion CPU

Graphic expression:



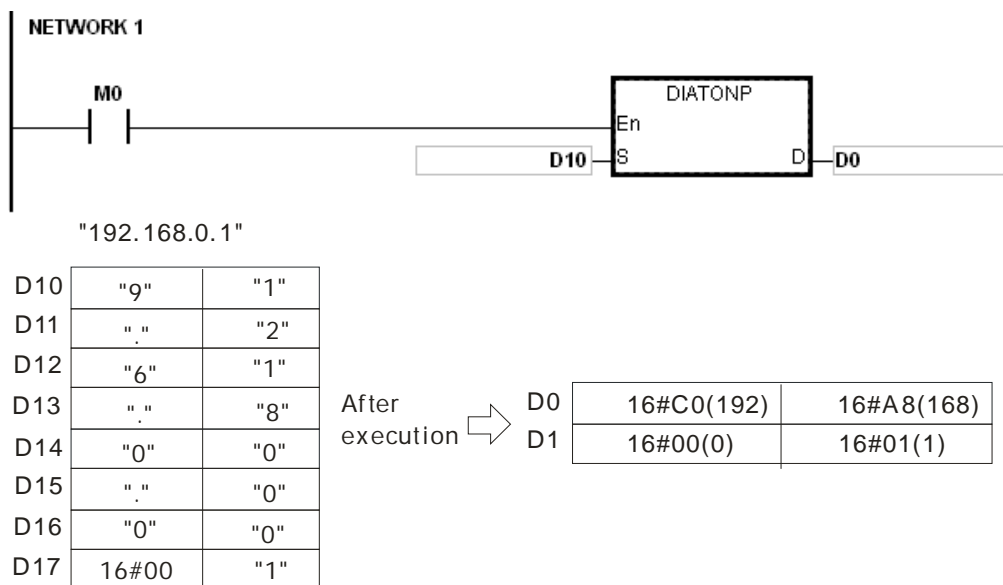
Explanation:

- The IP address of the string type in **S** is converted into the IP address of the integer type, and the conversion result is stored in D.
- The operand **S** occupies eight devices.

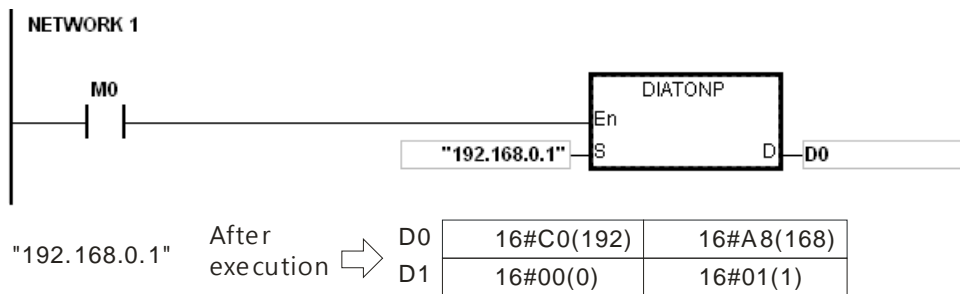


- The IP address of the string type in **S** is divided into four sections. These sections are separated by "." (16#2E), and there are three characters in every section.
- The value converted from the characters in every section of the IP address of the string type in **S** can not be larger than 255.
- If **S** is a string, there are not necessarily three characters in every section of the IP address of the string type. For example, users can enter "192.168.0.1" instead of "192.168.000.001".

Example 1:



3 Example 2:



Additional remark:

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 other binary codes have to be within the range between 16#30 and 16#39. If the other binary codes are not within the range between 16#30 and 16#39, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
3. If the fourth character, the eighth character, and the twelfth character in the string in S are not 16#2E, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
4. If the number of decimals in the string in S is not equal to 3, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
5. If the value converted from the characters in any section of the IP address of the string type in S is larger than 255, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
6. If the number of characters in any section of the IP address of the string type in S is larger than 3, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.
7. If users declare the operand S in ISPSOft, the data type will be ARRAY [8] of WORD/INT.

3.25 Memory Card Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>MWRIT</u>	–	–	Writing the data from the PLC into the memory card	13
FC	<u>MREAD</u>	–	–	Reading the data from the memory card into the PLC	13
FC	<u>MTWRIT</u>	–	–	Writing the string into the memory card	11

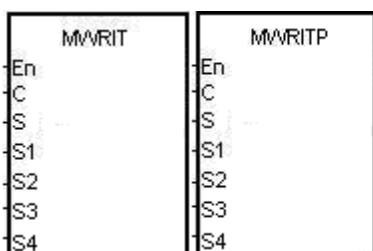
FB/FC	Instruction		Operand				Description						
FC	MWRIT	P	C, S, S₁, S₂, S₃, S₄				Writing the data from the PLC into the memory card						

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●					●							
S		●					●							
S₁			●					●						
S₂		●					●							
S₃		●					●							
S₄			●					●						

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
C								●	●				●	○	○		
S								●	●				●				
S₁								●	●				●	○	○		
S₂								●	●				●	○	○		
S₃								●	●				●				
S₄								●	●				●	○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

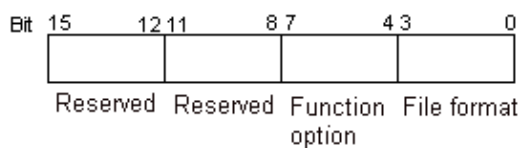


- C** : Control parameter
- S** : Data source
- S₁** : Data length
- S₂** : Line advance
- S₃** : File name
- S₄** : Data address in the file

Explanation:

1. The description of the operands:

- **C**: The control parameter



Item	Code	Description
File format	0	Binary value

Item	Code	Description	
		Default value	
		The file name extension is .dmd.	
		The unit of the value is the word.	
	1		The values are separated by a comma.
			The unit of the value is the word.
			The file name extension is .cvs.
			The ASCII codes are adopted.
			The value which is stored is a hexadecimal value.
	2		The values are separated by a comma.
			The unit of the value is the double word.
	File format	2	The file name extension is .cvs.
			The ASCII codes are adopted.
The value which is stored is a hexadecimal value.			
3			The values are separated by a tab.
			The unit of the value is the word.
			The file name extension is .txt.
			The ASCII codes are adopted.
4			The value which is stored is a hexadecimal value.
			The values are separated by a tab.
			The unit of the value is the double word.
			The file name extension is .txt.
5			The ASCII codes are adopted.
			The value which is stored is a hexadecimal value.
			The values are not separated by any mark.
			The unit of the value is the word.
6			The file name extension is .txt.
			The ASCII codes are adopted.
			The value which is stored is a hexadecimal value.
			The values are not separated by any mark.
The unit of the value is the double word.			
Function option		0	Appending
			The data which is written into the memory card is added after the last value in the file.
			Default value
	If the file does not exist, it is created automatically.		
	1		Overwriting
			The data which is written into the memory card replaces the values in the file starting from the value indicated by the value in S₄ .

Item	Code	Description
		If the file does not exist, it is created automatically.
Reserved	-	The values of bit8~bit15 are 0.

- **S**: The data source
- **S₁**: The length of the data which is written into the file
If the value in **S₁** is 0, the data is not written into the file.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	Double word
Length of the data	The devices in which the data is stored can not exceed the device range, and the size of the data which is written into the file can not be more than four gigabytes. (Please refer to chapter 2 for more information about the devices.)

- **S₂**: The line advance
The value in **S₂** should be within the range between 0 and 256.
- **S₃~S₃+4**: **S₃** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is "Test1", the characters are written into the devices as follows.



- The default folder path:

Model name	Folder path
AHxxEMC-5A	PLC CARD\AH500\UserProg

- **S₄**: The value in the file which is overwritten is indicated by the value in **S₄**.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	The parameter unit is the double word.
Usage	If the function option is 0, S₄ is not used.
	If the function option is 1, the data which is written into the memory card replaces the values in the file starting from the value indicated by the value in S₄ .
	The value in S₄ should indicate the value in the file. If the value in S₄ is 0, the first value in the file is overwritten.

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
SM451	The write protection switch on the memory card ON: The memory card is write protected.

Flag	Description
	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	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

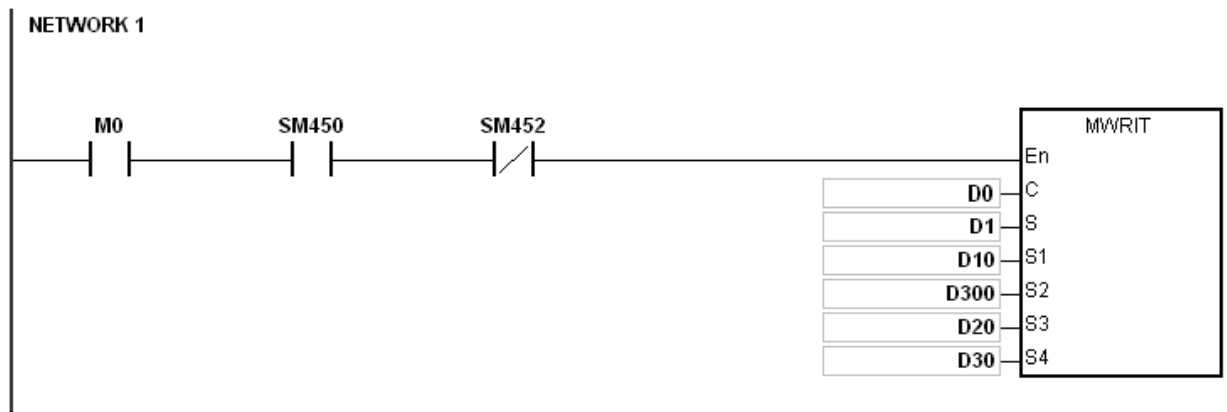
3. The related error codes:

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 can not 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 can not be read from the memory card.
16#0065	The file is a read-only file.

4. If the format of the file into which the data is written is 0, the format of the file from which the data is read is 0. Otherwise, the data can not be read, 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 MWRIT is executed; SM452 is OFF when the execution of MWRIT is complete. You can use MWRITP instruction to avoid the continuous execution of the write-in function on the memory card. Continuous execution could result in errors due to the limit of write-in times is exceeded.



Operand	Setting value	Description
		The file into which the data is written
D0	16#0011	The file format: The values are separated by a comma. The unit of the value is the word. The file name extension is .cvs. The ASCII codes are adopted.
D1	-	The data which is written into the file
D10, D11	16#00000030	The size of the data which is written into the file is 48 words.
D300	16#000A	Ten values are written into every line.
D20	D20=16#6554	The file name is "Test1".

Operand	Setting value	Description
	D21=16#7473 D22=16#0031	
D30 · D31	16#00000000	The data which is written into the memory card replaces the values in the file starting from the first value.

Additional remark:

1. If the value in C exceeds the range, the 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 S1 exceeds the range, the 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 S2 exceeds the range, the 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 S3 exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

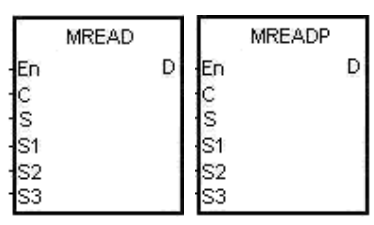
FB/FC	Instruction		Operand		Description
FC	MREAD	P	C, S, S₁, S₂, S₃, D		Reading the data from the memory card into the PLC

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●					●							
S		●					●							
S₁			●					●						
S₂		●					●							
S₃			●					●						
D		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
C								●	●				●	○	○		
S								●	●				●				
S₁								●	●				●	○	○		
S₂								●	●				●	○	○		
S₃								●	●				●	○	○		
D								●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:

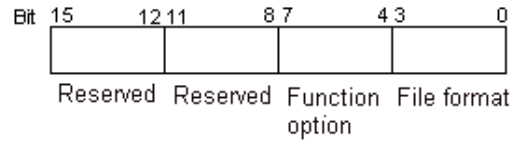


- C** : Control parameter
- S** : File name
- S₁** : Data address in the file
- S₂** : Reserved
- S₃** : Data length
- D** : Data destination

Explanation:

1. The description of the operands:

- **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 the word.

3

Item	Code	Description	
File format	1	The values are separated by a comma.	
		The unit of the value is the word.	
		The file name extension is .cvs.	
		The ASCII codes are adopted.	
		The value which is stored is a hexadecimal value.	
	2	The values are separated by a comma.	
		The unit of the value is the double word.	
		The file name extension is .cvs.	
		The ASCII codes are adopted.	
		The value which is stored is a hexadecimal value.	
	3	The values are separated by a tab.	
	File format	3	The unit of the value is the word.
			The file name extension is .txt.
			The ASCII codes are adopted.
			The value which is stored is a hexadecimal value.
4		The values are separated by a tab.	
		The unit of the value is the double word.	
		The file name extension is .txt.	
		The ASCII codes are adopted.	
		The value which is stored is a hexadecimal value.	
5		The values are not separated by any mark.	
		The unit of the value is the word.	
		The file name extension is .txt.	
	The ASCII codes are adopted.		
	The value which is stored is a hexadecimal value.		
6	The values are not separated by any mark.		
	The unit of the value is the double word.		
	The file name extension is .txt.		
	The ASCII codes are adopted.		
	The value which is stored is a hexadecimal value.		
Function option	0	The values in the file starting from the value indicated by the value in S₁ . 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 the word. If the file format is 2, 4, or 6, the unit of the value is the double word.	
Reserved	-	The values of bit8~bit15 are 0.	

- **S~S+4**: **S** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is "Test1", the characters are

written into the devices as follows.

D	'e'	'T'	ASCII code →	D	16#65	16#54
D+1	't'	's'		D+1	16#74	16#73
D+2	NUL	'1'		D+2	16#00	16#31

- The default folder path:

Model name	Folder path
AHxxEMC-5A	PLC CARD\AH500\UserProg

- **S₁**: The value in the file which is read is indicated by the value in **S₁**.

Item	Description
Value unit	If the file format is 0, 1, 3, or 5, the unit of the value is the word. If the file format is 2, 4, or 6, the unit of the value is the double word.
Parameter unit	The parameter unit is the double word.
Usage	The value in S₁ should indicate the value in the file. If the value in S₁ is 0, the first value in the file is read.

- **S₃**: The length of the data which is read from the file

The devices in which the data is stored can not exceed the device range. If the value in **S₃** 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 **S₃** is the double word.

- **D**: The initial device in which the data is stored.

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
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	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

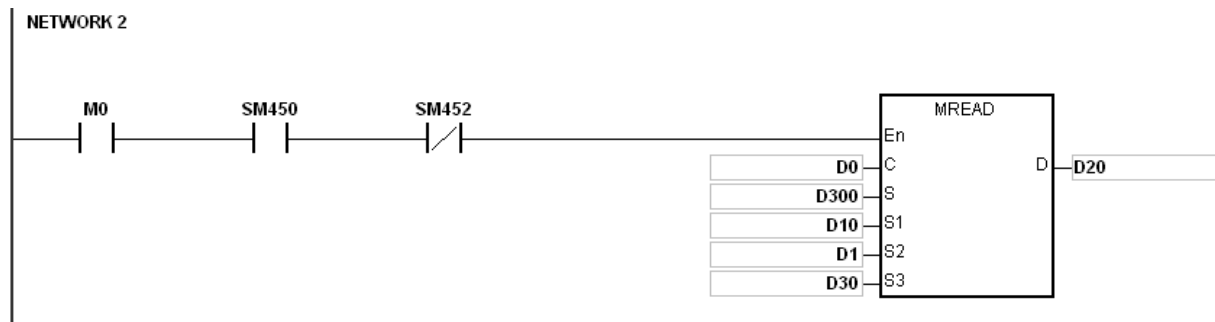
The related error codes:

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 can not 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 can not be read from the memory card.

3. If the format of the file into which the data is written is 0, the format of the file from which the data is read is 0. Otherwise, the data can not be read, 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 MREAD is executed; SM452 is OFF when the execution of MREAD is complete.



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 the word. The file name extension is .csv. The ASCII codes are adopted.
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 which is read from the file is 32 words.
D20	-	The data which is read is stored in D20.

Additional remark:

1. If the value in C exceeds the range, the 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 S2 exceeds the range, the 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 S3 exceeds the range, the 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, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

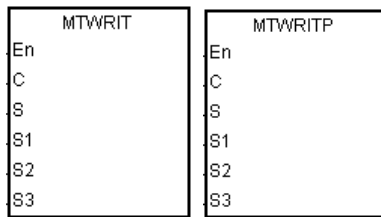
FB/FC	Instruction		Operand	Description
FC	MTWRIT	P	C, S, S₁, S₂, S₃	Writing the string into the memory card

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
C		●					●							
S		●					●							
S₁, S₂		●					●							
S₃		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
C								●	●					○	○		
S								●	●								
S₁, S₂								●	●					○	○		
S₃								●	●								

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



- C** : Control parameter
- S** : Data source
- S₁** : Data length
- S₂** : Separation mark
- S₃** : File name

Explanation:

1. The description of the operands:

- **C**: The control parameter

Parameter value	Description
0 (Appending)	If the file exists, the data which is written into the memory card is added after the last byte in the file. If the file does not exist, it is created automatically.
1 (Overwriting)	If the file exists, the new data which is written into the memory card replaces the old data in the 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 which is written into the file is “12345”, the characters are stored in the devices as follows. Owing to the fact that a byte is taken as the basic unit, the first character is stored in the low byte in D300, the second character is stored in the high byte in D300. The same applies to the other characters. “16#00” is stored in the high byte in D300+2, and indicates the end of the string.

D300	D300+1	D300+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

- **S₁**: The length of the data which is written into the memory card

A byte is taken as the basic unit. The devices in which the data is stored can not exceed the device range, and the length of the data which is written into the memory card can not be more than 255 bytes.

- **S₂**: The separation mark

If the value in **S₁** is 6, the value in **S₂** is written into the memory card as follows.

S₂		Description
High byte	Low byte	
16#00	16#00 or not 16#00	The 6-byte data is written into the file.
Not 16#00	16#00	The 7-byte data is written into the file. The value in the high byte in S₂ is the value in the seventh byte.
Not 16#00	Not 16#00	The 8-byte data is written into the file. The value in the high byte in S₂ is the value in the seventh byte, and the value in low byte in S₂ is the value in the eighth byte.

- **S₃~S₃₊₄**: **S₃** occupies five devices. Nine characters at most constitute a file name, including 16#00. If the string does not end with 16#00, the error occurs. If the ending character is read, the reading of the characters stops, and whether the file name is legal is checked. The characters which can be used to constitute a file name are A~Z, a~z, and 0~9. Besides, the file name extension depends on the file format. The file which is created is in the default folder. If the file name is "Test1", the characters are written into the devices as follows.



- The default folder path

Model name	Folder path
AHxxEMC-5A	PLC CARD\AH500\UserProg

2. The related flags:

Flag	Description
SM450	If the memory card is in the CPU module, the flag is ON.
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	If an error occurs during the operation of the memory card, the flag is ON. If the flag is ON, users have to reset it to OFF. The error code is stored in SR453.

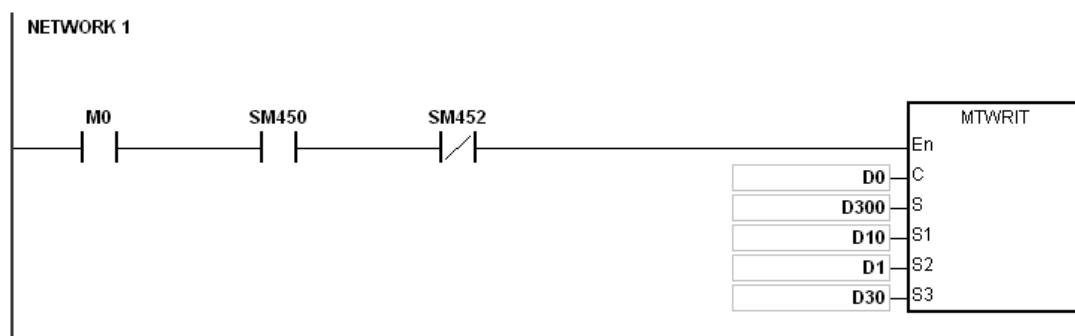
3. The related error codes:

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.

Error code	Description
16#0060	The default folder can not 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 can not 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 MTWRIT is executed; SM452 is OFF when the execution of MTWRIT is complete.



Operand	Setting value	Description
D0	16#0001	The file into which the data is written The file format: The unit of the character is the byte. The file name extension is .txt. The ASCII codes are adopted. The data in D300 is written into the file.
D300	-	The data which is written into the file
D10	16#000A	The size of the string which is written into the file is 32 bytes.
D1	16#0A00	After the data is written into the file, the separation mark is added after the last byte in the file.
D30	D30=16#6554 D31=16#7473 D32=16#0031	The file name is "Test1".

Additional remark:

1. If the value in C exceeds the range, the 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 S1 exceeds the range, the 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 S3 exceeds the range, the operation error occurs, the instruction is not executed, SM0 is ON, and the error code in SR0 is 16#2003.

3.26 Task Control Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>TKON</u>	–	✓	Enabling the cyclic task	3
FC	<u>TKOFF</u>	–	✓	Disabling the cyclic task	3

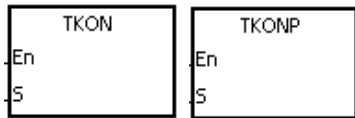
FB/FC	Instruction		Operand		Description
FC	TKON	P	S		Enabling the cyclic task

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	"\$"	DF
S	●	●						●	●		●						S

Pulse instruction	116-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S : Task number

Explanation:

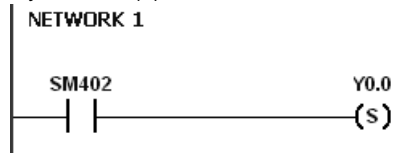
1. The cyclic task specified by **S** is enabled.
2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSOft.
3. The description of the operands:
 - The operand **S** should be within the range between 0 and 31.
 - Please refer to *ISPSOft User Manual* for more information about creating and enabling the tasks.

Example:

1. When the PLC runs, cyclic task (0) is enabled. Since the instruction TKON in cyclic task (0) is executed, cyclic task (1) is enabled, and Y0.0 is ON.
2. The two cyclic tasks are created in ISPSOft. Cyclic task (0) is enabled when the PLC runs, and cyclic task (1) is not enabled when the PLC runs.
3. Cyclic task (1) is enabled by the execution of the instruction TKON in cyclic task (0).



Cyclic task (1) is executed.



Additional remark:

Please refer to *ISPSOft User Manual* for more information related to tasks.

FB/FC	Instruction		Operand	Description
FC	TKOFF	P	S	Disabling the cyclic task

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S	●	●						●	●		●						S

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



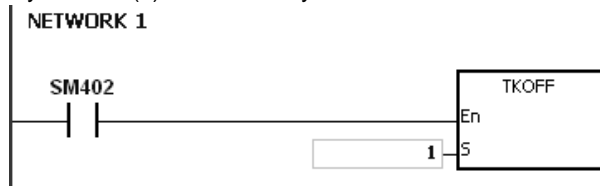
S : Task number

Explanation:

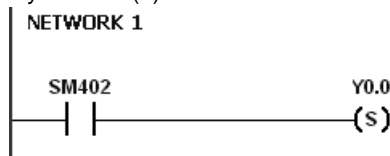
1. The cyclic task specified by **S** is disabled.
2. When the PLC runs, the execution of the cyclic tasks depends on the setting of the cyclic tasks in ISPSOft.
3. The description of the operands:
 - The operand **S** should be within the range between 0 and 31.
 - Please refer to *ISPSOft User Manual* for more information about creating and enabling the tasks.

Example:

1. When the PLC runs, cyclic task (0) and cyclic task (1) are enabled. Since the instruction TKOOF in cyclic task (0) is executed, cyclic task (1) is disabled, and Y0.0 is OFF.
2. The two cyclic tasks are created in ISPSOft. Cyclic task (0) and cyclic task (1) are enabled when the PLC runs, and cyclic task (1) is disabled when the instruction TKOFF in cyclic task (0) is executed.
3. Cyclic task (1) is disabled by the execution of the instruction TKOFF in cyclic task (0).



Cyclic task (1) is not executed.



Additional remark:

Please refer to *ISPSOft User Manual* for more information related to tasks.

3.27 SFC Control Instructions

FB/FC	Instruction		Pulse instruction	Description	Step
	16-bit	32-bit			
FC	<u>SFCRUN</u>	–	–	SFC Run	3
FC	<u>SFCPSE</u>	–	–	SFC Pause	3
FC	<u>SFCSTP</u>	–	–	SFC Stop	

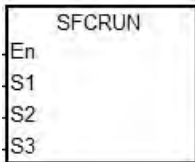
FB/FC	Instruction		Operand	Description
FC	SFCRUN	P	S₁, S₂, S₃	SFC Run

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁														
S₂		●					●							
S₃														

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁																	
S₂	●	●						●	●					○	○		
S₃																	

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : Name of the SFC POU

S₂ : Function code

S₃ : Device address

Explanation:

- The designated SFC program of **S₁** will be activated according to the setups of **S₂**.
- When the instruction is executed, the SFC POU designated by **S₁** will be activated only when the SFC POU is being scanned.
- Operand:
 - **S₁** defines the name of the SFC POU.
 - When the designated SFC POU of **S₁** is executed, the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC program will be cleared when **S₂**=0 or 1, and the execution will start according to the value specified in **S₂**.

S₂=0, the system will execute the SFC POU from the initial Step.

S₂=1, the system will execute the SFC POU from the designated Step of **S₃**.
 - **S₂**=2, the status and the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC will NOT be cleared and the system will start executing from where it pauses.
 - **S₃** designates the step to be started in the SFC program of **S₁**.
- The range of **S₂** is 0 to 2. When it is out of range, it will be seen as 0.
- When the state of the SFC POU is RUN, executing this instruction is invalid.

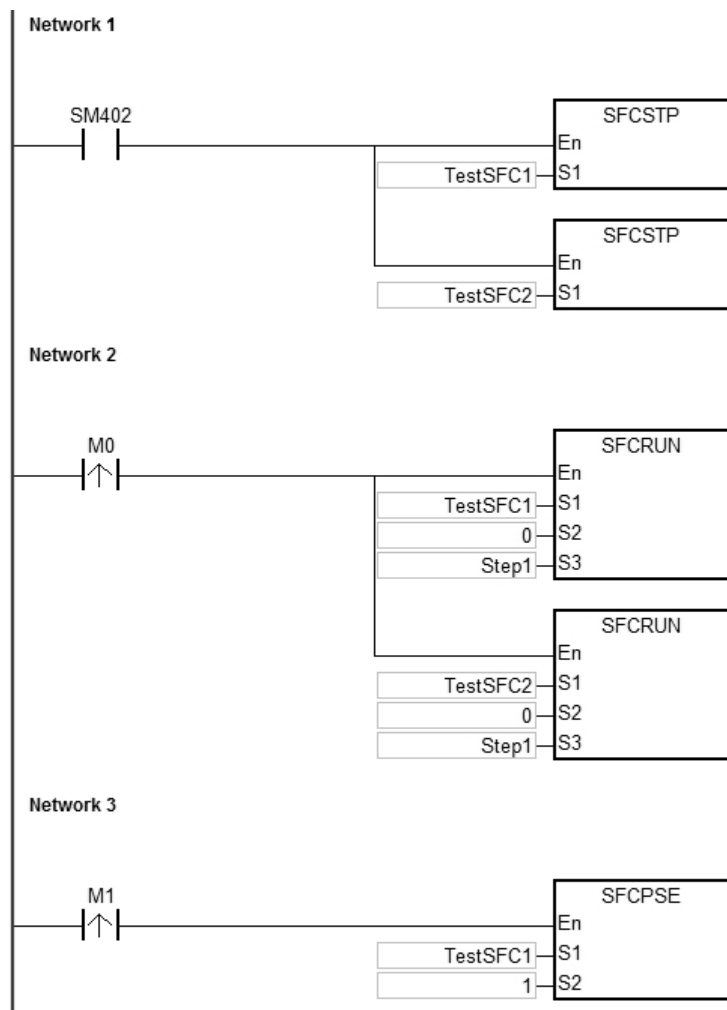
Programing Example:

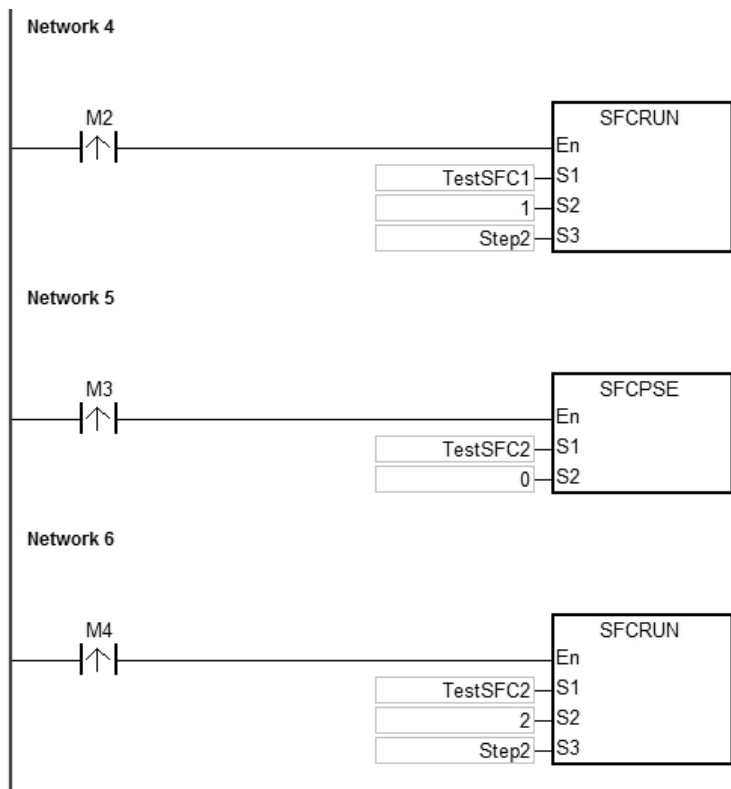
Set up one LD(ladder) POU and specify its POU name as Main, and 2 SFC POUs with the names of TestSFC1 and TestSFC2.

1. When the program is executed (RUN), TestSFC1 and TestSFC2 will execute the SFCSTP, and 2 SFC POU's will stop executing.
2. When M0 is set from OFF to ON, TestSFC1/ TestSFC2 POU will execute the SFCRUN* instructions. Refer to the contents of TestSFC1 and TestSFC2 for execution details of the 2 POU's. When S2 is set to 0, the status and the parameters of the SFC will be cleared and will begin to execute from STEP 1. When S2 is set to 1, the status and the parameters will be cleared and will begin to execute from the designated STEP of S3.
3. When M1 is set from OFF to ON, TestSFC1 POU will pause. When S2 is set to 1, all the executing actions and the outputs of the SFC will be cleared, and the system will run the final scan.
4. When M2 is set from OFF to ON, TestSFC1 POU will execute its actions. When S2 is set to 1, the status and the parameters will be cleared, and the system will begin to execute from STEP 2.
5. When M3 is set from OFF to ON, TestSFC2 POU will pause. When S2 is set to 0, all the executing actions of the SFC and the outputs will be kept, and the system will not run the final scan.
6. When M4 is set from OFF to ON, TestSFC1 POU will execute its actions. When S2 is set to 2, the status and the parameters will be kept and will begin to execute from where it pauses.

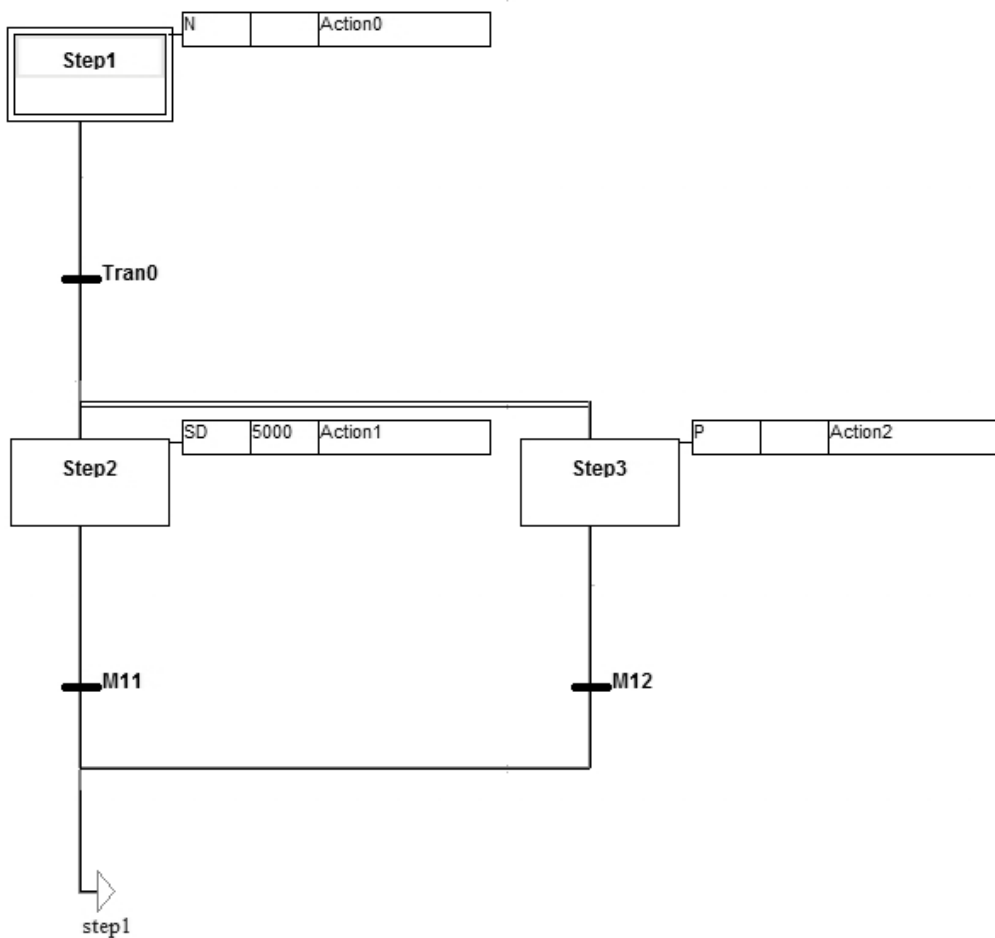
Note*: SFCRUN will activate SPC POU at the next scan.

Main POU

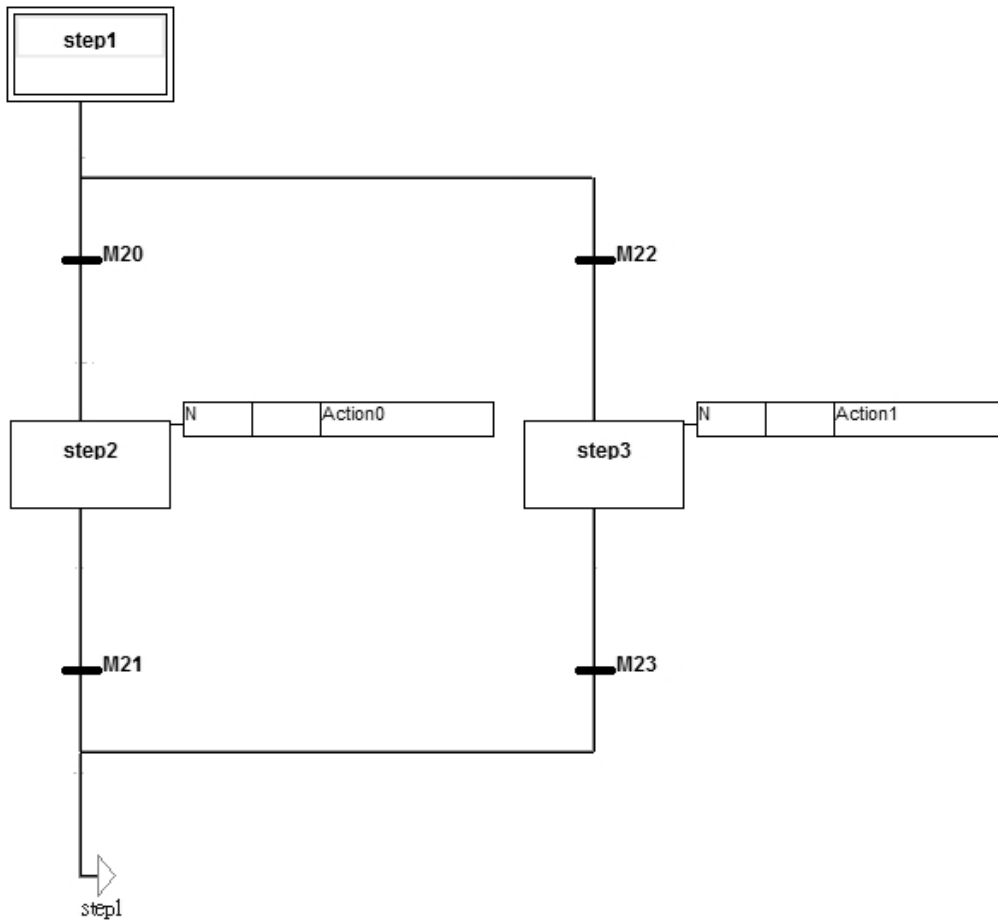




TestSFC1 POU



TestSFC2 POU

**Additional Remark:**

Please refer to **ISPSOft User Manual** for more information related to SFC.

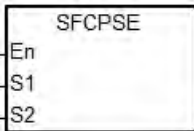
FB/FC	Instruction		Operand	Description
FC	SFCPSE	P	S₁, S₂	SFC Pause

Data type	BOOL	WORD	DWORD	LWORD	UINT	UDINT	INT	DINT	LINT	REAL	LREAL	TMR	CNT	STRING
S₁														
S₂		●					●							

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S₁																	
S₂	●	●						●	●		●			○	○		

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : Name of the SFC POU

S₂ : Function code

Explanation:

1. The designated SFC POU of **S₁** will pause according to the setups of **S₂**.
2. When the instruction is executed, the SFC POU designated by **S₁** will be paused only when the SFC POU is being scanned.
3. When pausing, the status and the parameters such as SFC/STEP/ACTION/TRANSITION of the SFC will be kept.
4. Operand:
 - **S₁** defines the name of the SFC POU.
 - When **S₂**=0, all the executing actions of the SFC and the outputs will be kept, and the system will not run the final scan.
 - When **S₂**=1, all the executing actions and the outputs of the SFC POU will be cleared, and the system will run the final scan.
5. The range of **S₂** is 0 to 1. When it is out of range, it will be seen as 0.
6. When the state of the SFC POU is PAUSE/STOP, executing this instruction is invalid.

Programing Example:

Please refer to the SFCRUN programing example for more information.

Additional Remark:

Please refer to **ISPSOFT User Manual** for more information related to SFC.

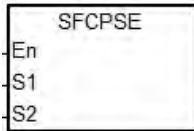
FB/FC	Instruction			Operand			Description		
FC	SFCSTP	P		S			SFC Stop		

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

Device	X	Y	M	S	T	C	HC/AC	D	L	SM/AM/AR	SR/AR	E	PR	K	16#	“\$”	DF
S																	

Pulse instruction	16-bit instruction	32-bit instruction
AH Motion CPU	AH Motion CPU	-

Graphic expression:



S₁ : Name of the SFC POU

S₂ : Function code

Explanation:

1. The designated SFC POU of **S** will stop.
2. When the instruction is executed, the SFC POU designated by **S₁** will stop only when the SFC POU is being scanned.
3. When stopping, the status and the parameters of the SFC will be cleared, and the system will run the final scan.
4. When the state of the SFC POU is STOP, executing this instruction is invalid.

Programming Example:

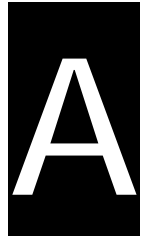
Please refer to the SFCRUN programming example for more information.

Additional Remark:

Please refer to ISPSOft User Manual for more information related to SFC.

Memo

3



Appendices

Table of Contents

A.1.	Special Auxiliary Relays Table (SM)	A-2
A.1.1.	Refresh Time of Special Auxiliary Relay	A-25
A.2.	Special Data Registers Table (SR)	A-30
A.2.1.	Refresh Time of Special Data Registers	A-45
A.3.	Additional Remarks on SM and SR	A-48
A.4.	Standard Instructions (Sort by Alphabet)	A-60
A.5.	Error Codes and Troubleshooting	A-78
A.5.1.	Error Codes and Indicators	A-78
AHxxEMC-5A	A-80	
Analog I/O Modules and Temperature Measurement Modules	A-106	
AH02HC-5A/AH04HC-5A	A-108	
AH05PM-5A/AH10PM-5A/AH15PM-5A	A-109	
AH20MC-5A	A-110	
AH10COPM-5A	A-112	
AH10SCM-5A	A-113	
A.5.2.	Error Codes and Troubleshooting	A-113
AHxxEMC-5A	A-113	
Analog I/O Modules and Temperature Measurement Modules	A-142	
AH02HC-5A/AH04HC-5A	A-145	
AH05PM-5A/AH10PM-5A/AH15PM-5A	A-148	
AH20MC-5A	A-149	
AH10SCM-5A	A-151	
AH10COPM-5A	A-152	
A.5.3.	Troubleshooting for Limitation Errors	A-153
Troubleshooting for the software limit errors	A-153	
Troubleshooting for the hardware limit errors	A-153	
A.6.	Table of Data Type Unit (DUT): Enum	A-155

A.1. Special Auxiliary Relays Table (SM)

Every special auxiliary relay has its specific function. Please do not use the special auxiliary relays which are not defined. The special auxiliary relays and their functions are listed as follows. **For more information regarding the SM numbers marked “*”, refer to A.3 Additional Remarks on SM and SR.** “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	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM0	Operation error	○	○	OFF	OFF	-	R	OFF
SM1	The operation error is locked.	○	○	OFF	OFF	-	R	OFF
SM5	Instruction/Operand inspection error	○	○	OFF	OFF	-	R	OFF
*SM8	Watchdog timer error	○	○	OFF	-	-	R	OFF
SM9	System error	○	○	OFF	-	-	R	OFF
SM10	I/O bus error	○	○	OFF	-	-	R	OFF
*SM22	Clearing the error log	○	○	OFF	-	-	R/W	OFF
SM23	Clearing the download log	○	○	OFF	-	-	R/W	OFF
SM24	Clearing the state-changing log of the PLC	○	○	OFF	-	-	R/W	OFF
SM25	The online-editing processing flag is on when the online-editing mode starts.	○	○	OFF	-	-	R	OFF
SM26	The debugging mode processing flag is on when the debugging mode starts.	○	○	OFF	-	-	R	OFF
*SM96	The data is sent through COM1.	×	×	OFF	OFF	-	R/W	OFF
*SM97	The data is sent through COM2.	○	×	OFF	OFF	-	R/W	OFF
*SM98	Waiting to receive the reply through COM1	×	×	OFF	OFF	-	R	OFF
*SM99	Waiting to receive the reply through COM2	○	×	OFF	OFF	-	R	OFF
*SM100	Reception through COM1 is complete.	×	×	OFF	OFF	-	R/W	OFF
*SM101	Reception through COM2 is complete.	○	×	OFF	OFF	-	R/W	OFF
*SM102	An error occurs during the reception of the data through COM1 by using the instruction MODRW or the instruction RS.	×	×	OFF	OFF	-	R	OFF
*SM103	An error occurs during the reception of the data through COM2 by using the instruction MODRW or the instruction RS.	○	×	OFF	OFF	-	R	OFF
*SM104	No data is received through COM1 after a	×	×	OFF	OFF	-	R/W	OFF

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	specified period of time.							
*SM105	No data is received through COM2 after a specified period of time.	○	×	OFF	OFF	–	R/W	OFF
*SM106	Choice made by COM1 between the 8-bit processing mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	×	×	OFF	–	–	R/W	OFF
*SM107	Choice made by COM2 between the 8-bit mode and the 16-bit processing mode ON: The 8-bit processing mode OFF: The 16-bit processing mode	○	×	OFF	–	–	R/W	OFF
*SM109	Reception of specific word through COM2 is complete.	○	×	–	–	–	R/W	OFF
*SM204	All non-latched areas are cleared.	○	○	OFF	–	–	R/W	OFF
*SM205	All latched areas are cleared.	○	○	OFF	–	–	R/W	OFF
SM206	Inhibiting all output	○	○	OFF	–	–	R/W	OFF
*SM209	The communication protocol of COM1 changes (in accordance with SM210, SR201, SR209, and SR215).	○	○	OFF	–	–	R/W	OFF
*SM210	Choice made by COM1 between the ASCII mode and the RTU mode ON: The RTU mode	○	○	OFF	–	–	R/W	OFF
*SM211	The communication protocol of COM2 changes (in accordance with SM212, SR202, SR212, and SR216).	○	×	OFF	–	–	R/W	OFF
*SM212	Choice made by COM2 between the ASCII mode and the RTU mode ON: The RTU mode	○	×	OFF	–	–	R/W	OFF
SM215	Running state of the PLC	○	○	OFF	ON	OFF	R/W	OFF
SM220	Calibrating the real-time clock within ±30 seconds	○	○	OFF	OFF	–	R/W	OFF
*SM400	Normally-open contact	○	○	ON	ON	ON	R	ON
*SM401	Normally-closed contact	○	○	OFF	OFF	OFF	R	OFF
*SM402	The pulse is ON at the time when the PLC runs.	○	○	OFF	ON	OFF	R	OFF
*SM403	The pulse is OFF at the time when the PLC runs.	○	○	ON	OFF	ON	R	ON
*SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and is OFF for 5 milliseconds	○	○	OFF	–	–	R	OFF
*SM405	100 millisecond clock pulse during which the pulse	○	○	OFF	–	–	R	OFF

SM	Function	AH-xxEMC(CPU)		AH-xxEMC(Module)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	is ON for 50 milliseconds and is OFF for 50 milliseconds									
*SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and is OFF for 100 milliseconds	○	○	OFF	–	–	R	OFF		
*SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and is OFF for 500 milliseconds	○	○	OFF	–	–	R	OFF		
*SM408	Two second clock pulse during which the pulse is ON for one second and is OFF for one second	○	○	OFF	–	–	R	OFF		
*SM409	2n second clock pulse during which the pulse is ON for n seconds and is OFF for n seconds The interval n is specified by SR409.	○	○	OFF	–	–	R	OFF		
*SM410	2n millisecond clock pulse during which the pulse is ON for n milliseconds and is OFF for n milliseconds The interval n is specified by SR410.	○	○	OFF	–	–	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	–	–	R	OFF		
*SM453	An error occurs during the operation of the memory card. ON: An error occurs.	○	○	OFF	–	–	R	OFF		
SM600	Zero flag	○	○	OFF	–	–	R	OFF		
SM601	Borrow flag	○	○	OFF	–	–	R	OFF		
SM602	Carry flag	○	○	OFF	–	–	R	OFF		
SM603	The execution of the instruction SORT is complete.	○	○	OFF	–	–	R	OFF		
SM604	Setting the working mode of the instruction SORT. ON: The descending order OFF: The ascending order	○	○	OFF	–	–	R/W	OFF		
SM605	Designating the working mode of the instruction SMOV	○	○	OFF	–	–	R/W	OFF		
SM606	8-bit or 16-bit working mode	○	○	OFF	–	–	R/W	OFF		
SM607	It is the matrix comparison flag. ON: Comparing the equivalent values OFF: Comparing the different values	○	○	OFF	–	–	R/W	OFF		
SM608	The matrix comparison comes to an end. When the last bits are compared, SM608 is ON.	○	○	OFF	–	–	R	OFF		

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)		AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM609	When SM609 is ON, the comparison starts from bit 0.	○	○		OFF	–	–	R	OFF
SM610	It is the matrix bit search flag. When the matching bits are compared, the comparison stops immediately, and SM610 is ON.	○	○		OFF	–	–	R	OFF
SM611	It is the matrix pointer error flag. When the value of the pointer exceeds the comparison range, SM611 is ON.	○	○		OFF	–	–	R	OFF
SM612	It is the matrix pointer increasing flag. The current value of the pointer increases by one.	○	○		OFF	–	–	R/W	OFF
SM613	It is the matrix pointer clearing flag. The current value of the pointer is cleared to zero.	○	○		OFF	–	–	R/W	OFF
SM614	It is the carry flag for the matrix rotation/shift/output.	○	○		OFF	–	–	R	OFF
SM615	It is the borrow flag for the matrix shift/output.	○	○		OFF	–	–	R/W	OFF
SM616	It is the direction flag for the matrix rotation/shift. The bits are shifted leftward when SM616 is OFF, whereas the bits are shifted rightward when SM616 is ON.	○	○		OFF	–	–	R/W	OFF
SM617	The bits with the value 0 or 1 are counted.	○	○		OFF	–	–	R/W	OFF
SM618	It is ON when the matrix counting result is 0.	○	○		OFF	–	–	R/W	OFF
SM619	It is ON when the instruction EI is executed.	○	○		OFF	OFF	–	R	OFF
SM620	When the results gotten from the comparison by using the instruction CMPT# are that all devices are ON, SM620 is ON.	○	○		OFF	–	–	R	OFF
SM621	It sets the counting mode of HC0. (HC0 counts down when SM621 is ON.)	○	○		OFF	–	–	R/W	OFF
SM622	It sets the counting mode of HC. (HC1 counts down when SM622 is ON.)	○	○		OFF	–	–	R/W	OFF
SM623	It sets the counting mode of HC2. (HC2 counts down when SM623 is ON.)	○	○		OFF	–	–	R/W	OFF
SM624	It sets the counting mode of HC3. (HC3 counts down when SM624 is ON.)	○	○		OFF	–	–	R/W	OFF
SM625	It sets the counting mode of HC4. (HC4 counts down when SM625 is ON.)	○	○		OFF	–	–	R/W	OFF
SM626	It sets the counting mode of HC5. (HC5 counts down when SM626 is ON.)	○	○		OFF	–	–	R/W	OFF
SM627	It sets the counting mode of HC6. (HC6 counts	○	○		OFF	–	–	R/W	OFF

A1

SM	Function	AH-xxEMC(CPU)		AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	down when SM627 is ON.)								
SM628	It sets the counting mode of HC7. (HC7 counts down when SM628 is ON.)	○	○		OFF	-	-	R/W	OFF
SM629	It sets the counting mode of HC8. (HC8 counts down when SM629 is ON.)	○	○		OFF	-	-	R/W	OFF
SM630	It sets the counting mode of HC9. (HC9 counts down when SM630 is ON.)	○	○		OFF	-	-	R/W	OFF
SM631	It sets the counting mode of HC10. (HC10 counts down when SM631 is ON.)	○	○		OFF	-	-	R/W	OFF
SM632	It sets the counting mode of HC11. (HC11 counts down when SM632 is ON.)	○	○		OFF	-	-	R/W	OFF
SM633	It sets the counting mode of HC12. (HC12 counts down when SM633 is ON.)	○	○		OFF	-	-	R/W	OFF
SM634	It sets the counting mode of HC13. (HC13 counts down when SM634 is ON.)	○	○		OFF	-	-	R/W	OFF
SM635	It sets the counting mode of HC14. (HC14 counts down when SM635 is ON.)	○	○		OFF	-	-	R/W	OFF
SM636	It sets the counting mode of HC15. (HC15 counts down when SM636 is ON.)	○	○		OFF	-	-	R/W	OFF
SM637	It sets the counting mode of HC16. (HC16 counts down when SM637 is ON.)	○	○		OFF	-	-	R/W	OFF
SM638	It sets the counting mode of HC17. (HC17 counts down when SM638 is ON.)	○	○		OFF	-	-	R/W	OFF
SM639	It sets the counting mode of HC18. (HC18 counts down when SM639 is ON.)	○	○		OFF	-	-	R/W	OFF
SM640	It sets the counting mode of HC19. (HC19 counts down when SM640 is ON.)	○	○		OFF	-	-	R/W	OFF
SM641	It sets the counting mode of HC20. (HC20 counts down when SM641 is ON.)	○	○		OFF	-	-	R/W	OFF
SM642	It sets the counting mode of HC21. (HC21 counts down when SM642 is ON.)	○	○		OFF	-	-	R/W	OFF
SM643	It sets the counting mode of HC22. (HC22 counts down when SM643 is ON.)	○	○		OFF	-	-	R/W	OFF
SM644	It sets the counting mode of HC23. (HC23 counts down when SM644 is ON.)	○	○		OFF	-	-	R/W	OFF
SM645	It sets the counting mode of HC24. (HC24 counts down when SM645 is ON.)	○	○		OFF	-	-	R/W	OFF
SM646	It sets the counting mode of HC25. (HC25 counts down when SM646 is ON.)	○	○		OFF	-	-	R/W	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM647	It sets the counting mode of HC26. (HC26 counts down when SM647 is ON.)	○	○	OFF	–	–	R/W	OFF
SM648	It sets the counting mode of HC27. (HC27 counts down when SM648 is ON.)	○	○	OFF	–	–	R/W	OFF
SM649	It sets the counting mode of HC28. (HC28 counts down when SM649 is ON.)	○	○	OFF	–	–	R/W	OFF
SM650	It sets the counting mode of HC29. (HC29 counts down when SM650 is ON.)	○	○	OFF	–	–	R/W	OFF
SM651	It sets the counting mode of HC30. (HC30 counts down when SM651 is ON.)	○	○	OFF	–	–	R/W	OFF
SM652	It sets the counting mode of HC31. (HC31 counts down when SM652 is ON.)	○	○	OFF	–	–	R/W	OFF
SM653	It sets the counting mode of HC32. (HC32 counts down when SM653 is ON.)	○	○	OFF	–	–	R/W	OFF
SM654	It sets the counting mode of HC33. (HC33 counts down when SM653 is ON.)	○	○	OFF	–	–	R/W	OFF
SM655	It sets the counting mode of HC34. (HC34 counts down when SM655 is ON.)	○	○	OFF	–	–	R/W	OFF
SM656	It sets the counting mode of HC35. (HC35 counts down when SM656 is ON.)	○	○	OFF	–	–	R/W	OFF
SM657	It sets the counting mode of HC36. (HC36 counts down when SM657 is ON.)	○	○	OFF	–	–	R/W	OFF
SM658	It sets the counting mode of HC37. (HC37 counts down when SM658 is ON.)	○	○	OFF	–	–	R/W	OFF
SM659	It sets the counting mode of HC38. (HC38 counts down when SM659 is ON.)	○	○	OFF	–	–	R/W	OFF
SM660	It sets the counting mode of HC39. (HC39 counts down when SM660 is ON.)	○	○	OFF	–	–	R/W	OFF
SM661	It sets the counting mode of HC40. (HC40 counts down when SM661 is ON.)	○	○	OFF	–	–	R/W	OFF
SM662	It sets the counting mode of HC41. (HC41 counts down when SM662 is ON.)	○	○	OFF	–	–	R/W	OFF
SM663	It sets the counting mode of HC42. (HC42 counts down when SM663 is ON.)	○	○	OFF	–	–	R/W	OFF
SM664	It sets the counting mode of HC43. (HC43 counts down when SM664 is ON.)	○	○	OFF	–	–	R/W	OFF
SM665	It sets the counting mode of HC44. (HC44 counts down when SM665 is ON.)	○	○	OFF	–	–	R/W	OFF
SM666	It sets the counting mode of HC45. (HC45 counts down when SM666 is ON.)	○	○	OFF	–	–	R/W	OFF

A1

SM	Function	AH-xxEMC(CPU)		AH-xxEMC(Module)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM667	It sets the counting mode of HC46. (HC46 counts down when SM667 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM668	It sets the counting mode of HC47. (HC47 counts down when SM668 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM669	It sets the counting mode of HC48. (HC48 counts down when SM669 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM670	It sets the counting mode of HC49. (HC49 counts down when SM670 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM671	It sets the counting mode of HC50. (HC50 counts down when SM671 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM672	It sets the counting mode of HC51. (HC51 counts down when SM672 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM673	It sets the counting mode of HC52. (HC52 counts down when SM673 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM674	It sets the counting mode of HC53. (HC53 counts down when SM674 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM675	It sets the counting mode of HC54. (HC54 counts down when SM675 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM676	It sets the counting mode of HC55. (HC55 counts down when SM676 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM677	It sets the counting mode of HC56. (HC56 counts down when SM677 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM678	It sets the counting mode of HC57. (HC57 counts down when SM678 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM679	It sets the counting mode of HC58. (HC58 counts down when SM679 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM680	It sets the counting mode of HC59. (HC59 counts down when SM680 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM681	It sets the counting mode of HC60. (HC60 counts down when SM681 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM682	It sets the counting mode of HC61. (HC61 counts down when SM682 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM683	It sets the counting mode of HC62. (HC62 counts down when SM683 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM684	It sets the counting mode of HC63. (HC63 counts down when SM684 is ON.)	○	○	○	○	OFF	–	–	R/W	OFF
SM685	The instruction DSCLP uses the floating-point operation.	○	○	○	○	OFF	–	–	R/W	OFF
SM686	When SM686 is ON, the instruction RAMP is executed.	○	○	○	○	OFF	–	–	R/W	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM687	The execution of the instruction RAMP is complete.	○	○	OFF	–	–	R/W	OFF
SM688	The execution of the instruction INCD is complete.	○	○	OFF	–	–	R/W	OFF
SM690	String control mode	○	○	OFF	–	–	R/W	OFF
SM691	The input mode of the instruction HKY is the 16-bit mode. The input is the hexadecimal input if SM691 is ON, whereas A~F are function keys if it is OFF.	○	○	OFF	–	–	R/W	OFF
SM692	After the execution of the instruction HKY is complete, SM692 is ON for a scan cycle.	○	○	OFF	–	–	R/W	OFF
SM693	After the execution of the instruction SEGL is complete, SM693 is ON for a scan cycle.	○	○	OFF	–	–	R/W	OFF
SM694	After the execution of the instruction DSW is complete, SM694 is ON for a scan cycle.	○	○	OFF	–	–	R/W	OFF
SM695	It is the radian/degree flag. ON: The degree	○	○	OFF	–	–	R/W	OFF
SM700	MODBUS TCP connection 1 is activated.	○	○	OFF	–	–	R/W	OFF
SM701	MODBUS TCP connection 2 is activated.	○	○	OFF	–	–	R/W	OFF
SM702	MODBUS TCP connection 3 is activated.	○	○	OFF	–	–	R/W	OFF
SM703	MODBUS TCP connection 4 is activated.	○	○	OFF	–	–	R/W	OFF
SM704	MODBUS TCP connection 5 is activated.	○	○	OFF	–	–	R/W	OFF
SM705	MODBUS TCP connection 6 is activated.	○	○	OFF	–	–	R/W	OFF
SM706	MODBUS TCP connection 7 is activated.	○	○	OFF	–	–	R/W	OFF
SM707	MODBUS TCP connection 8 is activated.	○	○	OFF	–	–	R/W	OFF
SM708	MODBUS TCP connection 9 is activated.	○	○	OFF	–	–	R/W	OFF
SM709	MODBUS TCP connection 10 is activated.	○	○	OFF	–	–	R/W	OFF
SM710	MODBUS TCP connection 11 is activated.	○	○	OFF	–	–	R/W	OFF
SM711	MODBUS TCP connection 12 is activated.	○	○	OFF	–	–	R/W	OFF
SM712	MODBUS TCP connection 13 is activated.	○	○	OFF	–	–	R/W	OFF
SM713	MODBUS TCP connection 14 is activated.	○	○	OFF	–	–	R/W	OFF
SM714	MODBUS TCP connection 15 is activated.	○	○	OFF	–	–	R/W	OFF
SM715	MODBUS TCP connection 16 is activated.	○	○	OFF	–	–	R/W	OFF
SM716	MODBUS TCP connection 17 is activated.	○	○	OFF	–	–	R/W	OFF
SM717	MODBUS TCP connection 18 is activated.	○	○	OFF	–	–	R/W	OFF

A1

SM	Function	AH-xxEMC		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
		(CPU)	(Module)					
SM718	MODBUS TCP connection 19 is activated.	○	○	OFF	-	-	R/W	OFF
SM719	MODBUS TCP connection 20 is activated.	○	○	OFF	-	-	R/W	OFF
SM720	MODBUS TCP connection 21 is activated.	○	○	OFF	-	-	R/W	OFF
SM721	MODBUS TCP connection 22 is activated.	○	○	OFF	-	-	R/W	OFF
SM722	MODBUS TCP connection 23 is activated.	○	○	OFF	-	-	R/W	OFF
SM723	MODBUS TCP connection 24 is activated.	○	○	OFF	-	-	R/W	OFF
SM724	MODBUS TCP connection 25 is activated.	○	○	OFF	-	-	R/W	OFF
SM725	MODBUS TCP connection 26 is activated.	○	○	OFF	-	-	R/W	OFF
SM726	MODBUS TCP connection 27 is activated.	○	○	OFF	-	-	R/W	OFF
SM727	MODBUS TCP connection 28 is activated.	○	○	OFF	-	-	R/W	OFF
SM728	MODBUS TCP connection 29 is activated.	○	○	OFF	-	-	R/W	OFF
SM729	MODBUS TCP connection 30 is activated.	○	○	OFF	-	-	R/W	OFF
SM730	MODBUS TCP connection 31 is activated.	○	○	OFF	-	-	R/W	OFF
SM731	MODBUS TCP connection 32 is activated.	○	○	OFF	-	-	R/W	OFF
SM732	MODBUS TCP connection 33 is activated.	○	○	OFF	-	-	R/W	OFF
SM733	MODBUS TCP connection 34 is activated.	○	○	OFF	-	-	R/W	OFF
SM734	MODBUS TCP connection 35 is activated.	○	○	OFF	-	-	R/W	OFF
SM735	MODBUS TCP connection 36 is activated.	○	○	OFF	-	-	R/W	OFF
SM736	MODBUS TCP connection 37 is activated.	○	○	OFF	-	-	R/W	OFF
SM737	MODBUS TCP connection 38 is activated.	○	○	OFF	-	-	R/W	OFF
SM738	MODBUS TCP connection 39 is activated.	○	○	OFF	-	-	R/W	OFF
SM739	MODBUS TCP connection 40 is activated.	○	○	OFF	-	-	R/W	OFF
SM740	MODBUS TCP connection 41 is activated.	○	○	OFF	-	-	R/W	OFF
SM741	MODBUS TCP connection 42 is activated.	○	○	OFF	-	-	R/W	OFF
SM742	MODBUS TCP connection 43 is activated.	○	○	OFF	-	-	R/W	OFF
SM743	MODBUS TCP connection 44 is activated.	○	○	OFF	-	-	R/W	OFF
SM744	MODBUS TCP connection 45 is activated.	○	○	OFF	-	-	R/W	OFF
SM745	MODBUS TCP connection 46 is activated.	○	○	OFF	-	-	R/W	OFF
SM746	MODBUS TCP connection 47 is activated.	○	○	OFF	-	-	R/W	OFF
SM747	MODBUS TCP connection 48 is activated.	○	○	OFF	-	-	R/W	OFF
SM748	MODBUS TCP connection 49 is activated.	○	○	OFF	-	-	R/W	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM749	MODBUS TCP connection 50 is activated.	○	○	OFF	-	-	R/W	OFF
SM750	MODBUS TCP connection 51 is activated.	○	○	OFF	-	-	R/W	OFF
SM751	MODBUS TCP connection 52 is activated.	○	○	OFF	-	-	R/W	OFF
SM752	MODBUS TCP connection 53 is activated.	○	○	OFF	-	-	R/W	OFF
SM753	MODBUS TCP connection 54 is activated.	○	○	OFF	-	-	R/W	OFF
SM754	MODBUS TCP connection 55 is activated.	○	○	OFF	-	-	R/W	OFF
SM755	MODBUS TCP connection 56 is activated.	○	○	OFF	-	-	R/W	OFF
SM756	MODBUS TCP connection 57 is activated.	○	○	OFF	-	-	R/W	OFF
SM757	MODBUS TCP connection 58 is activated.	○	○	OFF	-	-	R/W	OFF
SM758	MODBUS TCP connection 59 is activated.	○	○	OFF	-	-	R/W	OFF
SM759	MODBUS TCP connection 60 is activated.	○	○	OFF	-	-	R/W	OFF
SM760	MODBUS TCP connection 61 is activated.	○	○	OFF	-	-	R/W	OFF
SM761	MODBUS TCP connection 62 is activated.	○	○	OFF	-	-	R/W	OFF
SM762	MODBUS TCP connection 63 is activated.	○	○	OFF	-	-	R/W	OFF
SM763	MODBUS TCP connection 64 is activated.	○	○	OFF	-	-	R/W	OFF
SM764	MODBUS TCP connection 65 is activated.	○	○	OFF	-	-	R/W	OFF
SM765	MODBUS TCP connection 66 is activated.	○	○	OFF	-	-	R/W	OFF
SM766	MODBUS TCP connection 67 is activated.	○	○	OFF	-	-	R/W	OFF
SM767	MODBUS TCP connection 68 is activated.	○	○	OFF	-	-	R/W	OFF
SM768	MODBUS TCP connection 69 is activated.	○	○	OFF	-	-	R/W	OFF
SM769	MODBUS TCP connection 70 is activated.	○	○	OFF	-	-	R/W	OFF
SM770	MODBUS TCP connection 71 is activated.	○	○	OFF	-	-	R/W	OFF
SM771	MODBUS TCP connection 72 is activated.	○	○	OFF	-	-	R/W	OFF
SM772	MODBUS TCP connection 73 is activated.	○	○	OFF	-	-	R/W	OFF
SM773	MODBUS TCP connection 74 is activated.	○	○	OFF	-	-	R/W	OFF
SM774	MODBUS TCP connection 75 is activated.	○	○	OFF	-	-	R/W	OFF
SM775	MODBUS TCP connection 76 is activated.	○	○	OFF	-	-	R/W	OFF
SM776	MODBUS TCP connection 77 is activated.	○	○	OFF	-	-	R/W	OFF
SM777	MODBUS TCP connection 78 is activated.	○	○	OFF	-	-	R/W	OFF
SM778	MODBUS TCP connection 79 is activated.	○	○	OFF	-	-	R/W	OFF
SM779	MODBUS TCP connection 80 is activated.	○	○	OFF	-	-	R/W	OFF

A1

SM	Function	AH-xxEMC		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
		(CPU)	(Module)					
SM780	MODBUS TCP connection 81 is activated.	○	○	OFF	-	-	R/W	OFF
SM781	MODBUS TCP connection 82 is activated.	○	○	OFF	-	-	R/W	OFF
SM782	MODBUS TCP connection 83 is activated.	○	○	OFF	-	-	R/W	OFF
SM783	MODBUS TCP connection 84 is activated.	○	○	OFF	-	-	R/W	OFF
SM784	MODBUS TCP connection 85 is activated.	○	○	OFF	-	-	R/W	OFF
SM785	MODBUS TCP connection 86 is activated.	○	○	OFF	-	-	R/W	OFF
SM786	MODBUS TCP connection 87 is activated.	○	○	OFF	-	-	R/W	OFF
SM787	MODBUS TCP connection 88 is activated.	○	○	OFF	-	-	R/W	OFF
SM788	MODBUS TCP connection 89 is activated.	○	○	OFF	-	-	R/W	OFF
SM789	MODBUS TCP connection 90 is activated.	○	○	OFF	-	-	R/W	OFF
SM790	MODBUS TCP connection 91 is activated.	○	○	OFF	-	-	R/W	OFF
SM791	MODBUS TCP connection 92 is activated.	○	○	OFF	-	-	R/W	OFF
SM792	MODBUS TCP connection 93 is activated.	○	○	OFF	-	-	R/W	OFF
SM793	MODBUS TCP connection 94 is activated.	○	○	OFF	-	-	R/W	OFF
SM794	MODBUS TCP connection 95 is activated.	○	○	OFF	-	-	R/W	OFF
SM795	MODBUS TCP connection 96 is activated.	○	○	OFF	-	-	R/W	OFF
SM796	MODBUS TCP connection 97 is activated.	○	○	OFF	-	-	R/W	OFF
SM797	MODBUS TCP connection 98 is activated.	○	○	OFF	-	-	R/W	OFF
SM798	MODBUS TCP connection 99 is activated.	○	○	OFF	-	-	R/W	OFF
SM799	MODBUS TCP connection 100 is activated.	○	○	OFF	-	-	R/W	OFF
SM800	MODBUS TCP connection 101 is activated.	○	○	OFF	-	-	R/W	OFF
SM801	MODBUS TCP connection 102 is activated.	○	○	OFF	-	-	R/W	OFF
SM802	MODBUS TCP connection 103 is activated.	○	○	OFF	-	-	R/W	OFF
SM803	MODBUS TCP connection 104 is activated.	○	○	OFF	-	-	R/W	OFF
SM804	MODBUS TCP connection 105 is activated.	○	○	OFF	-	-	R/W	OFF
SM805	MODBUS TCP connection 106 is activated.	○	○	OFF	-	-	R/W	OFF
SM806	MODBUS TCP connection 107 is activated.	○	○	OFF	-	-	R/W	OFF
SM807	MODBUS TCP connection 108 is activated.	○	○	OFF	-	-	R/W	OFF
SM808	MODBUS TCP connection 109 is activated.	○	○	OFF	-	-	R/W	OFF
SM809	MODBUS TCP connection 110 is activated.	○	○	OFF	-	-	R/W	OFF
SM810	MODBUS TCP connection 111 is activated.	○	○	OFF	-	-	R/W	OFF

A1

SM	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
			AH-xxEMC(Module)					
SM811	MODBUS TCP connection 112 is activated.	○	○	OFF	–	–	R/W	OFF
SM812	MODBUS TCP connection 113 is activated.	○	○	OFF	–	–	R/W	OFF
SM813	MODBUS TCP connection 114 is activated.	○	○	OFF	–	–	R/W	OFF
SM814	MODBUS TCP connection 115 is activated.	○	○	OFF	–	–	R/W	OFF
SM815	MODBUS TCP connection 116 is activated.	○	○	OFF	–	–	R/W	OFF
SM816	MODBUS TCP connection 117 is activated.	○	○	OFF	–	–	R/W	OFF
SM817	MODBUS TCP connection 118 is activated.	○	○	OFF	–	–	R/W	OFF
SM818	MODBUS TCP connection 119 is activated.	○	○	OFF	–	–	R/W	OFF
SM819	MODBUS TCP connection 120 is activated.	○	○	OFF	–	–	R/W	OFF
SM820	MODBUS TCP connection 121 is activated.	○	○	OFF	–	–	R/W	OFF
SM821	MODBUS TCP connection 122 is activated.	○	○	OFF	–	–	R/W	OFF
SM822	MODBUS TCP connection 123 is activated.	○	○	OFF	–	–	R/W	OFF
SM823	MODBUS TCP connection 124 is activated.	○	○	OFF	–	–	R/W	OFF
SM824	MODBUS TCP connection 125 is activated.	○	○	OFF	–	–	R/W	OFF
SM825	MODBUS TCP connection 126 is activated.	○	○	OFF	–	–	R/W	OFF
SM826	MODBUS TCP connection 127 is activated.	○	○	OFF	–	–	R/W	OFF
SM827	MODBUS TCP connection 128 is activated.	○	○	OFF	–	–	R/W	OFF
SM828	MODBUS TCP connection 1 error	○	○	OFF	–	–	R/W	OFF
SM829	MODBUS TCP connection 2 error	○	○	OFF	–	–	R/W	OFF
SM830	MODBUS TCP connection 3 error	○	○	OFF	–	–	R/W	OFF
SM831	MODBUS TCP connection 4 error	○	○	OFF	–	–	R/W	OFF
SM832	MODBUS TCP connection 5 error	○	○	OFF	–	–	R/W	OFF
SM833	MODBUS TCP connection 6 error	○	○	OFF	–	–	R/W	OFF
SM834	MODBUS TCP connection 7 error	○	○	OFF	–	–	R/W	OFF
SM835	MODBUS TCP connection 8 error	○	○	OFF	–	–	R/W	OFF
SM836	MODBUS TCP connection 9 error	○	○	OFF	–	–	R/W	OFF
SM837	MODBUS TCP connection 10 error	○	○	OFF	–	–	R/W	OFF
SM838	MODBUS TCP connection 11 error	○	○	OFF	–	–	R/W	OFF
SM839	MODBUS TCP connection 12 error	○	○	OFF	–	–	R/W	OFF
SM840	MODBUS TCP connection 13 error	○	○	OFF	–	–	R/W	OFF
SM841	MODBUS TCP connection 14 error	○	○	OFF	–	–	R/W	OFF

SM	Function	AH-xxEMC		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
		(CPU)	(Module)					
SM842	MODBUS TCP connection 15 error	○	○	OFF	-	-	R/W	OFF
SM843	MODBUS TCP connection 16 error	○	○	OFF	-	-	R/W	OFF
SM844	MODBUS TCP connection 17 error	○	○	OFF	-	-	R/W	OFF
SM845	MODBUS TCP connection 18 error	○	○	OFF	-	-	R/W	OFF
SM846	MODBUS TCP connection 19 error	○	○	OFF	-	-	R/W	OFF
SM847	MODBUS TCP connection 20 error	○	○	OFF	-	-	R/W	OFF
SM848	MODBUS TCP connection 21 error	○	○	OFF	-	-	R/W	OFF
SM849	MODBUS TCP connection 22 error	○	○	OFF	-	-	R/W	OFF
SM850	MODBUS TCP connection 23 error	○	○	OFF	-	-	R/W	OFF
SM851	MODBUS TCP connection 24 error	○	○	OFF	-	-	R/W	OFF
SM852	MODBUS TCP connection 25 error	○	○	OFF	-	-	R/W	OFF
SM853	MODBUS TCP connection 26 error	○	○	OFF	-	-	R/W	OFF
SM854	MODBUS TCP connection 27 error	○	○	OFF	-	-	R/W	OFF
SM855	MODBUS TCP connection 28 error	○	○	OFF	-	-	R/W	OFF
SM856	MODBUS TCP connection 29 error	○	○	OFF	-	-	R/W	OFF
SM857	MODBUS TCP connection 30 error	○	○	OFF	-	-	R/W	OFF
SM858	MODBUS TCP connection 31 error	○	○	OFF	-	-	R/W	OFF
SM859	MODBUS TCP connection 32 error	○	○	OFF	-	-	R/W	OFF
SM860	MODBUS TCP connection 33 error	○	○	OFF	-	-	R/W	OFF
SM861	MODBUS TCP connection 34 error	○	○	OFF	-	-	R/W	OFF
SM862	MODBUS TCP connection 35 error	○	○	OFF	-	-	R/W	OFF
SM863	MODBUS TCP connection 36 error	○	○	OFF	-	-	R/W	OFF
SM864	MODBUS TCP connection 37 error	○	○	OFF	-	-	R/W	OFF
SM865	MODBUS TCP connection 38 error	○	○	OFF	-	-	R/W	OFF
SM866	MODBUS TCP connection 39 error	○	○	OFF	-	-	R/W	OFF
SM867	MODBUS TCP connection 40 error	○	○	OFF	-	-	R/W	OFF
SM868	MODBUS TCP connection 41 error	○	○	OFF	-	-	R/W	OFF
SM869	MODBUS TCP connection 42 error	○	○	OFF	-	-	R/W	OFF
SM870	MODBUS TCP connection 43 error	○	○	OFF	-	-	R/W	OFF
SM871	MODBUS TCP connection 44 error	○	○	OFF	-	-	R/W	OFF
SM872	MODBUS TCP connection 45 error	○	○	OFF	-	-	R/W	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
			AH-xxEMC(Module)					
SM873	MODBUS TCP connection 46 error	○	○	OFF	-	-	R/W	OFF
SM874	MODBUS TCP connection 47 error	○	○	OFF	-	-	R/W	OFF
SM875	MODBUS TCP connection 48 error	○	○	OFF	-	-	R/W	OFF
SM876	MODBUS TCP connection 49 error	○	○	OFF	-	-	R/W	OFF
SM877	MODBUS TCP connection 50 error	○	○	OFF	-	-	R/W	OFF
SM878	MODBUS TCP connection 51 error	○	○	OFF	-	-	R/W	OFF
SM879	MODBUS TCP connection 52 error	○	○	OFF	-	-	R/W	OFF
SM880	MODBUS TCP connection 53 error	○	○	OFF	-	-	R/W	OFF
SM881	MODBUS TCP connection 54 error	○	○	OFF	-	-	R/W	OFF
SM882	MODBUS TCP connection 55 error	○	○	OFF	-	-	R/W	OFF
SM883	MODBUS TCP connection 56 error	○	○	OFF	-	-	R/W	OFF
SM884	MODBUS TCP connection 57 error	○	○	OFF	-	-	R/W	OFF
SM885	MODBUS TCP connection 58 error	○	○	OFF	-	-	R/W	OFF
SM886	MODBUS TCP connection 59 error	○	○	OFF	-	-	R/W	OFF
SM887	MODBUS TCP connection 60 error	○	○	OFF	-	-	R/W	OFF
SM888	MODBUS TCP connection 61 error	○	○	OFF	-	-	R/W	OFF
SM889	MODBUS TCP connection 62 error	○	○	OFF	-	-	R/W	OFF
SM890	MODBUS TCP connection 63 error	○	○	OFF	-	-	R/W	OFF
SM891	MODBUS TCP connection 64 error	○	○	OFF	-	-	R/W	OFF
SM892	MODBUS TCP connection 65 error	○	○	OFF	-	-	R/W	OFF
SM893	MODBUS TCP connection 66 error	○	○	OFF	-	-	R/W	OFF
SM894	MODBUS TCP connection 67 error	○	○	OFF	-	-	R/W	OFF
SM895	MODBUS TCP connection 68 error	○	○	OFF	-	-	R/W	OFF
SM896	MODBUS TCP connection 69 error	○	○	OFF	-	-	R/W	OFF
SM897	MODBUS TCP connection 70 error	○	○	OFF	-	-	R/W	OFF
SM898	MODBUS TCP connection 71 error	○	○	OFF	-	-	R/W	OFF
SM899	MODBUS TCP connection 72 error	○	○	OFF	-	-	R/W	OFF
SM900	MODBUS TCP connection 73 error	○	○	OFF	-	-	R/W	OFF
SM901	MODBUS TCP connection 74 error	○	○	OFF	-	-	R/W	OFF
SM902	MODBUS TCP connection 75 error	○	○	OFF	-	-	R/W	OFF
SM903	MODBUS TCP connection 76 error	○	○	OFF	-	-	R/W	OFF

A1

SM	Function	AH-xxEMC		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
		Module)	(CPU)					
SM904	MODBUS TCP connection 77 error	○	○	OFF	-	-	R/W	OFF
SM905	MODBUS TCP connection 78 error	○	○	OFF	-	-	R/W	OFF
SM906	MODBUS TCP connection 79 error	○	○	OFF	-	-	R/W	OFF
SM907	MODBUS TCP connection 80 error	○	○	OFF	-	-	R/W	OFF
SM908	MODBUS TCP connection 81 error	○	○	OFF	-	-	R/W	OFF
SM909	MODBUS TCP connection 82 error	○	○	OFF	-	-	R/W	OFF
SM910	MODBUS TCP connection 83 error	○	○	OFF	-	-	R/W	OFF
SM911	MODBUS TCP connection 84 error	○	○	OFF	-	-	R/W	OFF
SM912	MODBUS TCP connection 85 error	○	○	OFF	-	-	R/W	OFF
SM913	MODBUS TCP connection 86 error	○	○	OFF	-	-	R/W	OFF
SM914	MODBUS TCP connection 87 error	○	○	OFF	-	-	R/W	OFF
SM915	MODBUS TCP connection 88 error	○	○	OFF	-	-	R/W	OFF
SM916	MODBUS TCP connection 89 error	○	○	OFF	-	-	R/W	OFF
SM917	MODBUS TCP connection 90 error	○	○	OFF	-	-	R/W	OFF
SM918	MODBUS TCP connection 91 error	○	○	OFF	-	-	R/W	OFF
SM919	MODBUS TCP connection 92 error	○	○	OFF	-	-	R/W	OFF
SM920	MODBUS TCP connection 93 error	○	○	OFF	-	-	R/W	OFF
SM921	MODBUS TCP connection 94 error	○	○	OFF	-	-	R/W	OFF
SM922	MODBUS TCP connection 95 error	○	○	OFF	-	-	R/W	OFF
SM923	MODBUS TCP connection 96 error	○	○	OFF	-	-	R/W	OFF
SM924	MODBUS TCP connection 97 error	○	○	OFF	-	-	R/W	OFF
SM925	MODBUS TCP connection 98 error	○	○	OFF	-	-	R/W	OFF
SM926	MODBUS TCP connection 99 error	○	○	OFF	-	-	R/W	OFF
SM927	MODBUS TCP connection 100 error	○	○	OFF	-	-	R/W	OFF
SM928	MODBUS TCP connection 101 error	○	○	OFF	-	-	R/W	OFF
SM929	MODBUS TCP connection 102 error	○	○	OFF	-	-	R/W	OFF
SM930	MODBUS TCP connection 103 error	○	○	OFF	-	-	R/W	OFF
SM931	MODBUS TCP connection 104 error	○	○	OFF	-	-	R/W	OFF
SM932	MODBUS TCP connection 105 error	○	○	OFF	-	-	R/W	OFF
SM933	MODBUS TCP connection 106 error	○	○	OFF	-	-	R/W	OFF
SM934	MODBUS TCP connection 107 error	○	○	OFF	-	-	R/W	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
			AH-xxEMC(Module)					
SM935	MODBUS TCP connection 108 error	○	○	OFF	–	–	R/W	OFF
SM936	MODBUS TCP connection 109 error	○	○	OFF	–	–	R/W	OFF
SM937	MODBUS TCP connection 110 error	○	○	OFF	–	–	R/W	OFF
SM938	MODBUS TCP connection 111 error	○	○	OFF	–	–	R/W	OFF
SM939	MODBUS TCP connection 112 error	○	○	OFF	–	–	R/W	OFF
SM940	MODBUS TCP connection 113 error	○	○	OFF	–	–	R/W	OFF
SM941	MODBUS TCP connection 114 error	○	○	OFF	–	–	R/W	OFF
SM942	MODBUS TCP connection 115 error	○	○	OFF	–	–	R/W	OFF
SM943	MODBUS TCP connection 116 error	○	○	OFF	–	–	R/W	OFF
SM944	MODBUS TCP connection 117 error	○	○	OFF	–	–	R/W	OFF
SM945	MODBUS TCP connection 118 error	○	○	OFF	–	–	R/W	OFF
SM946	MODBUS TCP connection 119 error	○	○	OFF	–	–	R/W	OFF
SM947	MODBUS TCP connection 120 error	○	○	OFF	–	–	R/W	OFF
SM948	MODBUS TCP connection 121 error	○	○	OFF	–	–	R/W	OFF
SM949	MODBUS TCP connection 122 error	○	○	OFF	–	–	R/W	OFF
SM950	MODBUS TCP connection 123 error	○	○	OFF	–	–	R/W	OFF
SM951	MODBUS TCP connection 124 error	○	○	OFF	–	–	R/W	OFF
SM952	MODBUS TCP connection 125 error	○	○	OFF	–	–	R/W	OFF
SM953	MODBUS TCP connection 126 error	○	○	OFF	–	–	R/W	OFF
SM954	MODBUS TCP connection 127 error	○	○	OFF	–	–	R/W	OFF
SM955	MODBUS TCP connection 128 error	○	○	OFF	–	–	R/W	OFF
SM1000	It is the Ethernet setting flag. When SM1000 is ON, the data in SR1000~SR1006 is written into the flash memory.	○	○	-	–	–	R/W	OFF
SM1090	The TCP connection is busy.	○	○	OFF	–	–	R	OFF
SM1091	The UDP connection is busy.	○	○	OFF	–	–	R	OFF
SM1106	Basic management–Ethernet connection error	○	○	OFF	–	–	R	OFF
SM1107	Basic management of Ethernet–Basic setting error	○	○	OFF	–	–	R	OFF
SM1108	Basic management of Ethernet–Filter setting error	○	○	OFF	–	–	R	OFF
SM1109	Basic management of the TCP/UDP socket–The local port is already used.	○	○	OFF	–	–	R	OFF

A1

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1752	COM2-MODBUS Slave 1 data exchanging	○	×	OFF	-	-	R	OFF
SM1753	COM2-MODBUS Slave 2 data exchanging	○	×	OFF	-	-	R	OFF
SM1754	COM2-MODBUS Slave 3 data exchanging	○	×	OFF	-	-	R	OFF
SM1755	COM2-MODBUS Slave 4 data exchanging	○	×	OFF	-	-	R	OFF
SM1756	COM2-MODBUS Slave 5 data exchanging	○	×	OFF	-	-	R	OFF
SM1757	COM2-MODBUS Slave 6 data exchanging	○	×	OFF	-	-	R	OFF
SM1758	COM2-MODBUS Slave 7 data exchanging	○	×	OFF	-	-	R	OFF
SM1759	COM2-MODBUS Slave 8 data exchanging	○	×	OFF	-	-	R	OFF
SM1760	COM2-MODBUS Slave 9 data exchanging	○	×	OFF	-	-	R	OFF
SM1761	COM2-MODBUS Slave 10 data exchanging	○	×	OFF	-	-	R	OFF
SM1762	COM2-MODBUS Slave 11 data exchanging	○	×	OFF	-	-	R	OFF
SM1763	COM2-MODBUS Slave 12 data exchanging	○	×	OFF	-	-	R	OFF
SM1764	COM2-MODBUS Slave 13 data exchanging	○	×	OFF	-	-	R	OFF
SM1765	COM2-MODBUS Slave 14 data exchanging	○	×	OFF	-	-	R	OFF
SM1766	COM2-MODBUS Slave 15 data exchanging	○	×	OFF	-	-	R	OFF
SM1767	COM2-MODBUS Slave 16 data exchanging	○	×	OFF	-	-	R	OFF
SM1768	COM2-MODBUS Slave 17 data exchanging	○	×	OFF	-	-	R	OFF
SM1769	COM2-MODBUS Slave 18 data exchanging	○	×	OFF	-	-	R	OFF
SM1770	COM2-MODBUS Slave 19 data exchanging	○	×	OFF	-	-	R	OFF
SM1771	COM2-MODBUS Slave 20 data exchanging	○	×	OFF	-	-	R	OFF
SM1772	COM2-MODBUS Slave 21 data exchanging	○	×	OFF	-	-	R	OFF
SM1773	COM2-MODBUS Slave 22 data exchanging	○	×	OFF	-	-	R	OFF
SM1774	COM2-MODBUS Slave 23 data exchanging	○	×	OFF	-	-	R	OFF
SM1775	COM2-MODBUS Slave 24 data exchanging	○	×	OFF	-	-	R	OFF
SM1776	COM2-MODBUS Slave 25 data exchanging	○	×	OFF	-	-	R	OFF
SM1777	COM2-MODBUS Slave 26 data exchanging	○	×	OFF	-	-	R	OFF
SM1778	COM2-MODBUS Slave 27 data exchanging	○	×	OFF	-	-	R	OFF
SM1779	COM2-MODBUS Slave 28 data exchanging	○	×	OFF	-	-	R	OFF
SM1780	COM2-MODBUS Slave 29 data exchanging	○	×	OFF	-	-	R	OFF
SM1781	COM2-MODBUS Slave 30 data exchanging	○	×	OFF	-	-	R	OFF
SM1782	COM2-MODBUS Slave 31 data exchanging	○	×	OFF	-	-	R	OFF

A1

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1783	COM2-MODBUS Slave 32 data exchanging	○	×	OFF	-	-	R	OFF
SM1784	COM2-MODBUS Slave 1 reading error	○	×	OFF	-	-	R	OFF
SM1785	COM2-MODBUS Slave 2 reading error	○	×	OFF	-	-	R	OFF
SM1786	COM2-MODBUS Slave 3 reading error	○	×	OFF	-	-	R	OFF
SM1787	COM2-MODBUS Slave 4 reading error	○	×	OFF	-	-	R	OFF
SM1788	COM2-MODBUS Slave 5 reading error	○	×	OFF	-	-	R	OFF
SM1789	COM2-MODBUS Slave 6 reading error	○	×	OFF	-	-	R	OFF
SM1790	COM2-MODBUS Slave 7 reading error	○	×	OFF	-	-	R	OFF
SM1791	COM2-MODBUS Slave 8 reading error	○	×	OFF	-	-	R	OFF
SM1792	COM2-MODBUS Slave 9 reading error	○	×	OFF	-	-	R	OFF
SM1793	COM2-MODBUS Slave 10 reading error	○	×	OFF	-	-	R	OFF
SM1794	COM2-MODBUS Slave 11 reading error	○	×	OFF	-	-	R	OFF
SM1795	COM2-MODBUS Slave 12 reading error	○	×	OFF	-	-	R	OFF
SM1796	COM2-MODBUS Slave 13 reading error	○	×	OFF	-	-	R	OFF
SM1797	COM2-MODBUS Slave 14 reading error	○	×	OFF	-	-	R	OFF
SM1798	COM2-MODBUS Slave 15 reading error	○	×	OFF	-	-	R	OFF
SM1799	COM2-MODBUS Slave 16 reading error	○	×	OFF	-	-	R	OFF
SM1800	COM2-MODBUS Slave 17 reading error	○	×	OFF	-	-	R	OFF
SM1801	COM2-MODBUS Slave 18 reading error	○	×	OFF	-	-	R	OFF
SM1802	COM2-MODBUS Slave 19 reading error	○	×	OFF	-	-	R	OFF
SM1803	COM2-MODBUS Slave 20 reading error	○	×	OFF	-	-	R	OFF
SM1804	COM2-MODBUS Slave 21 reading error	○	×	OFF	-	-	R	OFF
SM1805	COM2-MODBUS Slave 22 reading error	○	×	OFF	-	-	R	OFF
SM1806	COM2-MODBUS Slave 23 reading error	○	×	OFF	-	-	R	OFF
SM1807	COM2-MODBUS Slave 24 reading error	○	×	OFF	-	-	R	OFF
SM1808	COM2-MODBUS Slave 25 reading error	○	×	OFF	-	-	R	OFF
SM1809	COM2-MODBUS Slave 26 reading error	○	×	OFF	-	-	R	OFF
SM1810	COM2-MODBUS Slave 27 reading error	○	×	OFF	-	-	R	OFF
SM1811	COM2-MODBUS Slave 28 reading error	○	×	OFF	-	-	R	OFF
SM1812	COM2-MODBUS Slave 29 reading error	○	×	OFF	-	-	R	OFF
SM1813	COM2-MODBUS Slave 30 reading error	○	×	OFF	-	-	R	OFF

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1814	COM2-MODBUS Slave 31 reading error	○	×	OFF	-	-	R	OFF
SM1815	COM2-MODBUS Slave 32 reading error	○	×	OFF	-	-	R	OFF
SM1816	COM2-MODBUS Slave 1 writing error	○	×	OFF	-	-	R	OFF
SM1817	COM2-MODBUS Slave 2 writing error	○	×	OFF	-	-	R	OFF
SM1818	COM2-MODBUS Slave 3 writing error	○	×	OFF	-	-	R	OFF
SM1819	COM2-MODBUS Slave 4 writing error	○	×	OFF	-	-	R	OFF
SM1820	COM2-MODBUS Slave 5 writing error	○	×	OFF	-	-	R	OFF
SM1821	COM2-MODBUS Slave 6 writing error	○	×	OFF	-	-	R	OFF
SM1822	COM2-MODBUS Slave 7 writing error	○	×	OFF	-	-	R	OFF
SM1823	COM2-MODBUS Slave 8 writing error	○	×	OFF	-	-	R	OFF
SM1824	COM2-MODBUS Slave 9 writing error	○	×	OFF	-	-	R	OFF
SM1825	COM2-MODBUS Slave 10 writing error	○	×	OFF	-	-	R	OFF
SM1826	COM2-MODBUS Slave 11 writing error	○	×	OFF	-	-	R	OFF
SM1827	COM2-MODBUS Slave 12 writing error	○	×	OFF	-	-	R	OFF
SM1828	COM2-MODBUS Slave 13 writing error	○	×	OFF	-	-	R	OFF
SM1829	COM2-MODBUS Slave 14 writing error	○	×	OFF	-	-	R	OFF
SM1830	COM2-MODBUS Slave 15 writing error	○	×	OFF	-	-	R	OFF
SM1831	COM2-MODBUS Slave 16 writing error	○	×	OFF	-	-	R	OFF
SM1832	COM2-MODBUS Slave 17 writing error	○	×	OFF	-	-	R	OFF
SM1833	COM2-MODBUS Slave 18 writing error	○	×	OFF	-	-	R	OFF
SM1834	COM2-MODBUS Slave 19 writing error	○	×	OFF	-	-	R	OFF
SM1835	COM2-MODBUS Slave 20 writing error	○	×	OFF	-	-	R	OFF
SM1836	COM2-MODBUS Slave 21 writing error	○	×	OFF	-	-	R	OFF
SM1837	COM2-MODBUS Slave 22 writing error	○	×	OFF	-	-	R	OFF
SM1838	COM2-MODBUS Slave 23 writing error	○	×	OFF	-	-	R	OFF
SM1839	COM2-MODBUS Slave 24 writing error	○	×	OFF	-	-	R	OFF
SM1840	COM2-MODBUS Slave 25 writing error	○	×	OFF	-	-	R	OFF
SM1841	COM2-MODBUS Slave 26 writing error	○	×	OFF	-	-	R	OFF
SM1842	COM2-MODBUS Slave 27 writing error	○	×	OFF	-	-	R	OFF
SM1843	COM2-MODBUS Slave 28 writing error	○	×	OFF	-	-	R	OFF
SM1844	COM2-MODBUS Slave 29 writing error	○	×	OFF	-	-	R	OFF

A1

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1845	COM2-MODBUS Slave 30 writing error	○	×	OFF	-	-	R	OFF
SM1846	COM2-MODBUS Slave 31 writing error	○	×	OFF	-	-	R	OFF
SM1847	COM2-MODBUS Slave 32 writing error	○	×	OFF	-	-	R	OFF
SM1848	COM2-MODBUS Slave 1 reading completed	○	×	OFF	-	-	R	OFF
SM1849	COM2-MODBUS Slave 2 reading completed	○	×	OFF	-	-	R	OFF
SM1850	COM2-MODBUS Slave 3 reading completed	○	×	OFF	-	-	R	OFF
SM1851	COM2-MODBUS Slave 4 reading completed	○	×	OFF	-	-	R	OFF
SM1852	COM2-MODBUS Slave 5 reading completed	○	×	OFF	-	-	R	OFF
SM1853	COM2-MODBUS Slave 6 reading completed	○	×	OFF	-	-	R	OFF
SM1854	COM2-MODBUS Slave 7 reading completed	○	×	OFF	-	-	R	OFF
SM1855	COM2-MODBUS Slave 8 reading completed	○	×	OFF	-	-	R	OFF
SM1856	COM2-MODBUS Slave 9 reading completed	○	×	OFF	-	-	R	OFF
SM1857	COM2-MODBUS Slave 10 reading completed	○	×	OFF	-	-	R	OFF
SM1858	COM2-MODBUS Slave 11 reading completed	○	×	OFF	-	-	R	OFF
SM1859	COM2-MODBUS Slave 12 reading completed	○	×	OFF	-	-	R	OFF
SM1860	COM2-MODBUS Slave 13 reading completed	○	×	OFF	-	-	R	OFF
SM1861	COM2-MODBUS Slave 14 reading completed	○	×	OFF	-	-	R	OFF
SM1862	COM2-MODBUS Slave 15 reading completed	○	×	OFF	-	-	R	OFF
SM1863	COM2-MODBUS Slave 16 reading completed	○	×	OFF	-	-	R	OFF
SM1864	COM2-MODBUS Slave 17 reading completed	○	×	OFF	-	-	R	OFF
SM1865	COM2-MODBUS Slave 18 reading completed	○	×	OFF	-	-	R	OFF
SM1866	COM2-MODBUS Slave 19 reading completed	○	×	OFF	-	-	R	OFF
SM1867	COM2-MODBUS Slave 20 reading completed	○	×	OFF	-	-	R	OFF
SM1868	COM2-MODBUS Slave 21 reading completed	○	×	OFF	-	-	R	OFF
SM1869	COM2-MODBUS Slave 22 reading completed	○	×	OFF	-	-	R	OFF
SM1870	COM2-MODBUS Slave 23 reading completed	○	×	OFF	-	-	R	OFF
SM1871	COM2-MODBUS Slave 24 reading completed	○	×	OFF	-	-	R	OFF
SM1872	COM2-MODBUS Slave 25 reading completed	○	×	OFF	-	-	R	OFF
SM1873	COM2-MODBUS Slave 26 reading completed	○	×	OFF	-	-	R	OFF
SM1874	COM2-MODBUS Slave 27 reading completed	○	×	OFF	-	-	R	OFF
SM1875	COM2-MODBUS Slave 28 reading completed	○	×	OFF	-	-	R	OFF

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1876	COM2-MODBUS Slave 29 reading completed	○	×	OFF	-	-	R	OFF
SM1877	COM2-MODBUS Slave 30 reading completed	○	×	OFF	-	-	R	OFF
SM1878	COM2-MODBUS Slave 31 reading completed	○	×	OFF	-	-	R	OFF
SM1879	COM2-MODBUS Slave 32 reading completed	○	×	OFF	-	-	R	OFF
SM1880	COM2-MODBUS Slave 1 writing completed	○	×	OFF	-	-	R	OFF
SM1881	COM2-MODBUS Slave 2 writing completed	○	×	OFF	-	-	R	OFF
SM1882	COM2-MODBUS Slave 3 writing completed	○	×	OFF	-	-	R	OFF
SM1883	COM2-MODBUS Slave 4 writing completed	○	×	OFF	-	-	R	OFF
SM1884	COM2-MODBUS Slave 5 writing completed	○	×	OFF	-	-	R	OFF
SM1885	COM2-MODBUS Slave 6 writing completed	○	×	OFF	-	-	R	OFF
SM1886	COM2-MODBUS Slave 7 writing completed	○	×	OFF	-	-	R	OFF
SM1887	COM2-MODBUS Slave 8 writing completed	○	×	OFF	-	-	R	OFF
SM1888	COM2-MODBUS Slave 9 writing completed	○	×	OFF	-	-	R	OFF
SM1889	COM2-MODBUS Slave 10 writing completed	○	×	OFF	-	-	R	OFF
SM1890	COM2-MODBUS Slave 11 writing completed	○	×	OFF	-	-	R	OFF
SM1891	COM2-MODBUS Slave 12 writing completed	○	×	OFF	-	-	R	OFF
SM1892	COM2-MODBUS Slave 13 writing completed	○	×	OFF	-	-	R	OFF
SM1893	COM2-MODBUS Slave 14 writing completed	○	×	OFF	-	-	R	OFF
SM1894	COM2-MODBUS Slave 15 writing completed	○	×	OFF	-	-	R	OFF
SM1895	COM2-MODBUS Slave 16 writing completed	○	×	OFF	-	-	R	OFF
SM1896	COM2-MODBUS Slave 17 writing completed	○	×	OFF	-	-	R	OFF
SM1897	COM2-MODBUS Slave 18 writing completed	○	×	OFF	-	-	R	OFF
SM1898	COM2-MODBUS Slave 19 writing completed	○	×	OFF	-	-	R	OFF
SM1899	COM2-MODBUS Slave 20 writing completed	○	×	OFF	-	-	R	OFF
SM1900	COM2-MODBUS Slave 21 writing completed	○	×	OFF	-	-	R	OFF
SM1901	COM2-MODBUS Slave 22 writing completed	○	×	OFF	-	-	R	OFF
SM1902	COM2-MODBUS Slave 23 writing completed	○	×	OFF	-	-	R	OFF
SM1903	COM2-MODBUS Slave 24 writing completed	○	×	OFF	-	-	R	OFF
SM1904	COM2-MODBUS Slave 25 writing completed	○	×	OFF	-	-	R	OFF
SM1905	COM2-MODBUS Slave 26 writing completed	○	×	OFF	-	-	R	OFF
SM1906	COM2-MODBUS Slave 27 writing completed	○	×	OFF	-	-	R	OFF

A1

Appendix 1. Special Auxiliary Relays Table

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1907	COM2-MODBUS Slave 28 writing completed	○	×	OFF	-	-	R	OFF
SM1908	COM2-MODBUS Slave 29 writing completed	○	×	OFF	-	-	R	OFF
SM1909	COM2-MODBUS Slave 30 writing completed	○	×	OFF	-	-	R	OFF
SM1910	COM2-MODBUS Slave 31 writing completed	○	×	OFF	-	-	R	OFF
SM1911	COM2-MODBUS Slave 32 writing completed	○	×	OFF	-	-	R	OFF
SM1912	COM2-MODBUS 同步讀/寫功能	○	×	OFF	-	-	R/W	OFF
SM1913	COM2-MODBUS Slave 1 is activated.	○	×	OFF	-	-	R/W	OFF
SM1914	COM2-MODBUS Slave 2 is activated.	○	×	OFF	-	-	R/W	OFF
SM1915	COM2-MODBUS Slave 3 is activated.	○	×	OFF	-	-	R/W	OFF
SM1916	COM2-MODBUS Slave 4 is activated.	○	×	OFF	-	-	R/W	OFF
SM1917	COM2-MODBUS Slave 5 is activated.	○	×	OFF	-	-	R/W	OFF
SM1918	COM2-MODBUS Slave 6 is activated.	○	×	OFF	-	-	R/W	OFF
SM1919	COM2-MODBUS Slave 7 is activated.	○	×	OFF	-	-	R/W	OFF
SM1920	COM2-MODBUS Slave 8 is activated.	○	×	OFF	-	-	R/W	OFF
SM1921	COM2-MODBUS Slave 9 is activated.	○	×	OFF	-	-	R/W	OFF
SM1922	COM2-MODBUS Slave 10 is activated.	○	×	OFF	-	-	R/W	OFF
SM1923	COM2-MODBUS Slave 11 is activated.	○	×	OFF	-	-	R/W	OFF
SM1924	COM2-MODBUS Slave 12 is activated.	○	×	OFF	-	-	R/W	OFF
SM1925	COM2-MODBUS Slave 13 is activated.	○	×	OFF	-	-	R/W	OFF
SM1926	COM2-MODBUS Slave 14 is activated.	○	×	OFF	-	-	R/W	OFF
SM1927	COM2-MODBUS Slave 15 is activated.	○	×	OFF	-	-	R/W	OFF
SM1928	COM2-MODBUS Slave 16 is activated.	○	×	OFF	-	-	R/W	OFF
SM1929	COM2-MODBUS Slave 17 is activated.	○	×	OFF	-	-	R/W	OFF
SM1930	COM2-MODBUS Slave 18 is activated.	○	×	OFF	-	-	R/W	OFF
SM1931	COM2-MODBUS Slave 19 is activated.	○	×	OFF	-	-	R/W	OFF
SM1932	COM2-MODBUS Slave 20 is activated.	○	×	OFF	-	-	R/W	OFF
SM1933	COM2-MODBUS Slave 21 is activated.	○	×	OFF	-	-	R/W	OFF
SM1934	COM2-MODBUS Slave 22 is activated.	○	×	OFF	-	-	R/W	OFF
SM1935	COM2-MODBUS Slave 23 is activated.	○	×	OFF	-	-	R/W	OFF
SM1936	COM2-MODBUS Slave 24 is activated.	○	×	OFF	-	-	R/W	OFF
SM1937	COM2-MODBUS Slave 25 is activated.	○	×	OFF	-	-	R/W	OFF

A1

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SM1938	COM2-MODBUS Slave 26 is activated.	○	×	OFF	-	-	R/W	OFF
SM1939	COM2-MODBUS Slave 27 is activated.	○	×	OFF	-	-	R/W	OFF
SM1940	COM2-MODBUS Slave 28 is activated.	○	×	OFF	-	-	R/W	OFF
SM1941	COM2-MODBUS Slave 29 is activated.	○	×	OFF	-	-	R/W	OFF
SM1942	COM2-MODBUS Slave 30 is activated.	○	×	OFF	-	-	R/W	OFF
SM1943	COM2-MODBUS Slave 31 is activated.	○	×	OFF	-	-	R/W	OFF
SM1944	COM2-MODBUS Slave 32 is activated.	○	×	OFF	-	-	R/W	OFF
*SM2000	The data is sent by using EMDRW 1.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2001	The PLC waits for the data after EMDRW 1 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2002	The data is received by using EMDRW 1.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2003	An error occurs when EMDRW 1 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2004	There is a timeout after EMDRW 1 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2005	The connection is closed after EMDRW 1 is used.	○	○	ON	ON	ON	R	ON
*SM2006	The data is sent by using EMDRW 2.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2007	The PLC waits for the data after EMDRW 2 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2008	The data is received by using EMDRW 2.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2009	An error occurs when EMDRW 2 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2010	There is a timeout after EMDRW 2 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2011	The connection is closed after EMDRW 2 is used.	○	○	ON	ON	ON	R	ON
*SM2012	The data is sent by using EMDRW 3.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2013	The PLC waits for the data after EMDRW 3 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2014	The data is received by using EMDRW 3.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2015	An error occurs when EMDRW 3 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2016	There is a timeout after EMDRW 3 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2017	The connection is closed after EMDRW 3 is used.	○	○	ON	ON	ON	R	ON
*SM2018	The data is sent by using EMDRW 4.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2019	The PLC waits for the data after EMDRW 4 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2020	The data is received by using EMDRW 4.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2021	An error occurs when EMDRW 4 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2022	There is a timeout after EMDRW 4 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2023	The connection is closed after EMDRW 4 is used.	○	○	ON	ON	ON	R	ON

A1

SM	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SM2024	The data is sent by using EMDRW 5.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2025	The PLC waits for the data after EMDRW 5 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2026	The data is received by using EMDRW 5.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2027	An error occurs when EMDRW 5 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2028	There is a timeout after EMDRW 5 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2029	The connection is closed after EMDRW 5 is used.	○	○	ON	ON	ON	R	ON
*SM2030	The data is sent by using EMDRW 6.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2031	The PLC waits for the data after EMDRW 6 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2032	The data is received by using EMDRW 6.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2033	An error occurs when EMDRW 6 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2034	There is a timeout after EMDRW 6 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2035	The connection is closed after EMDRW 6 is used.	○	○	ON	ON	ON	R	ON
*SM2036	The data is sent by using EMDRW 7.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2037	The PLC waits for the data after EMDRW 7 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2038	The data is received by using EMDRW 7.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2039	An error occurs when EMDRW 7 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2040	There is a timeout after EMDRW 7 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2041	The connection is closed after EMDRW 7 is used.	○	○	ON	ON	ON	R	ON
*SM2042	The data is sent by using EMDRW 8.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2043	The PLC waits for the data after EMDRW 8 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2044	The data is received by using EMDRW 8.	○	○	OFF	OFF	OFF	R/W	OFF
*SM2045	An error occurs when EMDRW 8 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2046	There is a timeout after EMDRW 8 is used.	○	○	OFF	OFF	OFF	R	OFF
*SM2047	The connection is closed after EMDRW 8 is used.	○	○	ON	ON	ON	R	ON

Note: As to the SM numbers marked “*”, refer to **A.3 Additional Remarks on SM and SR**.

A.1.1. Refresh Time of Special Auxiliary Relay

Special auxiliary relay	Refresh time
SM0~SM1	The system automatically sets the flag to ON and resets it to OFF. The flag is automatically set to ON when there is an operation error.
SM5	The system automatically sets SM5 to ON and resets it to OFF. (1) SM5 is refreshed when the program is rewritten in the PLC. (2) SM5 is refreshed when the PLC is supplied with power and starts to run for the first time.

Special auxiliary relay	Refresh time
SM8	The system automatically sets SM8 to ON and resets it to OFF. SM8 is automatically set to ON when there is a watchdog timer error.
SM9	The system automatically sets SM9 to ON and resets it to OFF. SM9 is automatically set to ON when there is a system error.
SM10	The system automatically sets SM10 to ON and resets it to OFF. SM10 is automatically set to ON when there is an I/O bus error.
SM22, SM23, and SM24	Users set the flag to ON, and the system automatically resets it to OFF. The log is cleared when the flag is ON.
SM25–SM26	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed every scan cycle.
SM96–SM97	Users set the flag to ON. After the data is sent, the system automatically resets the flag to OFF.
SM98–SM99	The system automatically sets the flag to ON and resets it to OFF. The flag is automatically set to ON when the command is sent.
SM100–SM101	The system automatically sets the flag to ON, and users reset it to OFF. The flag is set to ON when the command is received.
SM102–SM103	The system automatically sets the flag to ON, and users reset it to OFF. The flag is automatically set to ON when the command received is wrong.
SM104–SM105	The system automatically sets the flag to ON, and users reset it to OFF. The flag is set to ON when there is a receive timeout.
SM106–SM107	Users set the flag to ON and reset it to OFF. ON: The 8-bit mode OFF: The 16-bit mode
SM108–SM109	Users set the flag to ON and reset it to OFF.
SM204–SM205	Users set the flag to ON, and the system automatically resets it to OFF. ON: Clearing the non-latched/latched areas
SM206	Users set SM206 to ON and reset it to OFF. ON: Inhibiting all output
SM209	Users set SM209 to ON, and the system automatically resets it to OFF. ON: The communication protocol of COM1 changes.
SM210	Users set SM210 to ON and reset it to OFF. ON: The RTU mode
SM211	Users set SM211 to ON, and the system automatically resets it to OFF. ON: The communication protocol of COM2 changes.
SM212	Users set SM212 to ON and reset it to OFF. ON: The RTU mode
SM215	Users set SM215 to ON and reset it to OFF. ON: The PLC runs.
SM220	Users set SM220 to ON and reset it to OFF. ON: Calibrating the real-time clock within ± 30 seconds
SM400–SM401	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed every scan cycle.
SM402–SM403	The system automatically sets the flag to ON and resets it to OFF.

Appendix 1. Special Auxiliary Relays Table

Special auxiliary relay	Refresh time
	The flag is refreshed whenever the instruction END is executed.
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 500 seconds.
SM408	The system automatically sets SM408 to ON and resets it to OFF. SM408 is refreshed every second.
SM409	The system automatically sets SM409 to ON and resets it to OFF. SM409 is refreshed every n seconds, n is specified by SR409.
SM410	The system automatically sets SM410 to ON and resets it to OFF. SM410 is refreshed every n seconds, and n is specified by SR410.
SM450	The system automatically sets SM450 to ON and resets it to OFF. ON: The memory card is inserted into the PLC.
SM451	Users set SM451 to ON and reset it to OFF. ON: The memory card is write protected.
SM452	The system automatically sets SM452 to ON and resets it to OFF. ON: The data in the memory card is being accessed.
SM453	The system automatically sets SM453 to ON and resets it to OFF. ON: An error occurs during the operation of the memory card.
SM600~SM602	The system automatically sets the flag to ON and resets it to OFF. The flag is refreshed when the instruction is executed.
SM603	The system automatically sets SM603 to ON and resets it to OFF. SM603 is refreshed when the instruction SORT is executed.
SM604	Users set SM604 to ON and reset it to OFF. SM604 is refreshed when the instruction SORT whose mode is the descending order is executed.
SM605	Users set SM605 to ON and reset it to OFF.
SM606	Users set SM606 to ON and reset it to OFF. ON: The 8-bit mode
SM607	Users set SM607 to ON or OFF.
SM608	SM608 is refreshed when the instruction is executed.
SM609	Users set the flag to ON or OFF.
SM610~SM611	The flag is refreshed when the instruction is executed.
SM612~SM613	Users set the flag to ON or OFF.
SM614	SM614 is refreshed when the instruction is executed.
SM615~SM617	Users set the flag to ON or OFF.
SM618	SM618 is refreshed when the instruction is executed.

A1

Special auxiliary relay	Refresh time
SM619	SM619 is refreshed when EI or DI is executed.
SM620	SM620 is refreshed when the instruction CMPT is executed.
SM621~SM686	Users set the flag to ON or OFF.
SM687	SM687 is refreshed when the instruction RAMP is executed.
SM688	SM688 is refreshed when the instruction INCD is executed.
SM690~SM691	Users set the flag to ON or OFF.
SM692	SM692 is refreshed when the instruction HKY is executed.
SM693	SM693 is refreshed when the instruction SEGL is executed.
SM694	SM694 is refreshed when the instruction DSW is executed.
SM695 and SM1000	Users set the flag to ON or OFF.
SM1090	SM1090 is ON when the TCP connection is busy.
SM1091	SM1091 is ON when the UDP connection is busy.
SM1106	SM1106 is ON when the PHY initialization fails.
SM1107	SM1107 is ON when the IP address, the netmask address, and the gateway address are set incorrectly.
SM1108	SM1108 is ON when there is a filter setting error.
SM1109	SM1109 is ON when the function of the socket is enabled and the same port is used.
SM1752~SM1911	Updates on every scan.
SM1912	Users set SM19121 to ON and reset it to OFF.
SM1913~SM1944	Updates on every scan.
SM2000	Users set SM2000 to ON and reset it to OFF.
SM2001~SM2002	The flag is refreshed when the instruction EMDRW is executed.
SM2003	SM2003 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2004	SM2004 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2005	Users set SM2005 to ON and reset it to OFF.
SM2006~SM2007	The flag is refreshed when the instruction EMDRW is executed.
SM2008	SM2008 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2009	SM2009 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2010	Users set SM2010 to ON and reset it to OFF.
SM2011~SM2012	The flag is refreshed when the instruction EMDRW is executed.
SM2013	SM2013 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2014	SM2014 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2015	Users set SM2015 to ON and reset it to OFF.
SM2016~SM2017	The flag is refreshed when the instruction EMDRW is executed.

A1

Appendix 1. Special Auxiliary Relays Table

Special auxiliary relay	Refresh time
SM2018	SM2018 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2019	SM2019 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2020	Users set SM2020 to ON and reset it to OFF.
SM2021~SM2022	The flag is refreshed when the instruction EMDRW is executed.
SM2023	SM2023 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2024	SM2024 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2025	Users set SM2025 to ON and reset it to OFF.
SM2026~SM2027	The flag is refreshed when the instruction EMDRW is executed.
SM2028	SM2028 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2029	SM2029 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2030	Users set SM2030 to ON and reset it to OFF.
SM2031~SM2032	The flag is refreshed when the instruction EMDRW is executed.
SM2033	SM2033 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2034	SM2034 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2035	Users set SM2035 to ON and reset it to OFF.
SM2036~SM2037	The flag is refreshed when the instruction EMDRW is executed.
SM2038	SM2038 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2039	SM2039 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2040	Users set SM2040 to ON and reset it to OFF.
SM2041~SM2042	The flag is refreshed when the instruction EMDRW is executed.
SM2043	SM2043 is refreshed when the instruction EMDRW is executed and an error occurs.
SM2044	SM2044 is refreshed when the instruction EMDRW is executed and there is a response timeout.
SM2045	Users set SM2045 to ON and reset it to OFF.
SM2046~SM2047	The flag is refreshed when the instruction EMDRW is executed.

A1

A.2. Special Data Registers Table (SR)

Every special data register has its definition and specific function. The system statuses and the error messages are stored in the special data registers. Besides, the special data registers can be used to monitor the system statuses. The special data registers and their functions are listed as follows. **For more information regarding the SR numbers marked “*”, refer to A.3 Additional Remarks on SM and SR.** The “R” in the attribute column indicates that the special data register can read the data, whereas the “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 data register 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.

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SR0	Error-detecting code of the PLC operation error	○	○	0	0	–	R	0
SR1	The address of the operation error is locked.	○	○	0	0	–	R	0
SR2		○	○	0	0	–	R	0
SR4	Error-detecting code of the grammar check error	○	○	0	0	–	R	0
SR5	Address of the instruction/operand check error	○	○	0	0	–	R	0
SR6		○	○	0	0	–	R	0
SR8	Step address at which the watchdog timer is ON	○	○	0	–	–	R	0
*SR40	Number of error logs	○	○	–	–	–	R	0
*SR41	Error log pointer	○	○	–	–	–	R	0
*SR42	Error log 1: The rack number and the slot number	○	○	–	–	–	R	0
*SR43	Error log 1: The module ID	○	○	–	–	–	R	0
*SR44	Error log 1: The error code	○	○	–	–	–	R	0
*SR45	Error log 1: The year and the month	○	○	–	–	–	R	0
*SR46	Error log 1: The day and the hour	○	○	–	–	–	R	0
*SR47	Error log 1: The minute and the second	○	○	–	–	–	R	0
*SR48	Error log 2: The rack number and the slot number	○	○	–	–	–	R	0
*SR49	Error log 2: The module ID	○	○	–	–	–	R	0
*SR50	Error log 2: The error code	○	○	–	–	–	R	0
*SR51	Error log 2: The year and the month	○	○	–	–	–	R	0
*SR52	Error log 2: The day and the hour	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR53	Error log 2: The minute and the second	○	○	–	–	–	R	0
*SR54	Error log 3: The rack number and the slot number	○	○	–	–	–	R	0
*SR55	Error log 3: The module ID	○	○	–	–	–	R	0
*SR56	Error log 3: The error code	○	○	–	–	–	R	0
*SR57	Error log 3: The year and the month	○	○	–	–	–	R	0
*SR58	Error log 3: The day and the hour	○	○	–	–	–	R	0
*SR59	Error log 3: The minute and the second	○	○	–	–	–	R	0
*SR60	Error log 4: The rack number and the slot number	○	○	–	–	–	R	0
*SR61	Error log 4: The module ID	○	○	–	–	–	R	0
*SR62	Error log 4: The error code	○	○	–	–	–	R	0
*SR63	Error log 4: The year and the month	○	○	–	–	–	R	0
*SR64	Error log 4: The day and the hour	○	○	–	–	–	R	0
*SR65	Error log 4: The minute and the second	○	○	–	–	–	R	0
*SR66	Error log 4: The rack number and the slot number	○	○	–	–	–	R	0
SR67	Error log 5: The rack number and the slot number	○	○	–	–	–	R	0
SR68	Error log 5: The module ID	○	○	–	–	–	R	0
SR69	Error log 5: The error code	○	○	–	–	–	R	0
SR70	Error log 5: The year and the month	○	○	–	–	–	R	0
SR71	Error log 5: The day and the hour	○	○	–	–	–	R	0
SR72	Error log 6: The rack number and the slot number	○	○	–	–	–	R	0
*SR73	Error log 6: The module ID	○	○	–	–	–	R	0
*SR74	Error log 6: The error code	○	○	–	–	–	R	0
*SR75	Error log 6: The year and the month	○	○	–	–	–	R	0
*SR76	Error log 6: The day and the hour	○	○	–	–	–	R	0
*SR77	Error log 6: The minute and the second	○	○	–	–	–	R	0
*SR78	Error log 7: The rack number and the slot number	○	○	–	–	–	R	0
*SR79	Error log 7: The module ID	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR80	Error log 7: The error code	○	○	-	-	-	R	0
*SR81	Error log 7: The year and the month	○	○	-	-	-	R	0
*SR82	Error log 7: The day and the hour	○	○	-	-	-	R	0
*SR83	Error log 7: The minute and the second	○	○	-	-	-	R	0
*SR84	Error log 8: The rack number and the slot number	○	○	-	-	-	R	0
*SR85	Error log 8: The module ID	○	○	-	-	-	R	0
*SR86	Error log 8: The error code	○	○	-	-	-	R	0
*SR87	Error log 8: The year and the month	○	○	-	-	-	R	0
*SR88	Error log 8: The day and the hour	○	○	-	-	-	R	0
*SR89	Error log 8: The minute and the second	○	○	-	-	-	R	0
*SR90	Error log 9: The rack number and the slot number	○	○	-	-	-	R	0
*SR91	Error log 9: The module ID	○	○	-	-	-	R	0
*SR92	Error log 9: The error code	○	○	-	-	-	R	0
*SR93	Error log 9: The year and the month	○	○	-	-	-	R	0
*SR94	Error log 9: The day and the hour	○	○	-	-	-	R	0
*SR95	Error log 9: The minute and the second	○	○	-	-	-	R	0
*SR96	Error log 10: The rack number and the slot number	○	○	-	-	-	R	0
*SR97	Error log 10: The module ID	○	○	-	-	-	R	0
*SR98	Error log 10: The error code	○	○	-	-	-	R	0
*SR99	Error log 10: The year and the month	○	○	-	-	-	R	0
*SR100	Error log 10: The day and the hour	○	○	-	-	-	R	0
*SR101	Error log 10: The minute and the second	○	○	-	-	-	R	0
*SR102	Error log 11: The rack number and the slot number	○	○	-	-	-	R	0
*SR103	Error log 11: The module ID	○	○	-	-	-	R	0
*SR104	Error log 11: The error code	○	○	-	-	-	R	0
*SR105	Error log 11: The year and the month	○	○	-	-	-	R	0
*SR106	Error log 11: The day and the hour	○	○	-	-	-	R	0
*SR107	Error log 11: The minute and the second	○	○	-	-	-	R	0
*SR108	Error log 12: The rack number and the slot	○	○	-	-	-	R	0

A2

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	number							
*SR109	Error log 12: The module ID	○	○	–	–	–	R	0
*SR110	Error log 12: The error code	○	○	–	–	–	R	0
*SR111	Error log 12: The year and the month	○	○	–	–	–	R	0
*SR112	Error log 12: The day and the hour	○	○	–	–	–	R	0
*SR113	Error log 12: The minute and the second	○	○	–	–	–	R	0
*SR114	Error log 13: The rack number and the slot number	○	○	–	–	–	R	0
*SR115	Error log 13: The module ID	○	○	–	–	–	R	0
*SR116	Error log 13: The error code	○	○	–	–	–	R	0
*SR117	Error log 13: The year and the month	○	○	–	–	–	R	0
*SR118	Error log 13: The day and the hour	○	○	–	–	–	R	0
*SR119	Error log 13: The minute and the second	○	○	–	–	–	R	0
*SR120	Error log 13: The rack number and the slot number	○	○	–	–	–	R	0
*SR121	Error log 14: The rack number and the slot number	○	○	–	–	–	R	0
*SR122	Error log 14: The module ID	○	○	–	–	–	R	0
*SR123	Error log 14: The error code	○	○	–	–	–	R	0
*SR124	Error log 14: The year and the month	○	○	–	–	–	R	0
*SR125	Error log 14: The day and the hour	○	○	–	–	–	R	0
*SR126	Error log 15: The rack number and the slot number	○	○	–	–	–	R	0
*SR127	Error log 15: The module ID	○	○	–	–	–	R	0
*SR128	Error log 15: The error code	○	○	–	–	–	R	0
*SR129	Error log 15: The year and the month	○	○	–	–	–	R	0
*SR130	Error log 15: The day and the hour	○	○	–	–	–	R	0
*SR131	Error log 15: The minute and the second	○	○	–	–	–	R	0
*SR132	Error log 16: The rack number and the slot number	○	○	–	–	–	R	0
*SR133	Error log 16: The module ID	○	○	–	–	–	R	0
*SR134	Error log 16: The error code	○	○	–	–	–	R	0
*SR135	Error log 16: The year and the month	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR136	Error log 16: The day and the hour	○	○	-	-	-	R	0
*SR137	Error log 16: The minute and the second	○	○	-	-	-	R	0
*SR138	Error log 17: The rack number and the slot number	○	○	-	-	-	R	0
*SR139	Error log 17: The module ID	○	○	-	-	-	R	0
*SR140	Error log 17: The error code	○	○	-	-	-	R	0
*SR141	Error log 17: The year and the month	○	○	-	-	-	R	0
SR142	Error log 17: The day and the hour	○	○	-	-	-	R	0
*SR143	Error log 17: The minute and the second	○	○	-	-	-	R	0
*SR144	Error log 18: The rack number and the slot number	○	○	-	-	-	R	0
*SR145	Error log 18: The module ID	○	○	-	-	-	R	0
*SR146	Error log 18: The error code	○	○	-	-	-	R	0
*SR147	Error log 18: The year and the month	○	○	-	-	-	R	0
*SR148	Error log 18: The day and the hour	○	○	-	-	-	R	0
*SR149	Error log 18: The minute and the second	○	○	-	-	-	R	0
*SR150	Error log 19: The rack number and the slot number	○	○	-	-	-	R	0
*SR151	Error log 19: The module ID	○	○	-	-	-	R	0
*SR152	Error log 19: The error code	○	○	-	-	-	R	0
*SR153	Error log 19: The year and the month	○	○	-	-	-	R	0
*SR154	Error log 19: The day and the hour	○	○	-	-	-	R	0
*SR155	Error log 19: The minute and the second	○	○	-	-	-	R	0
*SR156	Error log 20: The rack number and the slot number	○	○	-	-	-	R	0
*SR157	Error log 20: The module ID	○	○	-	-	-	R	0
*SR158	Error log 20: The error code	○	○	-	-	-	R	0
*SR159	Error log 20: The year and the month	○	○	-	-	-	R	0
*SR160	Error log 20: The day and the hour	○	○	-	-	-	R	0
*SR161	Error log 20: The minute and the second	○	○	-	-	-	R	0
*SR201	Communication address of COM1	○	○	-	-	-	R/W	1
*SR202	Communication address of COM2	○	×	-	-	-	R/W	3
*SR209	Communication protocol of COM1	○	○	-	-	-	R/W	16#0024

A2

SR	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR210	COM1 communication timeout	○	○	3000 ms	–	–	R/W	3000 ms
*SR211	Number of times the command is resent through COM1	○	○	–	–	–	R/W	3
*SR212	Communication protocol of COM2	○	×	–	–	–	R/W	16#0024
*SR213	COM2 communication timeout	○	×	3000 ms	–	–	R/W	3000 ms
*SR214	Number of times the command is resent through COM2	○	○	–	–	–	R/W	3
*SR215	Interface code of COM1	○	○	–	–	–	R/W	0
*SR216	Interface code of COM2	○	×	–	–	–	R/W	0
*SR220	Value of the year in the real-time clock (RTC): 00~99 (A.D.)	○	○	–	–	–	R	0
*SR221	Value of the month in the real-time clock (RTC): 01~12	○	○	–	–	–	R	1
*SR222	Value of the day in the real-time clock (RTC): 1~31	○	○	–	–	–	R	1
*SR223	Value of the hour in the real-time clock (RTC): 00~23	○	○	–	–	–	R	0
*SR224	Value of the minute in the real-time clock (RTC): 00~59	○	○	–	–	–	R	0
*SR225	Value of the second in the real-time clock (RTC): 00~59	○	○	–	–	–	R	0
*SR226	Value of the week in the real-time clock (RTC): 1~7	○	○	–	–	–	R	1
*SR227	Number of download logs (The maximum number is 20.)	○	○	–	–	–	R	0
*SR228	Download log pointer	○	○	–	–	–	R	0
*SR229	Download log 1: The action number	○	○	–	–	–	R	0
*SR230	Download log 1: The year and the month	○	○	–	–	–	R	0
*SR231	Download log 1: The day and the hour	○	○	–	–	–	R	0
*SR232	Download log 1: The minute and the second	○	○	–	–	–	R	0
*SR233	Download log 2: The action number	○	○	–	–	–	R	0
*SR234	Download log 2: The year and the month	○	○	–	–	–	R	0
*SR235	Download log 2: The day and the hour	○	○	–	–	–	R	0
*SR236	Download log 2: The minute and the second	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR237	Download log 3: The action number	○	○	-	-	-	R	0
*SR238	Download log 3: The year and the month	○	○	-	-	-	R	0
*SR239	Download log 3: The day and the hour	○	○	-	-	-	R	0
*SR240	Download log 3: The minute and the second	○	○	-	-	-	R	0
*SR241	Download log 4: The action number	○	○	-	-	-	R	0
*SR242	Download log 4: The year and the month	○	○	-	-	-	R	0
*SR243	Download log 4: The day and the hour	○	○	-	-	-	R	0
*SR244	Download log 4: The minute and the second	○	○	-	-	-	R	0
*SR245	Download log 5: The action number	○	○	-	-	-	R	0
*SR246	Download log 5: The year and the month	○	○	-	-	-	R	0
*SR247	Download log 5: The day and the hour	○	○	-	-	-	R	0
*SR248	Download log 5: The minute and the second	○	○	-	-	-	R	0
*SR249	Download log 6: The action number	○	○	-	-	-	R	0
*SR250	Download log 6: The year and the month	○	○	-	-	-	R	0
*SR251	Download log 6: The day and the hour	○	○	-	-	-	R	0
*SR252	Download log 6: The minute and the second	○	○	-	-	-	R	0
*SR253	Download log 7: The action number	○	○	-	-	-	R	0
*SR254	Download log 7: The year and the month	○	○	-	-	-	R	0
*SR255	Download log 7: The day and the hour	○	○	-	-	-	R	0
*SR256	Download log 7: The minute and the second	○	○	-	-	-	R	0
*SR257	Download log 8: The action number	○	○	-	-	-	R	0
*SR258	Download log 8: The year and the month	○	○	-	-	-	R	0
*SR259	Download log 8: The day and the hour	○	○	-	-	-	R	0
*SR260	Download log 8: The minute and the second	○	○	-	-	-	R	0
*SR261	Download log 9: The action number	○	○	-	-	-	R	0
*SR262	Download log 9: The year and the month	○	○	-	-	-	R	0
*SR263	Download log 9: The day and the hour	○	○	-	-	-	R	0
*SR264	Download log 9: The minute and the second	○	○	-	-	-	R	0
*SR265	Download log 10: The action number	○	○	-	-	-	R	0
*SR266	Download log 10: The year and the month	○	○	-	-	-	R	0

A2

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR267	Download log 10: The day and the hour	○	○	–	–	–	R	0
*SR268	Download log 10: The minute and the second	○	○	–	–	–	R	0
*SR269	Download log 11: The action number	○	○	–	–	–	R	0
*SR270	Download log 11: The year and the month	○	○	–	–	–	R	0
*SR271	Download log 11: The day and the hour	○	○	–	–	–	R	0
*SR272	Download log 11: The minute and the second	○	○	–	–	–	R	0
*SR273	Download log 12: The action number	○	○	–	–	–	R	0
*SR274	Download log 12: The year and the month	○	○	–	–	–	R	0
*SR275	Download log 12: The day and the hour	○	○	–	–	–	R	0
*SR276	Download log 12: The minute and the second	○	○	–	–	–	R	0
*SR277	Download log 13: The action number	○	○	–	–	–	R	0
*SR278	Download log 13: The year and the month	○	○	–	–	–	R	0
*SR279	Download log 13: The day and the hour	○	○	–	–	–	R	0
*SR280	Download log 13: The minute and the second	○	○	–	–	–	R	0
*SR281	Download log 14: The action number	○	○	–	–	–	R	0
*SR282	Download log 14: The year and the month	○	○	–	–	–	R	0
*SR283	Download log 14: The day and the hour	○	○	–	–	–	R	0
*SR284	Download log 14: The minute and the second	○	○	–	–	–	R	0
*SR285	Download log 15: The action number	○	○	–	–	–	R	0
*SR286	Download log 15: The year and the month	○	○	–	–	–	R	0
*SR287	Download log 15: The day and the hour	○	○	–	–	–	R	0
*SR288	Download log 15: The minute and the second	○	○	–	–	–	R	0
*SR289	Download log 16: The action number	○	○	–	–	–	R	0
*SR290	Download log 16: The year and the month	○	○	–	–	–	R	0
*SR291	Download log 16: The day and the hour	○	○	–	–	–	R	0
*SR292	Download log 16: The minute and the second	○	○	–	–	–	R	0
*SR293	Download log 17: The action number	○	○	–	–	–	R	0
*SR294	Download log 17: The year and the month	○	○	–	–	–	R	0
*SR295	Download log 17: The day and the hour	○	○	–	–	–	R	0
*SR296	Download log 17: The minute and the second	○	○	–	–	–	R	0
*SR297	Download log 18: The action number	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(Module) AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR298	Download log 18: The year and the month	○	○	—	—	—	R	0
*SR299	Download log 18: The day and the hour	○	○	—	—	—	R	0
*SR300	Download log 18: The minute and the second	○	○	—	—	—	R	0
*SR301	Download log 19: The action number	○	○	—	—	—	R	0
*SR302	Download log 19: The year and the month	○	○	—	—	—	R	0
*SR303	Download log 19: The day and the hour	○	○	—	—	—	R	0
*SR304	Download log 19: The minute and the second	○	○	—	—	—	R	0
*SR305	Download log 20: The action number	○	○	—	—	—	R	0
*SR306	Download log 20: The year and the month	○	○	—	—	—	R	0
*SR307	Download log 20: The day and the hour	○	○	—	—	—	R	0
*SR308	Download log 20: The minute and the second	○	○	—	—	—	R	0
*SR309	Number of PLC status change logs (The maximum number is 20.)	○	○	—	—	—	R	0
*SR310	PLC status change log pointer	○	○	—	—	—	R	0
*SR311	PLC status change log 1: The action number	○	○	—	—	—	R	0
*SR312	PLC status change log 1: The year and the month	○	○	—	—	—	R	0
*SR313	PLC status change log 1: The day and the hour	○	○	—	—	—	R	0
*SR314	PLC status change log 1: The minute and the second	○	○	—	—	—	R	0
*SR315	PLC status change log 2: The action number	○	○	—	—	—	R	0
*SR316	PLC status change log 2: The year and the month	○	○	—	—	—	R	0
*SR317	PLC status change log 2: The day and the hour	○	○	—	—	—	R	0
*SR318	PLC status change log 2: The minute and the second	○	○	—	—	—	R	0
*SR319	PLC status change log 3: The action number	○	○	—	—	—	R	0
*SR320	PLC status change log 3: The year and the month	○	○	—	—	—	R	0
*SR321	PLC status change log 3: The day and the hour	○	○	—	—	—	R	0
*SR322	PLC status change log 3: The minute and the second	○	○	—	—	—	R	0
*SR323	PLC status change log 4: The action number	○	○	—	—	—	R	0
*SR324	PLC status change log 4: The year and the	○	○	—	—	—	R	0

A2

SR	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	month							
*SR325	PLC status change log 4: The day and the hour	○	○	–	–	–	R	0
*SR326	PLC status change log 4: The minute and the second	○	○	–	–	–	R	0
*SR327	PLC status change log 5: The action number	○	○	–	–	–	R	0
*SR328	PLC status change log 5: The year and the month	○	○	–	–	–	R	0
*SR329	PLC status change log 5: The day and the hour	○	○	–	–	–	R	0
*SR330	PLC status change log 5: The minute and the second	○	○	–	–	–	R	0
*SR331	PLC status change log 6: The action number	○	○	–	–	–	R	0
*SR332	PLC status change log 6: The year and the month	○	○	–	–	–	R	0
*SR333	PLC status change log 6: The day and the hour	○	○	–	–	–	R	0
*SR334	PLC status change log 6: The minute and the second	○	○	–	–	–	R	0
*SR335	PLC status change log 7: The action number	○	○	–	–	–	R	0
*SR336	PLC status change log 7: The year and the month	○	○	–	–	–	R	0
*SR337	PLC status change log 7: The day and the hour	○	○	–	–	–	R	0
*SR338	PLC status change log 7: The minute and the second	○	○	–	–	–	R	0
*SR339	PLC status change log 8: The action number	○	○	–	–	–	R	0
*SR340	PLC status change log 8: The year and the month	○	○	–	–	–	R	0
*SR341	PLC status change log 8: The day and the hour	○	○	–	–	–	R	0
*SR342	PLC status change log 8: The minute and the second	○	○	–	–	–	R	0
*SR343	PLC status change log 9: The action number	○	○	–	–	–	R	0
*SR344	PLC status change log 9: The year and the month	○	○	–	–	–	R	0
*SR345	PLC status change log 9: The day and the hour	○	○	–	–	–	R	0
*SR346	PLC status change log 9: The minute and the second	○	○	–	–	–	R	0
*SR347	PLC status change log 10: The action number	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(Module) AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR348	PLC status change log 10: The year and the month	○	○	-	-	-	R	0
*SR349	PLC status change log 10: The day and the hour	○	○	-	-	-	R	0
*SR350	PLC status change log 10: The minute and the second	○	○	-	-	-	R	0
*SR351	PLC status change log 11: The action number	○	○	-	-	-	R	0
*SR352	PLC status change log 11: The year and the month	○	○	-	-	-	R	0
*SR353	PLC status change log 11: The day and the hour	○	○	-	-	-	R	0
*SR354	PLC status change log 11: The minute and the second	○	○	-	-	-	R	0
*SR355	PLC status change log 12: The action number	○	○	-	-	-	R	0
*SR356	PLC status change log 12: The year and the month	○	○	-	-	-	R	0
*SR357	PLC status change log 12: The day and the hour	○	○	-	-	-	R	0
*SR358	PLC status change log 12: The minute and the second	○	○	-	-	-	R	0
*SR359	PLC status change log 13: The action number	○	○	-	-	-	R	0
*SR360	PLC status change log 13: The year and the month	○	○	-	-	-	R	0
*SR361	PLC status change log 13: The day and the hour	○	○	-	-	-	R	0
*SR362	PLC status change log 13: The minute and the second	○	○	-	-	-	R	0
*SR363	PLC status change log 14: The action number	○	○	-	-	-	R	0
*SR364	PLC status change log 14: The year and the month	○	○	-	-	-	R	0
*SR365	PLC status change log 14: The day and the hour	○	○	-	-	-	R	0
*SR366	PLC status change log 14: The minute and the second	○	○	-	-	-	R	0
*SR367	PLC status change log 15: The action number	○	○	-	-	-	R	0
*SR368	PLC status change log 15: The year and the month	○	○	-	-	-	R	0

A2

SR	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR369	PLC status change log 15: The day and the hour	○	○	–	–	–	R	0
*SR370	PLC status change log 15: The minute and the second	○	○	–	–	–	R	0
*SR371	PLC status change log 16: The action number	○	○	–	–	–	R	0
*SR372	PLC status change log 16: The year and the month	○	○	–	–	–	R	0
*SR373	PLC status change log 16: The day and the hour	○	○	–	–	–	R	0
*SR374	PLC status change log 16: The minute and the second	○	○	–	–	–	R	0
*SR375	PLC status change log 17: The action number	○	○	–	–	–	R	0
*SR376	PLC status change log 17: The year and the month	○	○	–	–	–	R	0
*SR377	PLC status change log 17: The day and the hour	○	○	–	–	–	R	0
*SR378	PLC status change log 17: The minute and the second	○	○	–	–	–	R	0
*SR379	PLC status change log 18: The action number	○	○	–	–	–	R	0
*SR380	PLC status change log 18: The year and the month	○	○	–	–	–	R	0
*SR381	PLC status change log 18: The day and the hour	○	○	–	–	–	R	0
*SR382	PLC status change log 18: The minute and the second	○	○	–	–	–	R	0
*SR383	PLC status change log 19: The action number	○	○	–	–	–	R	0
*SR384	PLC status change log 19: The year and the month	○	○	–	–	–	R	0
*SR385	PLC status change log 19: The day and the hour	○	○	–	–	–	R	0
*SR386	PLC status change log 19: The minute and the second	○	○	–	–	–	R	0
*SR387	PLC status change log 20: The action number	○	○	–	–	–	R	0
*SR388	PLC status change log 20: The year and the month	○	○	–	–	–	R	0
*SR389	PLC status change log 20: The day and the hour	○	○	–	–	–	R	0
*SR390	PLC status change log 20: The minute and the	○	○	–	–	–	R	0

SR	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
		○	○					
	second							
SR391	Value of the year in the real-time clock (RTC): 00~99 (A.D.)	○	○	-	-	-	R	0
SR392	Value of the month in the real-time clock (RTC): 01~12	○	○	-	-	-	R	1
SR393	Value of the day in the real-time clock (RTC): 1~31	○	○	-	-	-	R	1
SR394	Value of the hour in the real-time clock (RTC): 00~23	○	○	-	-	-	R	0
SR395	Value of the minute in the real-time clock (RTC): 00~59	○	○	-	-	-	R	0
SR396	Value of the second in the real-time clock (RTC): 00~59	○	○	-	-	-	R	0
SR397	Value of the week in the real-time clock (RTC): 1~7	○	○	-	-	-	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	-	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	-	R/W	0
*SR409	The pulse is ON for n seconds and is OFF for n seconds during the 2n second clock pulse. The interval n is stored in SR409, and the setting range is 1~32767.	○	○	-	-	-	R/W	30
*SR410	The pulse is ON for n milliseconds and is OFF for n milliseconds during the 2n millisecond clock pulse. The interval n is stored in SR410.	○	○	-	-	-	R/W	30
SR411	The current scan time is stored in SR411 and SR412, and the unit of measurement is 100 microseconds. The value of the millisecond is stored in SR411. (The range is 0~65535.) The value of the microsecond is stored in SR421.	○	○	0	-	-	R	0
SR412	(The range is 0~900.). For example, 12 is stored in SR411 and 300 is stored in SR412 when the current scan time is 12.3 milliseconds.	○	○	0	-	-	R	0
SR413	The maximum scan time is stored in SR413 and SR414, and the unit of measurement is	○	○	0	-	-	R	0

A2

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
SR414	100 microseconds. The value of the millisecond is stored in SR413.		○	0	–	–	R	0
SR415	The maximum scan time is stored in SR415 and SR416, and the unit of measurement is 100 microseconds. The value of the millisecond is stored in SR415.		○	0	–	–	R	0
SR416			○	0	–	–	R	0
*SR453	If an error occurs during the operation of the memory card, the error code will be recorded.	○	○	–	–	–	R	0
SR621	Interrupt character used in the instruction RS (COM1)	○	○	–	–	–	R/W	0
SR622	Interrupt character used in the instruction RS (COM2)	x	x	–	–	–	R/W	0
SR623	Bit 0–bit 15: The conditions of the interrupt programs I0–I15 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR624	Bit 0–bit 15: The conditions of the interrupt programs I16–I31 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR625	Bit 0–bit 15: The conditions of the interrupt programs I32–I47 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR626	Bit 0–bit 15: The conditions of the interrupt programs I48–I63 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR627	Bit 0–bit 15: The conditions of the interrupt programs I64–I79 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR628	Bit 0–bit 15: The conditions of the interrupt programs I80–I95 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR629	Bit 0–bit 15: The conditions of the interrupt programs I96–I111 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR630	Bit 0–bit 15: The conditions of the interrupt programs I112–I127 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR631	Bit 0–bit 15: The conditions of the interrupt programs I128–I143 are set by the instruction IMASK.	○	○	FFFF	–	–	R	FFFF
SR632	Bit 0–bit 15: The conditions of the interrupt programs I144–I159 are set by the instruction	○	○	FFFF	–	–	R	FFFF

SR	Function	AH-xxEMC(CPU)		OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
	IMASK.							
SR633	Bit 0~bit 15: The conditions of the interrupt programs I160~I175 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
SR634	Bit 0~bit 15: The conditions of the interrupt programs I176~I191 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
SR635	Bit 0~bit 15: The conditions of the interrupt programs I192~I207 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
SR636	Bit 0~bit 15: The conditions of the interrupt programs I208~I213 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
SR637	Bit 0~bit 15: The conditions of the interrupt programs I214~I229 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
SR638	Bit 0~bit 15: The conditions of the interrupt programs I230~I255 are set by the instruction IMASK.	○	○	FFFF	—	—	R	FFFF
*SR655	Recording the I/O table mapping error or I/O module error occurring on rack 1.	○	○	0	—	—	R	0
*SR663 ↓ SR674	Recording the error code of the mapping error occurring on slot 0 of rack 1. ↓ Recording the error code of the mapping error occurring on slot 11 of rack 1.	○	x	0	—	—	R	0
SR731	If the external 24 V voltage is abnormal, the value of the corresponding bit is 1.	○	○	0	—	—	R	0
*SR1000	High word in the Ethernet IP address	○	○	—	—	—	R/W	C0A8
*SR1001	Low word in the Ethernet IP address	○	○	—	—	—	R/W	0101
*SR1002	High word in the Ethernet netmask address	○	○	—	—	—	R/W	FFFF
*SR1003	Low word in the Ethernet netmask address	○	○	—	—	—	R/W	FF00
*SR1004	High word in the Ethernet gateway address	○	○	—	—	—	R/W	C0A8
*SR1005	Low word in the Ethernet gateway address	○	○	—	—	—	R/W	0101
*SR1006	Time for which the TCP connection has been persistent	○	○	—	—	—	R/W	0060
SR1007	Ethernet transmission speed	○	○	0	—	—	R	0
SR1008	Ethernet transmission mode	○	○	0	—	—	R	0

A2

SR	Function	AH-xxEMC(CPU)	AH-xxEMC(Module)	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Attribute	Default
*SR1329	Main backplane ID	○	×	–	–	–	R/W	0
*SR1330	Main slot number	○	×	–	–	–	R/W	0

Note: As to the SR numbers marked “*”, refer to **A.3 Additional Remarks on SM and SR**.

A.2.1. Refresh Time of Special Data Registers

Special data register	Refresh time
SR0~SR2	The register is refreshed when the program is executed in error.
SR4	SR4 is refreshed when there is a grammar check error
SR5~SR6	The register is refreshed when the program is downloaded to the PLC, or when the PLC is supplied with power and starts to run for the first time.
SR8	SR8 is refreshed when there is a watchdog timer error.
SR40~SR161	The register is refreshed when an error occurs.
SR201~SR216	Users set the value and clear it.
SR220~SR226	The register is refreshed every scan cycle.
SR227~SR308	The register is refreshed when the program is downloaded to the PLC.
SR309~SR390	The register is refreshed when the status of the PLC changes.
SR391~SR397	The register is refreshed every scan cycle.
SR407	SR407 is refreshed every second.
SR408	SR408 is refreshed whenever the instruction END is executed.
SR409~SR410	Users set the value and clear it.
SR411~SR416	The register is refreshed whenever the instruction END is executed.
SR453	SR453 is refreshed when an error occurs.
SR621~SR622	Users set the value and clear it.
SR623~SR638	The register is refreshed when the instruction IMASK is executed.
SR655~SR730	The register is refreshed when an error occurs in the I/O module.
SR1000~SR1006	Users set the value and clear it.
SR1007	Ethernet transmission speed
SR1008	Ethernet transmission mode
SR1100~SR1117	The register is refreshed every scan cycle.
SR1118~SR1128	The register is refreshed when the parameter is downloaded to the PLC.
SR1129~SR1130	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.

Special data register	Refresh time
SR1131~SR1141	The register is refreshed when the parameter is downloaded to the PLC.
SR1142~SR1143	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1144~SR1154	The register is refreshed when the parameter is downloaded to the PLC.
SR1155~SR1156	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1157~SR1167	The register is refreshed when the parameter is downloaded to the PLC.
SR1168~SR1169	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1170~SR1180	The register is refreshed when the parameter is downloaded to the PLC.
SR1181~SR1182	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1183~SR1193	The register is refreshed when the parameter is downloaded to the PLC.
SR1194~SR1195	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1196~SR1206	The register is refreshed when the parameter is downloaded to the PLC.
SR1207~SR1208	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1209~SR1219	The register is refreshed when the parameter is downloaded to the PLC.
SR1220~SR1221	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1222~SR1231	The register is refreshed when the parameter is downloaded to the PLC.
SR1232~SR1233	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1234~SR1243	The register is refreshed when the parameter is downloaded to the PLC.
SR1244~SR1245	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1246~SR1255	The register is refreshed when the parameter is downloaded to the PLC.
SR1256~SR1257	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1258~SR1267	The register is refreshed when the parameter is downloaded to the PLC.
SR1268~SR1269	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1270~SR1279	The register is refreshed when the parameter is downloaded to the PLC.
SR1280~SR1281	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1282~SR1291	The register is refreshed when the parameter is downloaded to the PLC.
SR1292~SR1293	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1294~SR1303	The register is refreshed when the parameter is downloaded to the PLC.
SR1304~SR1305	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.

A2

Appendix 2. Special Data Registers Table

Special data register	Refresh time
SR1306~SR1315	The register is refreshed when the parameter is downloaded to the PLC.
SR1316~SR1320	The register is refreshed when the parameter is downloaded to the PLC, or when the PLC is supplied with power.
SR1329~SR1334	Users set the value and clear it.
SR1335~SR1336	The register is refreshed every scan cycle when the PLC Link is enabled.
SR1337~SR1787	Users set the value and clear it.
SR1792~SR2047	The register is refreshed every scan cycle.

A.3. Additional Remarks on SM and SR

1. The scan timeout timer

- SM8/SR8

When a scan timeout occurs during the execution of the program, the error LED indicator on the PLC is ON all the time, and SM8 is ON.

The content of SR8 is the step address at which the watchdog timer is ON.

2. Clearing the warning light

- SM22

If SM22 is ON, the error log and the warning light will be cleared.

3. The real-time clock

- SM220, SR220~SR226, and SR391~SR397

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

When SM220 is switched from OFF to ON, the real-time clock is calibrated.

If the value of the second in the real-time clock is within the range between 0 and 29, the value of the minute is fixed, and the value of the second is cleared to zero.

If the value of the second in the real-time clock is within the range between 30 and 59, the value of the minute increases by one, and the value of the second is cleared to zero.

The corresponding functions and values of SR220~SR226 and SR391~SR397 are as follows.

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 lies in the fact that the former adopts the binary-coded decimal while the latter adopts the decimal system. For example, December is represented as 12 in SR392 while it is represented as 12 in the binary-coded decimal.

Please refer to section 3.19 for more information related to the real-time clock.

4. The functions related to communication

- SM96~SM109, SM209~SM212, SR201~SR202, and SR209~SR216

SR215 and SR216 are used to record the interface code of the communication port on the PLC. The functions represented by the interface codes are as follows.

Code	0	1	2
Function	RS-232	RS-485	RS-422

When the interface of the communication port on the PLC is RS-485, RS-232, or RS-422, SR209 records the communication format of COM1 on the PLC, and SR212 records the communication format of COM2 on the PLC. The setting values of the communication protocols are shown in the following table.

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 bit		1 bit (value=0)	2 bits (value=1)	
b4 b5 b6 b7	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	RS-232 does not support the baud rate.
	1000	(16#8)	:	460800	RS-232 does not support the baud rate.
	1001	(16#9)	:	921600	RS-232 does not support the baud rate.
b8~b15	Undefined (reserved)				

A3

5. Clearing the contents of the device

- SM204/SM205

Device number	Device which is cleared
SM204 All non-latched areas are cleared.	<p>The non-latched areas in the input relays, the output relays, the stepping relays, the auxiliary relays, and the link registers are cleared.</p> <p>The non-latched areas in the timers, the counters, and the 32-bit counters are cleared.</p> <p>The non-latched areas in the data registers and the index registers are cleared.</p> <p>It takes 530 milliseconds to clear the device. The watchdog timer does not act during this period of time.</p>

Device number	Device which is cleared
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. It takes 30 milliseconds to clear the device. The watchdog timer does not act during this period of time.

For more information of latched areas, refer to *AH Motion – Operation Manual*, Chapter 4 Operating ISPSOft.

6. The error log in the PLC

- SR40~SR161

SR40: The maximum number of error logs which are stored in SR40 is 20. Every 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 when the errors occur and the error positions are recorded in SR42 - SR161. The corresponding functions of these data registers are as follows.

No.	Rack	Slot	Module ID	Error code	Time when the error occurs					
					Year	Month	Day	Hour	Minute	Second
1	SR42 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	SR108 Low byte	SR109	SR110	SR111 High byte	SR111 Low byte	SR112 High byte	SR112 Low byte	SR113 High byte	SR113 Low byte

No.	Rack	Slot	Module ID	Error code	Time when the error occurs					
					Year	Month	Day	Hour	Minute	Second
13	SR114 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	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 High byte	SR156 Low byte	SR157	SR158	SR159 High byte	SR159 Low byte	SR160 High byte	SR160 Low byte	SR161 High byte	SR161 Low byte

7. The download log in the PLC

- 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
Downloading the setting of the PLC	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 when the downloading actions occur and the action numbers are recorded in SR229-SR305. The corresponding functions of these data registers are as follows.

No.	Action number	*Time when the download action 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	SR242	SR243	SR243	SR244	SR244

No.	Action number	*Time when the download action occurs					
		Year	Month	Day	Hour	Minute	Second
		High byte	Low byte	High byte	Low byte	High byte	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 High byte	SR298 Low byte	SR299 High byte	SR299 Low byte	SR300 High byte	SR300 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

*Time when the download action occurs: The data is stored as the values in the binary-coded decimal. The range of values is as follows.

Function	Value
Year	00~99 (A.D.)
Month	01~12
Day	01~31

Function	Value
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 which are stored in SR309 is 20. Every PLC status change log occupies 4 registers. The PLC status change actions which are recorded are numbered, as shown in the following table.

PLC status change	Number
The PLC is supplied with power.	1
The PLC is disconnected.	2
The PLC starts to run.	3
The PLC stops running.	4
Default setting of the PLC (1. RST button; 2. Communication command)	5
Pressing the CLR button on the PLC (Clearing the data in the latched device)	6

SR310: The PLC status change log pointer points to the latest PLC status change log. When the PLC status is changed once, the value of the PLC status change log pointer increases by one. The range of pointer values is 0~19. For example, the 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 corresponding functions of these data registers are as follows.

No.	Action number	*Time when the PLC status change action 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 High byte	SR320 Low byte	SR321 High byte	SR321 Low byte	SR322 High byte	SR322 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	SR332	SR333	SR333	SR334	SR334

No.	Action number	*Time when the PLC status change action occurs					
		Year	Month	Day	Hour	Minute	Second
		High byte	Low byte	High byte	Low byte	High byte	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

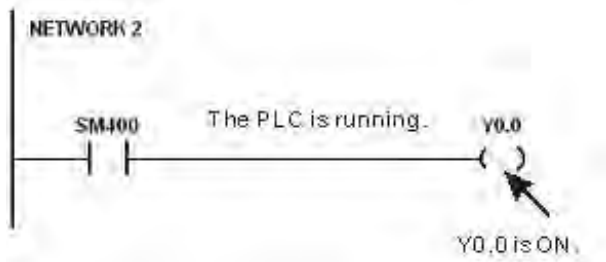
*Time when the PLC status change action occurs: The data is stored as the values in the binary-coded decimal. The range of values is as follows.

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

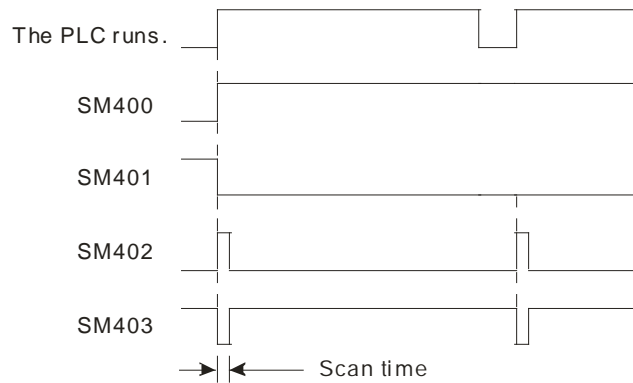
SM400: The normally-open contact



SM401: The normally-closed contact

SM402: SM402 is ON during the first scan time, and then is switched OFF. The pulse width equals one scan time. Users can use this contact to do the initial setting.

SM403: SM403 is OFF during the first scan time, and then is switched ON. That is, the negative pulse is generated the moment the PLC runs.



A3

10. The initial clock pulse

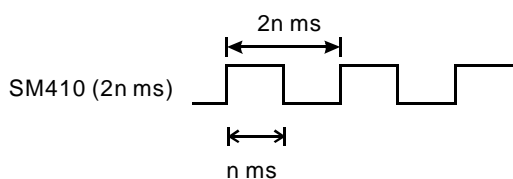
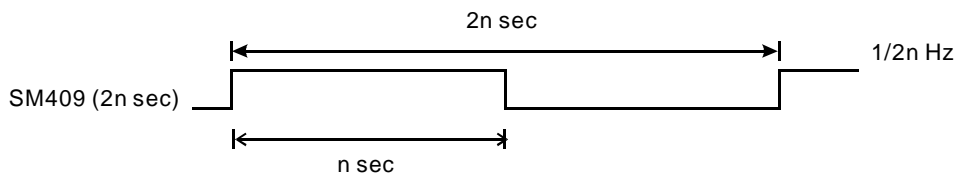
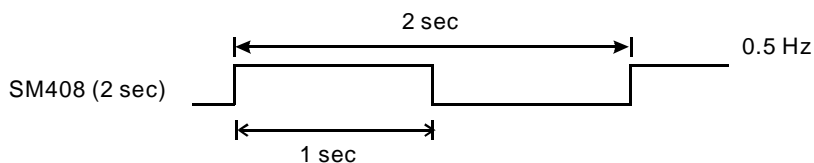
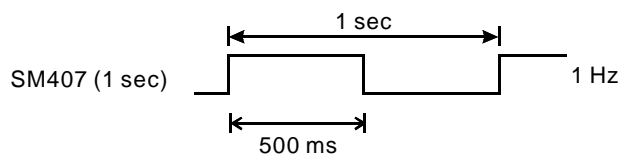
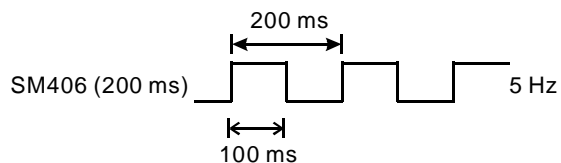
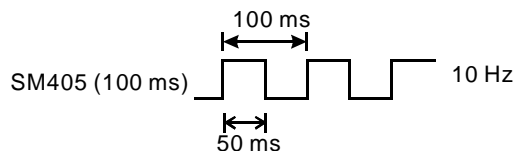
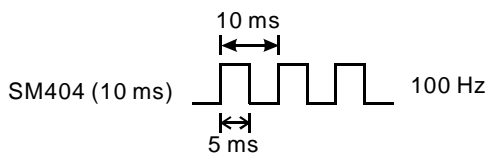
- SM404~SM410, and SR409~SR410

The PLC provides seven types of clock pulses. When the PLC is supplied with power, the seven types of clock pulses act automatically. Users can set the interval of the clock pulse in SM409 and SM410.

Device	Function
SM404	10 millisecond clock pulse during which the pulse is ON for 5 milliseconds and is OFF for 5 milliseconds
SM405	100 millisecond clock pulse during which the pulse is ON for 50 milliseconds and is OFF for 50 milliseconds
SM406	200 millisecond clock pulse during which the pulse is ON for 100 milliseconds and is OFF for 100 milliseconds
SM407	One second clock pulse during which the pulse is ON for 500 milliseconds and is OFF for 500 milliseconds
SM408	Two second clock pulse during which the pulse is ON for one second and is OFF for one second

Device	Function
SM409	2n second clock pulse during which the pulse is ON for n seconds and is OFF for n seconds The interval n is specified by SR409.
SM410	2n millisecond clock pulse during which the pulse is ON for n milliseconds and is OFF for n milliseconds The interval n is specified by SR410.

The clock pulses are illustrated as follows.



A3

11. The flags related to the memory card

- SM450~SM453, and SR453

The memory card is used to backup the data in the PLC. The corresponding functions of these special auxiliary relays and the corresponding function of SR453 are as follows.

Device	Function
SM450	Whether the memory card exists ON: The memory card exists. OFF: The memory card does not exist.
SM451	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 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.
SM453	An error occurs during the operation of the memory card. ON: An error occurs.
SR453	If an error occurs during the operation of the memory card, the error code will be recorded.

A3

12. The registers related to the I/O module

- SR655, SR663~SR674 indicate the slots with errors and their error codes.
 - SR655: slots with errors, i.e. mapping error occurring in the module table.

If the mapping error occurs in the module table, the corresponding bit which indicates the slot with error will be ON. For example, the ON state of bit 4 in SR655 indicates an error occurring at slot 4 in the main backplane.

	Main backplane
	Backplane 1
Device	SR655
Slot 0	Bit0
Slot 1	Bit1
Slot 2	Bit2
Slot 3	Bit3
Slot 4	Bit4
Slot 5	Bit5
Slot 6	Bit6
Slot 7	Bit7
Slot 8	Bit8

	Main backplane
	Backplane 1
Slot 9	Bit9
Slot 10	Bit10
Slot 11	Bit11

- SR663~SR674: error codes of the corresponding slots, i.e. the mapping error code in the module table.

If the mapping error occurs in the module table, the special data register will record the error code. You can read the error code in the special data register to get the error information.

	Main backplane
	Backplane 1
Slot 0	SR663
Slot 1	SR664
Slot 2	SR665
Slot 3	SR666
Slot 4	SR667
Slot 5	SR668
Slot 6	SR669
Slot 7	SR670
Slot 8	SR671
Slot 9	SR672
Slot 10	SR673
Slot 11	SR674

A3

13. The flags related to the Ethernet

- SM1090, SM1091, and SM1106~SM1109

SM number	Description	Function
SM1090	The TCP connection is busy.	ON: TCP connection timeout
SM1091	The UDP connection is busy.	ON: UDP connection timeout
SM1106	Ethernet connection error	OFF: The Ethernet auto-negotiation succeeds. ON: The Ethernet auto-negotiation fails.
SM1107	Basic setting error	OFF: The basic setting is correct. ON: The basic setting is incorrect.
SM1108	Filter setting error	OFF: The filter setting is correct. ON: The filter setting is incorrect.
SM1109	Basic management of the TCP/UDP	The flag is ON when the same port is used.

SM number	Description	Function
	socket—The local port is already used.	

Note: You can refer to *AH Motion - Operation Manual* for more information about troubleshooting.

A.4. Standard Instructions (Sort by Alphabet)

- Columns

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
A	FC	AND\$=	—	—	—	Comparing the strings ON: $S_1 = S_2$ OFF $S_1 \neq S_2$
	FC	AND\$>	—	—	—	Comparing the strings ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$

Items provided in the table

1	Category	The initial of the instruction name
2	FB/FC	FB : Function block; FC : Function
3	Instruction	The instruction name (16-bit/32-bit/64-bit) 32-bit : If the 16-bit instruction can be used as a 32-bit instruction, a D is added in front of the 16-bit instruction to form the 32-bit instruction. 64-bit : If the 32-bit floating-point instruction can be used as a 64-bit floating-point number instruction, a D is added in front of the 32-bit floating-point instruction to form a 64-bit floating-point instruction.
4	Pulse instruction	“✓” indicates that the instruction can be used as a pulse instruction, whereas “—” indicates that it can not. If users want to use the pulse instruction, they only need to add a P in back of the instruction.
5	Description	The function description of the instruction

- Instruction List

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
Symbol	FC	+	D+	—	✓	Addition of binary numbers
	FC	-	D-	—	✓	Subtraction of binary numbers
	FC	*	D*	—	✓	Multiplication of binary numbers

Appendix 4. Standard Instructions (Sort by Alphabet)

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
Symbol	FC	/	D/	—	✓	Division of binary numbers
	FC	\$+	—	—	✓	Linking the strings
	FC	\$CLR	—	—	✓	Clearing the string
	FC	\$DEL	—	—	✓	Deleting the characters in the string
	FC	\$FSTR	—	—	✓	Converting the floating-point number into the string
	FC	\$FVAL	—	—	✓	Converting the string into the floating-point number
	FC	\$INS	—	—	✓	Inserting the string
	FC	\$LEFT	—	—	✓	The retrieve of the characters in the string begins from the left.
	FC	\$LEN	—	—	✓	Calculating the length of the string
	FC	\$MIDR	—	—	✓	Retrieving a part of the string
	FC	\$MIDW	—	—	✓	Replacing a part of the string
	FC	\$MOV	—	—	✓	Transferring the string
	FC	\$RIGHT	—	—	✓	The retrieve of the characters in the string begins from the right.
	FC	\$RPLC	—	—	✓	Replacing the characters in the string
	FC	\$SER	—	—	✓	Searching the string
	FC	\$STR	D\$STR	—	✓	Converting the binary number into the string
	FC	\$VAL	D\$VAL	—	✓	Converting the string into the binary number
	FC	ABS	DABS	—	✓	Absolute value
	FC	ABSD	DABSD	—	✓	Absolute drum sequencer
FC	ALT	—	—	—	Alternating between ON and OFF	
FC	AND\$<	—	—	—	Comparing the strings	

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
A						ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	AND\$<=	—	—	—	Comparing the strings ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	AND\$<>	—	—	—	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	AND\$=	—	—	—	Comparing the strings ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	AND\$>	—	—	—	Comparing the strings ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	AND\$>=	—	—	—	Comparing the strings ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	AND&	DAND&	—	—	ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$
	FC	AND^	DAND^	—	—	ON: $S_1 \wedge S_2 \neq 0$ OFF: $S_1 \wedge S_2 = 0$
	FC	AND	DAND	—	—	ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2 = 0$
	FC	AND<	DAND<	—	—	Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	AND<=	DAND<=	—	—	Comparing the values ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	AND<>	DAND<>	—	—	Comparing the values ON: $S_1 \neq S_2$

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
						OFF: $S_1 = S_2$
	FC	AND=	DAND=	—	—	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	AND>	DAND>	—	—	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	AND>=	DAND>=	—	—	Comparing the values ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	ARWS	—	—	—	Arrow key input
B	FC	B+	DB+	—	✓	Addition of binary-coded decimal numbers $S_1 + S_2 = D$
	FC	B-	DB-	—	✓	Subtraction of binary-coded decimal numbers $S_1 - S_2 = D$
	FC	B*	DB*	—	✓	Multiplication of binary-coded decimal numbers $S_1 * S_2 = D$
	FC	B/	DB/	—	✓	Division of binary-coded decimal numbers $S_1 / S_2 = D$
	FC	BACOS	—	—	✓	Arccosine of the binary-coded decimal number
	FC	BAND	DBAND	—	✓	Deadband control
	FC	BASIN	—	—	✓	Arcsine of the binary-coded decimal number
	FC	BATAN	—	—	✓	Arctangent of the binary-coded decimal number

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
B	FC	BCD	DBCD	—	✓	Converting the binary number into the binary-coded decimal number
	FC	BCDDA	DBCDDA	—	✓	Converting the binary-coded decimal number into the ASCII code
	FC	BCOS	—	—	✓	Cosine of the binary-coded decimal number
	FC	BIN	DBIN	—	✓	Converting the binary-coded decimal number into the binary number
	FC	BINDA	DBINDA	—	✓	Converting the signed decimal number into the ASCII code
	FC	BINHA	DBINHA	—	✓	Converting the binary hexadecimal number into the hexadecimal ASCII code
	FC	BMOV	—	—	✓	Transferring all data
	FC	BON	DBON	—	✓	Checking the state of the bit
	FC	BREAK	—	—	✓	Terminating the FOR-NEXT loop
	FC	BRST	—	—	✓	Resetting the bit in the word device
	FC	BSET	—	—	✓	Setting the bit in the word device to ON
	FC	BSFL	—	—	✓	Shifting the states of the n bit devices by one bit to the left
	FC	BSFR	—	—	✓	Shifting the states of the n bit devices by one bit to the right
	FC	BSIN	—	—	✓	Sine of the binary-coded decimal number
	FC	BSQR	DBSQR	—	✓	Square root of the binary-coded decimal number
FC	BTAN	—	—	✓	Tangent of the binary-coded decimal number	

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	BXCH	—	—	✓	Exchanging all data
C	FC	CCD	—	—	✓	Sum check
	FC	CHKADR	—	—	—	Checking the address of the contact type of pointer register
	FC	CJ	—	—	✓	Conditional jump
	FC	COMRS	—	—	✓	Sending and receiving communication data
	FC	CML	DCML	—	✓	Inverting the data
	FC	CMP	DCMP	—	✓	Comparing the values
	FC	CMPT<	—	—	✓	Comparing the tables ON: <
	FC	CMPT<=	—	—	✓	Comparing the tables ON: ≤
	FC	CMPT<>	—	—	✓	Comparing the tables ON: ≠
	FC	CMPT=	—	—	✓	Comparing the tables ON: =
	FC	CMPT>	—	—	✓	Comparing the tables ON: >
	FC	CMPT>=	—	—	✓	Comparing the tables ON: ≥
	FC	CNT	—	—	—	16-bit counter
	FC	COLM	DCOLM	—	✓	Converting a line of data into a column of data
	FC	CRC	—	—	—	Cyclic Redundancy Check
D	FC	DABCD	DDABCD	—	✓	Converting the ASCII code into the binary-coded decimal number
	FC	DABIN	DDABIN	—	✓	Converting the signed decimal ASCII code into the signed decimal binary

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
						number
	FC	DCNT	—	—	—	32-bit counter
	FC	DEC	DDEC	—	✓	Subtracting one from the binary number
	FC	DECO	—	—	✓	Decoder
	FC	DELAY	—	—	✓	Delaying the execution of the program
	FC	—	—	DFMOV	✓	Transferring the 64-bit floating-point number
		DI	—	—	—	Disabling the interrupt
	FC	—	DIATON	—	✓	Converting the IP address of the string type into the IP address of the integer type
	FC	—	DINTOA	—	✓	Converting the IP address of the integer type into the IP address of the string type
	FC	DIS	—	—	✓	Disuniting the 16-bit data
	FC	DIV16	DIV32	—	✓	16-bit binary division/ 32-bit binary division
	FC	—	DPIDE	—	—	PID algorithm
	FC	DSW	—	—	—	Digital switch input
E	FC	EI	—	—	—	Enabling the interrupt
	FC	EMDRW	—	—	✓	Reading/Writing the MODBUS TCP data
	FC	ENCO	—	—	✓	Encoder
	FC	EPOP	—	—	✓	Reading the data into the index registers
	FC	EPUSH	—	—	✓	Storing the contents of the index registers

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
F	FC	—	F+	DF+	✓	Addition of floating-point numbers $S_1 + S_2 = D$
	FC	—	F-	DF-	✓	Subtraction of floating-point numbers $S_1 - S_2 = D$
	FC	—	F*	DF*	✓	Multiplication of floating-point numbers $S_1 * S_2 = D$
	FC	—	F/	DF/	✓	Division of floating-point numbers $S_1 / S_2 = D$
	FC	—	FACOS	—	✓	Arccosine of the floating-point number
	FC	—	FAND<	DFAND<	—	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	—	FAND<=	DFAND<=	—	Comparing the floating-point numbers ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	—	FAND<>	DFAND<>	—	Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	—	FAND=	DFAND=	—	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	—	FAND>	DFAND>	—	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	—	FAND>=	DFAND>=	—	Comparing the floating-point numbers ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	—	FASIN	—	✓	Arcsine of the floating-point number
	FC	—	FATAN	—	✓	Arctangent of the floating-point

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
F						number
	FC	—	FCMP	—	✓	Comparing the floating-point numbers
	FC	—	FCOS	—	✓	Cosine of the floating-point number
	FC	—	FCOSH	—	✓	Hyperbolic cosine of the floating-point number
	FC	—	FDEG	—	✓	Converting the radian to the degree
	FC	—	FEXP	—	✓	An exponent of the floating-point number
	FC	—	FINT	DFINT	✓	Converting the 64-bit floating-point number into the binary integer
	FC	—	FLD<	DFLD<	—	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	—	FLD<=	DFLD<=	—	Comparing the floating-point numbers ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	—	FLD<>	DFLD<>	—	Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	—	FLD=	DFLD=	—	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	—	FLD>	DFLD>	—	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	—	FLD>=	DFLD>=	—	Comparing the floating-point numbers ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	—	FLN	—	✓	Natural logarithm of the binary floating-point number

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
F	FC	—	FLOG	—	✓	Logarithm of the floating-point number
	FC	FLT	DFLT	—	✓	Converting the binary integer into the binary floating-point number
	FC	FLTD	DFLTD	—	✓	Converting the binary integer into the 64-bit floating-point number
	FC	—	FMOD	—	✓	Converting the floating-point number into the binary-coded decimal floating-point number
	FC	—	FNEG	—	✓	Reversing the sign of the 32-bit floating-point number
	FC	FOR	—	—	—	Start of the nested loop
	FC	—	FOR<	DFOR<	—	Comparing the floating-point numbers ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	—	FOR<=	DFOR<=	—	Comparing the floating-point numbers ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	—	FOR<>	DFOR<>	—	Comparing the floating-point numbers ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	—	FOR=	DFOR=	—	Comparing the floating-point numbers ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	—	FOR>	DFOR>	—	Comparing the floating-point numbers ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	—	FOR>=	DFOR>=	—	Comparing the floating-point numbers ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	—	FPOW	—	✓	A power of the floating-point number

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	—	FRAD	—	✓	Converting the degree to the radian
	FC	FREXP	—	—	✓	Converting the Binary-coded decimal floating-point number into the floating-point number
	FC	FROM	DFROM	—	✓	Reading the data from the control register in the special module
	FC	—	FSIN	—	✓	Sine of the floating-point number
	FC	—	FSINH	—	✓	Hyperbolic sine of the floating-point number
	FC	—	FSQR	—	✓	Square root of the floating-point number
	FC	—	FTAN	—	✓	Tangent of the floating-point number
	FC	—	FTANH	—	✓	Hyperbolic tangent of the floating-point number
	FC	—	FZCP	—	✓	Floating-point zone comparison
G	FC	GBIN	DGBIN	—	✓	Converting the Gray code into the binary number
	FC	GOEND	—	—	—	Jumping to END
	FC	GPWM	—	—	—	General pulse width modulation
	FC	GRY	DGRY	—	✓	Converting the binary number into the Gray code
H	FC	HABIN	DHABIN	—	✓	Converting the hexadecimal ASCII code into the hexadecimal binary number
	FC	HKY	DHKY	—	—	Hexadecimal key input
	FC	HOUR	DHOUR	—	—	Running-time meter
I	FC	IMASK	—	—	—	Controlling the interrupt
	FC	INC	DINC	—	✓	Adding one to the binary number

A4

Appendix 4. Standard Instructions (Sort by Alphabet)

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	INCD	—	—	—	Incremental drum sequencer
	FC	INT	DINT	—	✓	Converting the 32-bit floating-point number into the binary integer
J	FC	JMP	—	—	—	Unconditional jump
L	FC	LD\$<	—	—	—	Comparing the strings ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	LD\$<=	—	—	—	Comparing the strings ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	LD\$<>	—	—	—	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	LD\$=	—	—	—	Comparing the strings ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	LD\$>	—	—	—	Comparing the strings ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	LD\$>=	—	—	—	Comparing the strings ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	LD&	DLD&	—	—	ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$
	FC	LD^	DLD^	—	—	ON: $S_1 \wedge S_2 \neq 0$ OFF: $S_1 \wedge S_2 = 0$
	FC	LD	DLD	—	—	ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2 = 0$
	FC	LD<	DLD<	—	—	Comparing the values ON: $S_1 < S_2$

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
L						OFF: $S_1 \geq S_2$
	FC	LD<=	DLD<=	—	—	Comparing the values ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	LD<>	DLD<>	—	—	Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	LD=	DLD=	—	—	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	LD>	DLD>	—	—	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	LD>=	DLD>=	—	—	Comparing the values ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	LIMIT	DLIMIT	—	✓	Confining the value within the bounds
	FC	LINE	DLINE	—	✓	Converting a column of data into a line of data
	FC	LRC	—	—	—	Longitudinal parity check
M	FC	MAND	—	—	✓	Matrix AND operation
	FC	MBC	—	—	✓	Counting the bits with the value 0 or 1
	FC	MBR	—	—	✓	Rotating the matrix bits
	FC	MBRD	—	—	✓	Reading the matrix bit
	FC	MBS	—	—	✓	Shifting the matrix bits
	FC	MBWR	—	—	✓	Writing the matrix bit
	FC	MCMP	—	—	✓	Matrix comparison
	FC	MEAN	DMEAN	—	✓	Mean
	FC	MINV	—	—	✓	Inverting the matrix bits

A4

Appendix 4. Standard Instructions (Sort by Alphabet)

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	MMOV	—	—	✓	Converting the 16-bit value into the 32-bit value
	FC	MODRW	—	—	—	Reading/Writing the MODBUS data
	FC	MOR	—	—	✓	Matrix OR operation
	FC	MOV	DMOV	—	✓	Transferring the data
	FC	MOVB	—	—	✓	Transferring several bits
	FC	MREAD	—	—	—	Reading the data from the memory card into the PLC
	FC	MTR	—	—	—	Matrix input
	FC	MTWRIT	—	—	—	Writing the string into the memory card
	FC	MUL16	MUL32	—	✓	16-bit binary multiplication/ 32-bit binary multiplication
	FC	MWRIT	—	—	—	Writing the data from the PLC into the memory card
	FC	MXNR	—	—	✓	Matrix exclusive NOR operation
	FC	MXOR	—	—	✓	Matrix exclusive OR operation
	N	FC	NEG	DNEG	—	✓
FC		NEXT	—	—	—	End of the nested loop
FC		NMOV	DNMOV	—	✓	Transferring the data to several devices
FC		NSFL	—	—	✓	Shifting n registers to the left
FC		NSFR	—	—	✓	Shifting n registers to the right
O	FC	OR\$<	—	—	—	Comparing the strings ON: S1 < S2 OFF: S1 ≥ S2
	FC	OR\$<=	—	—	—	Comparing the strings ON: S1 ≤ S2

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
O						OFF: $S_1 > S_2$
	FC	OR\$<>	—	—	—	Comparing the strings ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	OR\$=	—	—	—	Comparing the strings ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$
	FC	OR\$>	—	—	—	Comparing the strings ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	OR\$>=	—	—	—	Comparing the strings ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
	FC	OR&	DOR&	—	—	ON: $S_1 \& S_2 \neq 0$ OFF: $S_1 \& S_2 = 0$
	FC	OR^	DOR^	—	—	ON: $S_1 \wedge S_2 \neq 0$ OFF: $S_1 \wedge S_2 = 0$
	FC	OR	DOR	—	—	ON: $S_1 S_2 \neq 0$ OFF: $S_1 S_2 = 0$
	FC	OR<	DOR<	—	—	Comparing the values ON: $S_1 < S_2$ OFF: $S_1 \geq S_2$
	FC	OR<=	DOR<=	—	—	Comparing the values ON: $S_1 \leq S_2$ OFF: $S_1 > S_2$
	FC	OR<>	DOR<>	—	—	Comparing the values ON: $S_1 \neq S_2$ OFF: $S_1 = S_2$
	FC	OR=	DOR=	—	—	Comparing the values ON: $S_1 = S_2$ OFF: $S_1 \neq S_2$

A4

Appendix 4. Standard Instructions (Sort by Alphabet)

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	OR>	DOR>	—	—	Comparing the values ON: $S_1 > S_2$ OFF: $S_1 \leq S_2$
	FC	OR>=	DOR>=	—	—	Comparing the values ON: $S_1 \geq S_2$ OFF: $S_1 < S_2$
P	FC	—	DPID	—	—	PID algorithm
R	FC	RAMP	—	—	—	Ramp signal
	FC	RAND	—	—	✓	Random number
	FC	RCL	DRCL	—	✓	Rotating to the left with the carry flag
	FC	RCR	DRCR	—	✓	Rotating to the right with the carry flag
	FC	REF	—	—	✓	Refreshing the I/O
	FC	RMOV	—	—	✓	Converting the 32-bit value into the 16-bit value
	FC	ROL	DROL	—	✓	Rotating to the left
	FC	ROR	DROR	—	✓	Rotating to the right
	FC	RS	—	—	—	Transmitting the user-defined communication command
	FC	RST	—	—	—	Resetting the contact or clearing the register
S	FC	SCAL	—	—	✓	Scale value operation
	FC	SCLP	DSCLP	—	✓	Parameter type of scale value operation
	FC	SEGD	—	—	✓	Seven-segment decoding
	FC	SER	DSER	—	✓	Searching the data
	FC	SFCPSE	—	—	—	SFC Run
	FC	SFCRUN	—	—	—	SFC Pause
	FC	SFCSTP	—	—	—	SFC Stop

A4

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	SFDEL	—	—	✓	Deleting the data from the data list
	FC	SFINS	—	—	✓	Inserting the data into the data list
	FC	SFL	—	—	✓	Shifting the values of the bits in the 16-bit registers by n bits to the left
	FC	SFPO	—	—	✓	Reading the latest data from the data list
	FC	SFR	—	—	✓	Shifting the values of the bits in the 16-bit registers by n bits to the right
	FC	SFRD	—	—	✓	Shifting the data and reading it from the word device
	FC	SFTL	—	—	✓	Shifting the states of the devices to the left
	FC	SFTR	—	—	✓	Shifting the states of the devices to the right
	FC	SFWR	—	—	✓	Shifting the data and writing it into the word device
	FC	SORT	DSORT	—	—	Sorting the data
	FC	SQR	DSQR	—	✓	Square root of the binary number
	FC	STMR	—	—	—	Special timer
	FC	SUM	DSUM	—	✓	Number of bits whose states are ON
	FC	SWAP	DSWAP	—	✓	Exchange the high byte with the low byte
T	FC	T-	—	—	✓	Subtracting the time
	FC	T+	—	—	✓	Adding the time
	FC	TCMP	—	—	✓	Comparing the time
	FC	TIMCHK	—	—	—	Checking time
	FC	TKOFF	—	—	✓	Disabling the cyclic task
	FC	TKON	—	—	✓	Enabling the cyclic task

A4

Appendix 4. Standard Instructions (Sort by Alphabet)

Category	FB/FC	Instruction			Pulse instruction	Description
		16-bit	32-bit	64-bit		
	FC	TKY	DTKY	—	—	Ten key input
	FC	TMR	—	—	—	16-bit timer
	FC	TMRH	—	—	—	16-bit timer
	FC	TO	DTO	—	✓	Writing the data into the control register in the special module
	FC	TRD	—	—	✓	Reading the time
	FC	TTMR	—	—	—	Teaching timer
	FC	TWR	—	—	✓	Writing the time
	FC	TZCP	—	—	✓	Time zone comparison
U	FC	UNI	—	—	✓	Uniting the 16-bit data
W	FC	WAND	DAND	—	✓	Logical AND operation
	FC	WDT	—	—	✓	Watchdog timer
	FC	WOR	DOR	—	✓	Logical OR operation
	FC	WSFL	—	—	✓	Shifting the data in the word devices to the left
	FC	WSFR	—	—	✓	Shifting the data in the word devices to the right
	FC	WSUM	DWSUM	—	✓	Getting the sum
	FC	WXNR	DXNR	—	✓	Logical exclusive NOR operation
	FC	WXOR	DXOR	—	✓	Logical exclusive OR operation
X	FC	XCH	DXCH	—	✓	Exchanging the data
Z	FC	ZCP	DZCP	—	✓	Zone comparison
	FC	ZONE	DZONE	—	✓	Controlling the zone
	FC	ZRST	—	—	✓	Resetting the zone

A4

A.5. Error Codes and Troubleshooting

When an error occurs, you can address the problem by the error codes and indicators and find out the corrective actions for troubleshooting the error. For detailed troubleshooting procedures, refer to *AH Motion – Operation Manual*.

A.5.1. Error Codes and Indicators

- **Columns**

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000A	Scan timeout (SM8: The watchdog timer error)	Stop	Blink	OFF
16#000B	The program in the PLC is damaged.	Stop	ON	OFF

↓
1

↓
2

↓
3

↓
4

A5

Items provided in the table		
1	Error code	If the error occurs in the system, the error code is generated
2	Description	The description of the error
3	CPU status	If the error occurs, the CPU stops running, keeps running, or in the status defined by users. Stop: The CPU stops running when the error occurs. Keep: The CPU keeps running when the error occurs. Self-defined: The status of the CPU can be defined by users. Please refer to section 8.2.1 in Operation Manual for more information.
4	LED indicator status	LED indicator status: If the error occurs, the LED indicator is ON, OFF, or Blinking. RUN: Operating status of the CPU ERROR: Error status of the CPU BUS FAULT: Error status of the I/O bus SYSTEM: System status of the CPU

- **LED indicators**

The AH Motion CPU can function as a motion CPU or a motion module. The effective LED indicators are different according to the applications of AH Motion CPU, either in CPU mode or in Module mode:

Mode	LED indicator	Description
CPU	RUN	Operating status of the CPU ON: The user program is being executed.

		<p>OFF: The execution of the user program stops.</p> <p>Blinking: The CPU runs in debug mode.</p>
	ERROR	<p>Error status of the CPU</p> <p>ON: A serious error occurs in the CPU.</p> <p>OFF: The system is normal.</p> <p>Blinking: A slight error occurs in the CPU.</p>
	BUS FAULT	<p>Error status of the I/O bus</p> <p>ON: A serious error occurs in the I/O bus.</p> <p>OFF: The I/O bus is normal.</p> <p>Blinking: A slight error occurs in the I/O bus.</p>
	SYSTEM	<p>System status of the CPU module</p> <p>ON: The external input/output is forced ON/OFF.</p> <p>OFF: The system is in the default status.</p> <p>Blinking: The CPU module is being reset./The retained values in the devices are being cleared .</p>
Module	RUN	<p>Operating status of the motion CPU functioning as a motion module</p> <p>ON: The user program is being executed.</p> <p>OFF: The execution of the user program stops.</p> <p>Blinking: The motion module runs in debug mode.</p>
	ERROR	<p>Error status of the motion CPU functioning as a motion module</p> <p>ON: A serious error occurs in the module.</p> <p>OFF: The system is normal.</p> <p>Blinking: A slight error occurs in the module.</p>

AHxxEMC-5A

After a program is written into an AH Motion series CPU, the ERROR LED indicator will blink and an error flag will be ON if an error occurs in main program or motion subroutine. The reason for the error occurring in the main program or motion subroutine may be that the use of operands (devices) is incorrect, syntax is incorrect, or the setting of motion parameters is incorrect. You can know the reasons for the errors occurring in an AH Motion series CPU by means of the error codes (hexadecimal codes) stored in error registers.

■ **Error flags and registers:**

SM*: Special auxiliary relay SR*: Special data register AR*: Axis register AM*: Axis flag	Program error	Motion error
	POU	mn=10~41 (10: 1 st axis; 41: 32 nd axis)
Error flag	-	AMmn49
Operation error	SM0	-
The operation error is locked	SM1	-
Syntax (Instruction/Operand) check error	SM5	-
Operation error code	SR0	-
Operation error address (step)	SR1/SR2	-
Syntax check error code	SR4	ARmn41
Syntax check error address (step)	SR5/SR6	-

*Note: you can refer to *AH Motion – Operation Manual* for the detailed explanation of SM, SR, AM and AR.

■ **Error codes and indicators**

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000A	Scan timeout (SM8: The watchdog timer error)	Stop	Blinking	Keep
16#000B	The program in the PLC is damaged.	Stop	ON	Keep
16#000C	The program downloaded to the PLC is incorrect.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000D	The CPU parameter is damaged.	Stop	ON	Keep
16#000E	The program or the parameter is being downloaded, and therefore the PLC can not run.	Stop	Blinking	Keep
16#000F	The original program in the PLC is damaged.	Continue	Keep	Keep
16#0010	The access to the memory in the CPU is denied.	Stop	ON	Keep
16#0011	The PLC ID is incorrect. (SM9)	Continue	ON	Keep
16#0012	The PLC password is incorrect.	Continue	ON	Keep
16#0013	The I/O module can not run/stop. (SM10)	Stop	Keep	ON
16#0014	The procedure of restoring the system can not be executed. (SM9)	Stop	ON	ON
16#0015	The module table is incorrect. (SM10)	Stop	ON	Keep
16#0016	The module setting is incorrect. (SM10)	Stop	ON	Keep
16#0017	The device which is associated with the data register is incorrect. (SM10)	Stop	ON	Keep
16#0018	The serial port is abnormal. (SM9)	Continue	Blinking	Keep
16#0019	The USB is abnormal. (SM9)	Continue	Blinking	Keep
16#001A	The contents of the system backup file (.dup file) are incorrect.	Continue	Blinking	Keep
16#001B	Timed interrupt 0 is set incorrectly.	Stop	ON	Keep
16#001C	Timed interrupt 1 is set incorrectly.	Stop	ON	Keep
16#001D	Timed interrupt 2 is set incorrectly.	Stop	ON	Keep
16#001E	Timed interrupt 3 is set incorrectly.	Stop	ON	Keep
16#001F	The watchdog timer is set incorrectly.	Stop	ON	Keep
16#0020	The setting of the fixed scan time is incorrect.	Stop	ON	Keep
16#0021	The setting of the fixed scan time is incorrect.	Stop	ON	Keep
16#0022	The CPU parameter downloaded to the PLC is incorrect.	Stop	ON	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#0023	The Y state (STOP->RUN) section in the PLC Parameter Setting window is set incorrectly.	Stop	ON	Keep
16#0026	The Communication Ratio box in the Communication Loading of Scan Time (%) section in the PLC Parameter Setting window is set incorrectly.	Stop	ON	Keep
16#0027	The latching auxiliary relay range which is set is incorrect.	Stop	ON	Keep
16#0028	The latching data register range which is set is incorrect.	Stop	ON	Keep
16#0029	The latching timer range which is set is incorrect.	Stop	ON	Keep
16#002A	The latching counter range which is set is incorrect.	Stop	ON	Keep
16#002B	The latching 32-bit counter range which is set is incorrect.	Stop	ON	Keep
16#0033	The communication setting of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0034	The setting of the station address of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0035	The setting of the communication type of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0038	The communication setting of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#0039	The setting of the station address of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#003A	The setting of the communication type of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#0050	The memories in the latched special auxiliary relays are abnormal.	Stop	ON	Keep
16#0051	The latched special data registers are abnormal.	Stop	ON	Keep
16#0052	The memories in the latched auxiliary relays are abnormal.	Stop	ON	Keep
16#0053	The latched timers are abnormal.	Stop	ON	Keep
16#0054	The latched counters are abnormal.	Stop	ON	Keep
16#0055	The latched 32-bit counters are abnormal.	Stop	ON	Keep
16#0056	The memories in the latched timers are abnormal.	Stop	ON	Keep
16#0057	The memories in the latched counters are abnormal.	Stop	ON	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#0058	The memories in the latched 32-bit counters are abnormal.	Stop	ON	Keep
16#0059	The latched data registers are abnormal.	Stop	ON	Keep
16#005A	The latched working registers are abnormal.	Stop	ON	Keep
16#005E	The memory card is initialized incorrectly. (SM453)	Continue	Blinking	Keep
16#005F	The data is read from the inexistent file in the memory card, or the data is written into the inexistent file in the memory card. (SM453)	Continue	Blinking	Keep
16#0061	The capacity of the memory card is not large enough. (SM453)	Continue	Blinking	Keep
16#0062	The memory card is write protected. (SM453)	Continue	Blinking	Keep
16#0063	An error occurs when the data is written into the memory card. (SM453)	Continue	Blinking	Keep
16#0064	The file in the memory card can not be read. (SM453)	Continue	Blinking	Keep
16#0065	The file in the memory card is a read-only file. (SM453)	Continue	Blinking	Keep
16#0066	An error occurs when the system is backedup.	Continue	Blinking	Keep
16#1401	An error occurs when the data in the I/O module is accessed. (SM9)	Stop	Keep	ON
16#1402	The actual arrangement of the I/O modules is not consistent with the module table. (SM9)	Stop	Keep	ON
16#1403	An error occurs when the data is read from the module. (SM9)	Stop	Keep	ON
16#1405	The setting parameter of the module is not found. (SM9)	Stop	Keep	ON
16#140B	The number of network modules exceeds the limit. (SM9)	Stop	Keep	ON
16#140C	The checksum of the high-speed data exchange is incorrect.	Stop	Keep	ON
16#140D	The ID of the actual power supply module is not the same as the ID of the power supply module set in HWCONFIG. (SM9)	Stop	Keep	ON
16#140E	The amount of data exchanged at a high speed exceeds the maximum amount supported.	Stop	Keep	ON
16#140F	High-speed data exchange error	Stop	Keep	ON

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#1801	There is no interrupt service routine in the CPU module.	Continue	Keep	Keep
16#2000	There is no END in the program in the PLC. (SM5)	Stop	Blinking	Keep
16#2001	The program is incorrect. There is a syntax error.	Stop	Blinking	Keep
16#2002	GOEND is used incorrectly. (SM5)	Stop	Blinking	Keep
16#2003	The devices used in the program exceed the range. (SM0/SM5)	Self-defined	Blinking	Keep
16#2004	The part of the program specified by the label used in CJ/JMP is incorrect, or the label is used repeatedly. (SM0/SM5)	Stop	Blinking	Keep
16#2005	The N value used in MC is not the same as the corresponding N value used in MCR, or the number of N values used in MC is not the same as the number of N values used in MCR. (SM5)	Stop	Blinking	Keep
16#2006	The N values used in MC do not start from 0, or the N values used in MC are not continuous. (SM5)	Stop	Blinking	Keep
16#2007	The operands used in ZRST are not used properly. (SM5)	Stop	Blinking	Keep
16#200A	Invalid instruction (SM5)	Stop	Blinking	Keep
16#200B	The operand n or the other constant operands exceed the range. (SM0/SM5)	Self-defined	Blinking	Keep
16#200C	The operands overlap. (SM0/SM5)	Self-defined	Blinking	Keep
16#200D	An error occurs when the binary number is converted into the binary-coded decimal number. (SM0/SM5)	Self-defined	Blinking	Keep
16#200E	The string does not end with 0x00. (SM0/SM5)	Self-defined	Blinking	Keep
16#200F	The instruction does not support the modification by an index register. (SM5)	Stop	Blinking	Keep
16#2010	1. The instruction does not support the device. 2. Encoding error 3. The instruction is a 16-bit instruction, but the constant operand is a 32-bit code. (SM5)	Stop	Blinking	Keep
16#2011	The number of operands is incorrect. (SM5)	Stop	Blinking	Keep
16#2012	Incorrect division operation (SM0/SM5).	Self-defined	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2013	The value exceeds the range of values which can be represented by the floating-point numbers. (SM0/SM5)	Self-defined	Blinking	Keep
16#2014	The task designated by TKON/TKOFF is incorrect, or exceeds the range. (SM5)	Stop	Blinking	Keep
16#2015	There are more than 32 levels of nested program structures supported by CALL. (SM0)	Self-defined	Blinking	Keep
16#2016	There are more than 32 levels of nested program structures supported by FOR/NEXT. (SM0/SM5)	Self-defined	Blinking	Keep
16#2017	The number of times FOR is used is different from the number of times NEXT is used. (SM5)	Stop	Blinking	Keep
16#2018	There is a label after FEND, but there is no SRET. There is SRET, but there is no label. (SM5)	Stop	Blinking	Keep
16#2019	The interrupt task is not after FEND. (SM5)	Stop	Blinking	Keep
16#201A	IRET/SRET is not after FEND. (SM5)	Stop	Blinking	Keep
16#201B	There is an interrupt task, but there is no IRET. There is IRET, but there is not interrupt task. (SM5)	Stop	Blinking	Keep
16#201C	End is not at the end of the program. (SM5)	Stop	Blinking	Keep
16#201D	There is CALL, but there is no MAR. (SM5)	Stop	Blinking	Keep
16#201E	The function code used in MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#201F	The length of the data set in MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#2020	The communication command received by using MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#2021	The checksum of the command received is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#2022	The format of the command used in MODRW does not conform to the ASCII format. (SM102/SM103)	Self-defined	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2023	There is a communication timeout when MODRW is executed. (SM120/SM103)	Self-defined	Blinking	Keep
16#2024	The setting value of the communication timeout is invalid. (SM120/SM103)	Self-defined	Blinking	Keep
16#2025	There is a communication timeout when RS is executed. (SM120/SM103)	Self-defined	Blinking	Keep
16#2026	The interrupt number used in RS is incorrect.	Self-defined	Keep	Keep
16#2027	The execution of FWD is abnormal.	Self-defined	Blinking	Keep
16#2028	The execution of REV is abnormal.	Self-defined	Blinking	Keep
16#2029	The execution of STOP is abnormal.	Self-defined	Blinking	Keep
16#202A	The execution of RSDT is abnormal.	Self-defined	Blinking	Keep
16#202B	The execution of RSTEF is abnormal.	Self-defined	Blinking	Keep
16#202C	I/O interrupt service routine 0 does not exist.	Stop	Blinking	Keep
16#202D	I/O interrupt service routine 1 does not exist.	Stop	Blinking	Keep
16#202E	I/O interrupt service routine 2 does not exist.	Stop	Blinking	Keep
16#202F	I/O interrupt service routine 3 does not exist.	Stop	Blinking	Keep
16#2030	I/O interrupt service routine 4 does not exist.	Stop	Blinking	Keep
16#2031	I/O interrupt service routine 5 does not exist.	Stop	Blinking	Keep
16#2032	I/O interrupt service routine 6 does not exist.	Stop	Blinking	Keep
16#2033	I/O interrupt service routine 7 does not exist.	Stop	Blinking	Keep
16#2034	I/O interrupt service routine 8 does not exist.	Stop	Blinking	Keep
16#2035	I/O interrupt service routine 9 does not exist.	Stop	Blinking	Keep
16#2036	I/O interrupt service routine 10 does not exist.	Stop	Blinking	Keep
16#2037	I/O interrupt service routine 11 does not exist.	Stop	Blinking	Keep
16#2038	I/O interrupt service routine 12 does not exist.	Stop	Blinking	Keep
16#2039	I/O interrupt service routine 13 does not exist.	Stop	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#203A	I/O interrupt service routine 14 does not exist.	Stop	Blinking	Keep
16#203B	I/O interrupt service routine 15 does not exist.	Stop	Blinking	Keep
16#203C	I/O interrupt service routine 16 does not exist.	Stop	Blinking	Keep
16#203D	I/O interrupt service routine 17 does not exist.	Stop	Blinking	Keep
16#203E	I/O interrupt service routine 18 does not exist.	Stop	Blinking	Keep
16#203F	I/O interrupt service routine 19 does not exist.	Stop	Blinking	Keep
16#2040	I/O interrupt service routine 20 does not exist.	Stop	Blinking	Keep
16#2041	I/O interrupt service routine 21 does not exist.	Stop	Blinking	Keep
16#2042	I/O interrupt service routine 22 does not exist.	Stop	Blinking	Keep
16#2043	I/O interrupt service routine 23 does not exist.	Stop	Blinking	Keep
16#2044	I/O interrupt service routine 24 does not exist.	Stop	Blinking	Keep
16#2045	I/O interrupt service routine 25 does not exist.	Stop	Blinking	Keep
16#2046	I/O interrupt service routine 26 does not exist.	Stop	Blinking	Keep
16#2047	I/O interrupt service routine 27 does not exist.	Stop	Blinking	Keep
16#2048	I/O interrupt service routine 28 does not exist.	Stop	Blinking	Keep
16#2049	I/O interrupt service routine 29 does not exist.	Stop	Blinking	Keep
16#204A	I/O interrupt service routine 30 does not exist.	Stop	Blinking	Keep
16#204B	I/O interrupt service routine 31 does not exist.	Stop	Blinking	Keep
16#2054	External interrupt service routine 40 does not exist.	Stop	Blinking	Keep
16#2055	External interrupt service routine 41 does not exist.	Stop	Blinking	Keep
16#2056	External interrupt service routine 42 does not exist.	Stop	Blinking	Keep
16#2057	External interrupt service routine 43 does not exist.	Stop	Blinking	Keep
16#2058	External interrupt service routine 44 does not exist.	Stop	Blinking	Keep
16#2059	External interrupt service routine 45 does not exist.	Stop	Blinking	Keep
16#205A	External interrupt service routine 46 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#205B	External interrupt service routine 47 does not exist.	Stop	Blinking	Keep
16#205C	External interrupt service routine 48 does not exist.	Stop	Blinking	Keep
16#205D	External interrupt service routine 49 does not exist.	Stop	Blinking	Keep
16#205E	External interrupt service routine 50 does not exist.	Stop	Blinking	Keep
16#205F	External interrupt service routine 51 does not exist.	Stop	Blinking	Keep
16#2060	External interrupt service routine 52 does not exist.	Stop	Blinking	Keep
16#2061	External interrupt service routine 53 does not exist.	Stop	Blinking	Keep
16#2062	External interrupt service routine 54 does not exist.	Stop	Blinking	Keep
16#2063	External interrupt service routine 55 does not exist.	Stop	Blinking	Keep
16#2064	External interrupt service routine 56 does not exist.	Stop	Blinking	Keep
16#2065	External interrupt service routine 57 does not exist.	Stop	Blinking	Keep
16#2066	External interrupt service routine 58 does not exist.	Stop	Blinking	Keep
16#2067	External interrupt service routine 59 does not exist.	Stop	Blinking	Keep
16#2068	External interrupt service routine 60 does not exist.	Stop	Blinking	Keep
16#2069	External interrupt service routine 61 does not exist.	Stop	Blinking	Keep
16#206A	External interrupt service routine 62 does not exist.	Stop	Blinking	Keep
16#206B	External interrupt service routine 63 does not exist.	Stop	Blinking	Keep
16#206C	External interrupt service routine 64 does not exist.	Stop	Blinking	Keep
16#206D	External interrupt service routine 65 does not exist.	Stop	Blinking	Keep
16#206E	External interrupt service routine 66 does not exist.	Stop	Blinking	Keep
16#206F	External interrupt service routine 67 does not exist.	Stop	Blinking	Keep
16#2070	External interrupt service routine 68 does not exist.	Stop	Blinking	Keep
16#2071	External interrupt service routine 69 does not exist.	Stop	Blinking	Keep
16#2072	External interrupt service routine 70 does not exist.	Stop	Blinking	Keep
16#2073	External interrupt service routine 71 does not exist.	Stop	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2074	External interrupt service routine 72 does not exist.	Stop	Blinking	Keep
16#2075	External interrupt service routine 73 does not exist.	Stop	Blinking	Keep
16#2076	External interrupt service routine 74 does not exist.	Stop	Blinking	Keep
16#2077	External interrupt service routine 75 does not exist.	Stop	Blinking	Keep
16#2078	External interrupt service routine 76 does not exist.	Stop	Blinking	Keep
16#2079	External interrupt service routine 77 does not exist.	Stop	Blinking	Keep
16#207A	External interrupt service routine 78 does not exist.	Stop	Blinking	Keep
16#207B	External interrupt service routine 79 does not exist.	Stop	Blinking	Keep
16#207C	External interrupt service routine 80 does not exist.	Stop	Blinking	Keep
16#207D	External interrupt service routine 81 does not exist.	Stop	Blinking	Keep
16#207E	External interrupt service routine 82 does not exist.	Stop	Blinking	Keep
16#207F	External interrupt service routine 83 does not exist.	Stop	Blinking	Keep
16#2080	External interrupt service routine 84 does not exist.	Stop	Blinking	Keep
16#2081	External interrupt service routine 85 does not exist.	Stop	Blinking	Keep
16#2082	External interrupt service routine 86 does not exist.	Stop	Blinking	Keep
16#2083	External interrupt service routine 87 does not exist.	Stop	Blinking	Keep
16#2084	External interrupt service routine 88 does not exist.	Stop	Blinking	Keep
16#2085	External interrupt service routine 89 does not exist.	Stop	Blinking	Keep
16#2086	External interrupt service routine 90 does not exist.	Stop	Blinking	Keep
16#2087	External interrupt service routine 91 does not exist.	Stop	Blinking	Keep
16#2088	External interrupt service routine 92 does not exist.	Stop	Blinking	Keep
16#2089	External interrupt service routine 93 does not exist.	Stop	Blinking	Keep
16#208A	External interrupt service routine 94 does not exist.	Stop	Blinking	Keep
16#208B	External interrupt service routine 95 does not exist.	Stop	Blinking	Keep
16#208C	External interrupt service routine 96 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#208D	External interrupt service routine 97 does not exist.	Stop	Blinking	Keep
16#208E	External interrupt service routine 98 does not exist.	Stop	Blinking	Keep
16#208F	External interrupt service routine 99 does not exist.	Stop	Blinking	Keep
16#2090	External interrupt service routine 100 does not exist.	Stop	Blinking	Keep
16#2091	External interrupt service routine 101 does not exist.	Stop	Blinking	Keep
16#2092	External interrupt service routine 102 does not exist.	Stop	Blinking	Keep
16#2093	External interrupt service routine 103 does not exist.	Stop	Blinking	Keep
16#2094	External interrupt service routine 104 does not exist.	Stop	Blinking	Keep
16#2095	External interrupt service routine 105 does not exist.	Stop	Blinking	Keep
16#2096	External interrupt service routine 106 does not exist.	Stop	Blinking	Keep
16#2097	External interrupt service routine 107 does not exist.	Stop	Blinking	Keep
16#2098	External interrupt service routine 108 does not exist.	Stop	Blinking	Keep
16#2099	External interrupt service routine 109 does not exist.	Stop	Blinking	Keep
16#209A	External interrupt service routine 110 does not exist.	Stop	Blinking	Keep
16#209B	External interrupt service routine 111 does not exist.	Stop	Blinking	Keep
16#209C	External interrupt service routine 112 does not exist.	Stop	Blinking	Keep
16#209D	External interrupt service routine 113 does not exist.	Stop	Blinking	Keep
16#209E	External interrupt service routine 114 does not exist.	Stop	Blinking	Keep
16#209F	External interrupt service routine 115 does not exist.	Stop	Blinking	Keep
16#20A0	External interrupt service routine 116 does not exist.	Stop	Blinking	Keep
16#20A1	External interrupt service routine 117 does not exist.	Stop	Blinking	Keep
16#20A2	External interrupt service routine 118 does not exist.	Stop	Blinking	Keep
16#20A3	External interrupt service routine 119 does not exist.	Stop	Blinking	Keep
16#20A4	External interrupt service routine 120 does not exist.	Stop	Blinking	Keep
16#20A5	External interrupt service routine 121 does not exist.	Stop	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20A6	External interrupt service routine 122 does not exist.	Stop	Blinking	Keep
16#20A7	External interrupt service routine 123 does not exist.	Stop	Blinking	Keep
16#20A8	External interrupt service routine 124 does not exist.	Stop	Blinking	Keep
16#20A9	External interrupt service routine 125 does not exist.	Stop	Blinking	Keep
16#20AA	External interrupt service routine 126 does not exist.	Stop	Blinking	Keep
16#20AB	External interrupt service routine 127 does not exist.	Stop	Blinking	Keep
16#20AC	External interrupt service routine 128 does not exist.	Stop	Blinking	Keep
16#20AD	External interrupt service routine 129 does not exist.	Stop	Blinking	Keep
16#20AE	External interrupt service routine 130 does not exist.	Stop	Blinking	Keep
16#20AF	External interrupt service routine 131 does not exist.	Stop	Blinking	Keep
16#20B0	External interrupt service routine 132 does not exist.	Stop	Blinking	Keep
16#20B1	External interrupt service routine 133 does not exist.	Stop	Blinking	Keep
16#20B2	External interrupt service routine 134 does not exist.	Stop	Blinking	Keep
16#20B3	External interrupt service routine 135 does not exist.	Stop	Blinking	Keep
16#20B4	External interrupt service routine 136 does not exist.	Stop	Blinking	Keep
16#20B5	External interrupt service routine 137 does not exist.	Stop	Blinking	Keep
16#20B6	External interrupt service routine 138 does not exist.	Stop	Blinking	Keep
16#20B7	External interrupt service routine 139 does not exist.	Stop	Blinking	Keep
16#20B8	External interrupt service routine 140 does not exist.	Stop	Blinking	Keep
16#20B9	External interrupt service routine 141 does not exist.	Stop	Blinking	Keep
16#20BA	External interrupt service routine 142 does not exist.	Stop	Blinking	Keep
16#20BB	External interrupt service routine 143 does not exist.	Stop	Blinking	Keep
16#20BC	External interrupt service routine 144 does not exist.	Stop	Blinking	Keep
16#20BD	External interrupt service routine 145 does not exist.	Stop	Blinking	Keep
16#20BE	External interrupt service routine 146 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20BF	External interrupt service routine 147 does not exist.	Stop	Blinking	Keep
16#20C0	External interrupt service routine 148 does not exist.	Stop	Blinking	Keep
16#20C1	External interrupt service routine 149 does not exist.	Stop	Blinking	Keep
16#20C2	External interrupt service routine 150 does not exist.	Stop	Blinking	Keep
16#20C3	External interrupt service routine 151 does not exist.	Stop	Blinking	Keep
16#20C4	External interrupt service routine 152 does not exist.	Stop	Blinking	Keep
16#20C5	External interrupt service routine 153 does not exist.	Stop	Blinking	Keep
16#20C6	External interrupt service routine 154 does not exist.	Stop	Blinking	Keep
16#20C7	External interrupt service routine 155 does not exist.	Stop	Blinking	Keep
16#20C8	External interrupt service routine 156 does not exist.	Stop	Blinking	Keep
16#20C9	External interrupt service routine 157 does not exist.	Stop	Blinking	Keep
16#20CA	External interrupt service routine 158 does not exist.	Stop	Blinking	Keep
16#20CB	External interrupt service routine 159 does not exist.	Stop	Blinking	Keep
16#20CC	External interrupt service routine 160 does not exist.	Stop	Blinking	Keep
16#20CD	External interrupt service routine 161 does not exist.	Stop	Blinking	Keep
16#20CE	External interrupt service routine 162 does not exist.	Stop	Blinking	Keep
16#20CF	External interrupt service routine 163 does not exist.	Stop	Blinking	Keep
16#20D0	External interrupt service routine 164 does not exist.	Stop	Blinking	Keep
16#20D1	External interrupt service routine 165 does not exist.	Stop	Blinking	Keep
16#20D2	External interrupt service routine 166 does not exist.	Stop	Blinking	Keep
16#20D3	External interrupt service routine 167 does not exist.	Stop	Blinking	Keep
16#20D4	External interrupt service routine 168 does not exist.	Stop	Blinking	Keep
16#20D5	External interrupt service routine 169 does not exist.	Stop	Blinking	Keep
16#20D6	External interrupt service routine 170 does not exist.	Stop	Blinking	Keep
16#20D7	External interrupt service routine 171 does not exist.	Stop	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20D8	External interrupt service routine 172 does not exist.	Stop	Blinking	Keep
16#20D9	External interrupt service routine 173 does not exist.	Stop	Blinking	Keep
16#20DA	External interrupt service routine 174 does not exist.	Stop	Blinking	Keep
16#20DB	External interrupt service routine 175 does not exist.	Stop	Blinking	Keep
16#20DC	External interrupt service routine 176 does not exist.	Stop	Blinking	Keep
16#20DD	External interrupt service routine 177 does not exist.	Stop	Blinking	Keep
16#20DE	External interrupt service routine 178 does not exist.	Stop	Blinking	Keep
16#20DF	External interrupt service routine 179 does not exist.	Stop	Blinking	Keep
16#20E0	External interrupt service routine 180 does not exist.	Stop	Blinking	Keep
16#20E1	External interrupt service routine 181 does not exist.	Stop	Blinking	Keep
16#20E2	External interrupt service routine 182 does not exist.	Stop	Blinking	Keep
16#20E3	External interrupt service routine 183 does not exist.	Stop	Blinking	Keep
16#20E4	External interrupt service routine 184 does not exist.	Stop	Blinking	Keep
16#20E5	External interrupt service routine 185 does not exist.	Stop	Blinking	Keep
16#20E6	External interrupt service routine 186 does not exist.	Stop	Blinking	Keep
16#20E7	External interrupt service routine 187 does not exist.	Stop	Blinking	Keep
16#20E8	External interrupt service routine 188 does not exist.	Stop	Blinking	Keep
16#20E9	External interrupt service routine 189 does not exist.	Stop	Blinking	Keep
16#20EA	External interrupt service routine 190 does not exist.	Stop	Blinking	Keep
16#20EB	External interrupt service routine 191 does not exist.	Stop	Blinking	Keep
16#20EC	External interrupt service routine 192 does not exist.	Stop	Blinking	Keep
16#20ED	External interrupt service routine 193 does not exist.	Stop	Blinking	Keep
16#20EE	External interrupt service routine 194 does not exist.	Stop	Blinking	Keep
16#20EF	External interrupt service routine 195 does not exist.	Stop	Blinking	Keep
16#20F0	External interrupt service routine 196 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20F1	External interrupt service routine 197 does not exist.	Stop	Blinking	Keep
16#20F2	External interrupt service routine 198 does not exist.	Stop	Blinking	Keep
16#20F3	External interrupt service routine 199 does not exist.	Stop	Blinking	Keep
16#20F4	External interrupt service routine 200 does not exist.	Stop	Blinking	Keep
16#20F5	External interrupt service routine 201 does not exist.	Stop	Blinking	Keep
16#20F6	External interrupt service routine 202 does not exist.	Stop	Blinking	Keep
16#20F7	External interrupt service routine 203 does not exist.	Stop	Blinking	Keep
16#20F8	External interrupt service routine 204 does not exist.	Stop	Blinking	Keep
16#20F9	External interrupt service routine 205 does not exist.	Stop	Blinking	Keep
16#20FA	External interrupt service routine 206 does not exist.	Stop	Blinking	Keep
16#20FB	External interrupt service routine 207 does not exist.	Stop	Blinking	Keep
16#20FC	External interrupt service routine 208 does not exist.	Stop	Blinking	Keep
16#20FD	External interrupt service routine 209 does not exist.	Stop	Blinking	Keep
16#20FE	External interrupt service routine 210 does not exist.	Stop	Blinking	Keep
16#20FF	External interrupt service routine 211 does not exist.	Stop	Blinking	Keep
16#2100	External interrupt service routine 212 does not exist.	Stop	Blinking	Keep
16#2101	External interrupt service routine 213 does not exist.	Stop	Blinking	Keep
16#2102	External interrupt service routine 214 does not exist.	Stop	Blinking	Keep
16#2103	External interrupt service routine 215 does not exist.	Stop	Blinking	Keep
16#2104	External interrupt service routine 216 does not exist.	Stop	Blinking	Keep
16#2105	External interrupt service routine 217 does not exist.	Stop	Blinking	Keep
16#2106	External interrupt service routine 218 does not exist.	Stop	Blinking	Keep
16#2107	External interrupt service routine 219 does not exist.	Stop	Blinking	Keep
16#2108	External interrupt service routine 220 does not exist.	Stop	Blinking	Keep
16#2109	External interrupt service routine 221 does not exist.	Stop	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#210A	External interrupt service routine 222 does not exist.	Stop	Blinking	Keep
16#210B	External interrupt service routine 223 does not exist.	Stop	Blinking	Keep
16#210C	External interrupt service routine 224 does not exist.	Stop	Blinking	Keep
16#210D	External interrupt service routine 225 does not exist.	Stop	Blinking	Keep
16#210E	External interrupt service routine 226 does not exist.	Stop	Blinking	Keep
16#210F	External interrupt service routine 227 does not exist.	Stop	Blinking	Keep
16#2110	External interrupt service routine 228 does not exist.	Stop	Blinking	Keep
16#2111	External interrupt service routine 229 does not exist.	Stop	Blinking	Keep
16#2112	External interrupt service routine 230 does not exist.	Stop	Blinking	Keep
16#2113	External interrupt service routine 231 does not exist.	Stop	Blinking	Keep
16#2114	External interrupt service routine 232 does not exist.	Stop	Blinking	Keep
16#2115	External interrupt service routine 233 does not exist.	Stop	Blinking	Keep
16#2116	External interrupt service routine 234 does not exist.	Stop	Blinking	Keep
16#2117	External interrupt service routine 235 does not exist.	Stop	Blinking	Keep
16#2118	External interrupt service routine 236 does not exist.	Stop	Blinking	Keep
16#2119	External interrupt service routine 237 does not exist.	Stop	Blinking	Keep
16#211A	External interrupt service routine 238 does not exist.	Stop	Blinking	Keep
16#211B	External interrupt service routine 239 does not exist.	Stop	Blinking	Keep
16#211C	External interrupt service routine 240 does not exist.	Stop	Blinking	Keep
16#211D	External interrupt service routine 241 does not exist.	Stop	Blinking	Keep
16#211E	External interrupt service routine 242 does not exist.	Stop	Blinking	Keep
16#211F	External interrupt service routine 243 does not exist.	Stop	Blinking	Keep
16#2120	External interrupt service routine 244 does not exist.	Stop	Blinking	Keep
16#2121	External interrupt service routine 245 does not exist.	Stop	Blinking	Keep
16#2122	External interrupt service routine 246 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2123	External interrupt service routine 247 does not exist.	Stop	Blinking	Keep
16#2124	External interrupt service routine 248 does not exist.	Stop	Blinking	Keep
16#2125	External interrupt service routine 249 does not exist.	Stop	Blinking	Keep
16#2126	External interrupt service routine 250 does not exist.	Stop	Blinking	Keep
16#2127	External interrupt service routine 251 does not exist.	Stop	Blinking	Keep
16#2128	An action in a sequential function chart is incorrectly assigned qualifiers related to time.	Stop	Blinking	Keep
16#2129	The modifier R is assigned to an action in a sequential function chart incorrectly.	Stop	Blinking	Keep
16#3047	No SD card is detected; or the specified file does not exist in the SD card.	Continue	Keep	Keep
16#3100	Input parameters exceed the available setting range.	Continue	Blinking	Keep
16#3102	An error occurs in a sub-function block inside the function block.	Continue	Blinking	Keep
16#3103	The distance between the detecting sensors used for identifying exceptional bags is a negative value.	Continue	Blinking	Keep
16#3104	Phasing is executed again before the previous phasing is completed.	Continue	Blinking	Keep
16#3105	Superimposing is executed again before the previous superimposing is completed.	Continue	Blinking	Keep
16#3106	Chain position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3107	Film axis position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3108	Knife position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3400	Motion axis number is incorrect.	Continue	Keep	Keep
16#3401	SDO Data Type setting error in DFB_ECATServoWrite.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3404	The number of the counting channel exceeds the available setting range.	Continue	Keep	Keep
16#3405	A negative value is given to <i>Velocity</i> .	Continue	Blinking	Keep
16#3407	The speed of returning to zero (V_{RT}) is incorrect.	Continue	Blinking	Keep
16#3408	The deceleration of returning to zero (V_{CR}) is incorrect.	Continue	Blinking	Keep
16#340A	Homing mode setting error.	Continue	Blinking	Keep
16#340B	Target distance is 0.	Continue	Blinking	Keep
16#340E	Incorrect value given to the source of the comparison in DFB_Compare.	Continue	Keep	Keep
16#3410	User unit setting error; or the output pulse type setting error.	Continue	Blinking	Keep
16#3411	Velocity override factor setting error.	Continue	Blinking	Keep
16#3412	Continuous interpolation velocity setting error.	Continue	Blinking	Keep
16#3414	Pulse type setting error in DFB_HCnt.	Continue	Keep	Keep
16#3415	Comparison condition setting error in DFB_Compare.	Continue	Keep	Keep
16#3419	Master axis position is negative value.	Continue	Blinking	Keep
16#341B	Maximum speed setting error.	Continue	Blinking	Keep
16#3422	Output device setting error in DFB_Compare.	Continue	Keep	Keep
16#3429	G-code compiling error.	Continue	Blinking	Keep
16#342A	G-code program source error.	Continue	Blinking	Keep
16#342B	G-code ID setting error.	Continue	Blinking	Keep
16#3431	Motion axis number is repeated in the same group in DFB_GroupEnable.	Continue	Blinking	Keep
16#3432	The specified group number does not exist.	Continue	Blinking	Keep
16#3433	The number of axes is insufficient for the specified group axes motion.	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3434	DFB_GroupDisable is executed when group motion is in progress.	Continue	Blinking	Keep
16#3435	Motion axis number is repeated between different groups when DFB_GroupEnable is enabled.	Continue	Blinking	Keep
16#3436	The axis number of the first order should be a positive number other than 0.	Continue	Blinking	Keep
16#3437	The group number exceeds the setting range.	Continue	Blinking	Keep
16#3438	The designated group is in "ErrorStop"	Continue	Blinking	Keep
16#3461	The required communication parameters for PDO settings are not specified	Continue	Blinking	Keep
16#3505	An error occurs when writing cam data.	Continue	Blinking	Keep
16#3506	The axis is in "Coordinated" state.	Continue	Blinking	Keep
16#3507	The axis is in "ErrorStop" state.	Continue	Blinking	Keep
16#3508	The axis is not in "StandStill" state.	Continue	Blinking	Keep
16#3600	The axis is currently in group motion	Continue	Blinking	Keep
16#3601	The limit of the number of buffering instructions is reached	Continue	Blinking	Keep
16#3602	A multiple instructions which are not allowed to be executed at the same time are executed.	Continue	Blinking	Keep
16#3603	Buffermode parameter setting error	Continue	Blinking	Keep
16#3604	Errors occur on the motion direction of the function block	Continue	Blinking	Keep
16#3605	P1 exceeds the available range	Continue	Blinking	Keep
16#3606	P2 exceeds the available range	Continue	Blinking	Keep
16#3607	V1 exceeds the available range	Continue	Blinking	Keep
16#3608	V2 exceeds the available range	Continue	Blinking	Keep
16#3612	LSP is On in positive travel	Continue	Blinking	Keep
16#3613	LSN is On in negative travel	Continue	Blinking	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3614	The servo limit is exceeded.	Continue	Blinking	Keep
16#3615	SuperImposing is on-going	Continue	Blinking	Keep
16#3800	Motion network disconnected during the execution of the instruction.	Continue	Blinking	Keep
16#3801	Axis errors occur in motion network	Continue	Blinking	Keep
16#3900	Failed to re-connect to the motion network.	Continue	Blinking	Keep
16#3902	The servo cannot be powered on. (ServoOn failed)	Continue	Blinking	Keep
16#3903	The servo cannot be powered off. (ServoOff failed)	Continue	Blinking	Keep
16#3904	Motion network master can not read Slave parameters via SDO.	Continue	Blinking	Keep
16#3905	Motion network master can not write Slave parameters via SDO.	Continue	Blinking	Keep
16#3906	Torque limit setting error in MC_SetTorqueLimit	Continue	Blinking	Keep
16#3909	The motion network is currently executing other network functions.	Continue	Blinking	Keep
16#390A	The axis is not in ServoOn	Continue	Blinking	Keep
16#390B	The axis can not enter the specified motion state.	Continue	Blinking	Keep
16#3910	Disengage when the axes are not in engaging status	Continue	Blinking	Keep
16#3911	Software limit error	Continue	Blinking	Keep
16#3D00	EtherCAT ENI file does not match current hardware configuration.	Continue	Blinking	Keep
16#3D01	Slave lost in motion network.	Continue	Blinking	Keep
16#6001	Illegal IP address (SM1107)	Continue	Blinking	Keep
16#6002	Illegal netmask address (SM1107)	Continue	Blinking	Keep
16#6003	Illegal gateway mask (SM1107)	Continue	Blinking	Keep
16#6004	The IP address filter is set incorrectly. (SM1108)	Continue	Blinking	Keep
16#6006	The static ARP table is set incorrectly. (SM1108)	Continue	Blinking	Keep
16#600D	The RJ45 port is not connected.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#620D	The length of the data which needs to be sent in a UDP Socket Configuration window is illegal.	Continue	Keep	Keep
16#6212	There is no response from the remote device after the timeout period.	Continue	Keep	Keep
16#6213	The data received exceeds the limit.	Continue	Keep	Keep
16#6214	The remote device refuses the connection.	Continue	Keep	Keep
16#6400	The number of TCP connections reaches the upper limit, or the flag which is related to the sending of the data is not set to ON.	Continue	Keep	Keep
16#6401	The remote device aborts the connection.	Continue	Keep	Keep
16#6402	There is no response from the remote device after the timeout period.	Continue	Keep	Keep
16#6403	The remote IP address used in the applied instruction is illegal.	Continue	Keep	Keep
16#6404	The MODBUS function code not supported is received.	Continue	Keep	Keep
16#6405	The number of data which will be received is not consistent with the actual length of the data.	Continue	Keep	Keep
16#6700	MODBUS TCP data exchange initialization error	Continue	Keep	Keep
16#6701	MODBUS TCP data exchange timeout	Continue	Keep	Keep
16#6702	MODBUS TCP data receiving error	Continue	Keep	Keep
16#7002	CPU do not support this function	Continue	Keep	Keep
16#7203	Invalid access code	Continue	Keep	Keep
16#7401	Function code error	Continue	Keep	Keep
16#7402	The packet exceeds the max. data length.	Continue	Keep	Keep
16#7407	Non-ASCII characters exist in the command.	Continue	Keep	Keep
16#7408	PLC is in RUN mode	Continue	Keep	Keep
16#740B	The Clear or Reset operation is in progress.	Continue	Keep	Keep
16#740C	The backplane number in a communication command is incorrect.	Continue	Keep	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#740D	The slot number in a communication command is incorrect.	Continue	Keep	Keep
16#740F	Communication timeout	Continue	Keep	Keep
16#7412	Data cannot be downloaded to CPU because SW1 is ON.	Continue	Keep	Keep
16#757D	Remaining times for entering PLC password is 0	Continue	Keep	Keep
16#757E	Incorrect PLC password	Continue	Keep	Keep
16#8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Continue	Keep	Keep
16#8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Continue	Keep	Keep
16#8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Continue	Keep	Keep
16#8230	The CPU parameter downloaded is incorrect. The IP address is illegal.	Continue	Blinking	Keep
16#8231	The CPU parameter downloaded is incorrect. The netmask address is illegal.	Continue	Blinking	Keep
16#8232	The CPU parameter downloaded is incorrect. The gateway address is illegal.	Continue	Blinking	Keep
16#8233	The CPU parameter downloaded is incorrect. The IP address filter is set incorrectly.	Continue	Blinking	Keep
16#8235	The CPU parameter downloaded is incorrect. The static ARP table is set incorrectly.	Continue	Blinking	Keep
16#8236	A CPU parameter downloaded is incorrect. The NTP client service is set incorrectly.	Continue	Keep	Keep
16#8240	A CPU parameter downloaded is incorrect. The data exchange by means of Ethernet is set incorrectly	Continue	Keep	Keep
16#8241	The setting of a DNS server is incorrect.	Continue	Keep	Keep
16#8522	A module configuration is being scanned.	Continue	Keep	Keep
16#853B	An I/O module is not configured.(wirte error)	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#853C	An I/O module does not exist. (write error)	Continue	Keep	Keep
16#854B	An I/O module is not configured. (read error)	Continue	Keep	Keep
16#854C	An I/O module does not exist. (read error)	Continue	Keep	Keep
16#8572	The checksum of the module configuration table is incorrect.	Continue	Keep	Keep
16#8576	The checksum of the module parameter setting is incorrect.	Continue	Keep	Keep
16#857A	The checksum of the module parameter mapping table is incorrect.	Continue	Keep	Keep
16#85E1	An I/O interrupt number is incorrect.	Continue	Keep	Keep
16#85E2	An I/O interrupt service routine does not exist.	Continue	Keep	Keep
16#860F	System restoration error	Continue	Keep	Keep
16#8611	No memory card exists, or the memory card format is incorrect.	Continue	Keep	Keep
16#9A33	An error occurs when COM1 communicates with slave 19 by Modbus.	Continue	Keep	Keep
16#9A34	An error occurs when COM1 communicates with slave 20 by Modbus.	Continue	Keep	Keep
16#9A35	An error occurs when COM1 communicates with slave 21 by Modbus.	Continue	Keep	Keep
16#9A47	COM1 receives no response from slave 7 by Modbus.	Continue	Keep	Keep
16#9B01	An error occurs when the Modbus connection of COM2 is initialized.	Continue	Keep	Keep
16#9B21	An error occurs when COM2 communicates with slave 1 by MODBUS.	Continue	Keep	Keep
16#9B22	An error occurs when COM2 communicates with slave 2 by MODBUS.	Continue	Keep	Keep
16#9B23	An error occurs when COM2 communicates with slave 3 by MODBUS.	Continue	Keep	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B24	An error occurs when COM2 communicates with slave 4 by MODBUS.	Continue	Keep	Keep
16#9B25	An error occurs when COM2 communicates with slave 5 by MODBUS.	Continue	Keep	Keep
16#9B26	An error occurs when COM2 communicates with slave 6 by MODBUS.	Continue	Keep	Keep
16#9B27	An error occurs when COM2 communicates with slave 7 by MODBUS.	Continue	Keep	Keep
16#9B28	An error occurs when COM2 communicates with slave 8 by MODBUS.	Continue	Keep	Keep
16#9B29	An error occurs when COM2 communicates with slave 9 by MODBUS.	Continue	Keep	Keep
16#9B2A	An error occurs when COM2 communicates with slave 10 by MODBUS.	Continue	Keep	Keep
16#9B2B	An error occurs when COM2 communicates with slave 11 by MODBUS.	Continue	Keep	Keep
16#9B2C	An error occurs when COM2 communicates with slave 12 by MODBUS.	Continue	Keep	Keep
16#9B2D	An error occurs when COM2 communicates with slave 13 by MODBUS.	Continue	Keep	Keep
16#9B2E	An error occurs when COM2 communicates with slave 14 by MODBUS.	Continue	Keep	Keep
16#9B2F	An error occurs when COM2 communicates with slave 15 by MODBUS.	Continue	Keep	Keep
16#9B30	An error occurs when COM2 communicates with slave 16 by MODBUS.	Continue	Keep	Keep
16#9B31	An error occurs when COM2 communicates with slave 17 by MODBUS.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B32	An error occurs when COM2 communicates with slave 18 by MODBUS.	Continue	Keep	Keep
16#9B33	An error occurs when COM2 communicates with slave 19 by MODBUS.	Continue	Keep	Keep
16#9B34	An error occurs when COM2 communicates with slave 20 by MODBUS.	Continue	Keep	Keep
16#9B35	An error occurs when COM2 communicates with slave 21 by MODBUS.	Continue	Keep	Keep
16#9B36	An error occurs when COM2 communicates with slave 22 by MODBUS.	Continue	Keep	Keep
16#9B37	An error occurs when COM2 communicates with slave 23 by MODBUS.	Continue	Keep	Keep
16#9B38	An error occurs when COM2 communicates with slave 24 by MODBUS.	Continue	Keep	Keep
16#9B39	An error occurs when COM2 communicates with slave 25 by MODBUS.	Continue	Keep	Keep
16#9B3A	An error occurs when COM2 communicates with slave 26 by MODBUS.	Continue	Keep	Keep
16#9B3B	An error occurs when COM2 communicates with slave 27 by MODBUS.	Continue	Keep	Keep
16#9B3C	An error occurs when COM2 communicates with slave 28 by MODBUS.	Continue	Keep	Keep
16#9B3D	An error occurs when COM2 communicates with slave 29 by MODBUS.	Continue	Keep	Keep
16#9B3E	An error occurs when COM2 communicates with slave 30 by MODBUS.	Continue	Keep	Keep
16#9B3F	An error occurs when COM2 communicates with slave 31 by MODBUS.	Continue	Keep	Keep

A5

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B40	An error occurs when COM2 communicates with slave 32 by MODBUS.	Continue	Keep	Keep
16#9B41	COM2 receives no response from slave 1 by MODBUS.	Continue	Keep	Keep
16#9B42	COM2 receives no response from slave 2 by MODBUS.	Continue	Keep	Keep
16#9B43	COM2 receives no response from slave 3 by MODBUS.	Continue	Keep	Keep
16#9B44	COM2 receives no response from slave 4 by MODBUS.	Continue	Keep	Keep
16#9B45	COM2 receives no response from slave 5 by MODBUS.	Continue	Keep	Keep
16#9B46	COM2 receives no response from slave 6 by MODBUS.	Continue	Keep	Keep
16#9B47	COM2 receives no response from slave 7 by MODBUS.	Continue	Keep	Keep
16#9B48	COM2 receives no response from slave 8 by MODBUS.	Continue	Keep	Keep
16#9B49	COM2 receives no response from slave 9 by MODBUS.	Continue	Keep	Keep
16#9B4A	COM2 receives no response from slave 10 by MODBUS.	Continue	Keep	Keep
16#9B4B	COM2 receives no response from slave 11 by MODBUS.	Continue	Keep	Keep
16#9B4C	COM2 receives no response from slave 12 by MODBUS.	Continue	Keep	Keep
16#9B4D	COM2 receives no response from slave 13 by MODBUS.	Continue	Keep	Keep
16#9B4E	COM2 receives no response from slave 14 by MODBUS.	Continue	Keep	Keep
16#9B4F	COM2 receives no response from slave 15 by MODBUS.	Continue	Keep	Keep
16#9B50	COM2 receives no response from slave 16 by MODBUS.	Continue	Keep	Keep
16#9B51	COM2 receives no response from slave 17 by MODBUS.	Continue	Keep	Keep
16#9B52	COM2 receives no response from slave 18 by MODBUS.	Continue	Keep	Keep
16#9B53	COM2 receives no response from slave 19 by MODBUS.	Continue	Keep	Keep
16#9B54	COM2 receives no response from slave 20 by MODBUS.	Continue	Keep	Keep
16#9B55	COM2 receives no response from slave 21 by MODBUS.	Continue	Keep	Keep
16#9B56	COM2 receives no response from slave 22 by MODBUS.	Continue	Keep	Keep
16#9B57	COM2 receives no response from slave 23 by MODBUS.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B58	COM2 receives no response from slave 24 by MODBUS.	Continue	Keep	Keep
16#9B59	COM2 receives no response from slave 25 by MODBUS.	Continue	Keep	Keep
16#9B5A	COM2 receives no response from slave 26 by MODBUS.	Continue	Keep	Keep
16#9B5B	COM2 receives no response from slave 27 by MODBUS.	Continue	Keep	Keep
16#9B5C	COM2 receives no response from slave 28 by MODBUS.	Continue	Keep	Keep
16#9B5D	COM2 receives no response from slave 29 by MODBUS.	Continue	Keep	Keep
16#9B5E	COM2 receives no response from slave 30 by MODBUS.	Continue	Keep	Keep
16#9B5F	COM2 receives no response from slave 31 by MODBUS.	Continue	Keep	Keep
16#9B60	COM2 receives no response from slave 32 by MODBUS.	Continue	Keep	Keep

Analog I/O Modules and Temperature Measurement Modules

A5

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A000	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A001	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A002	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A003	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A004	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A005	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A006	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	Blinking	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A007	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A400	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	ON	
16#A401	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	ON	
16#A402	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	ON	
16#A403	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	ON	
16#A404	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	ON	
16#A405	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	ON	
16#A406	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	ON	
16#A407	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	ON	
16#A600	Hardware failure	ON	
16#A601	The external voltage is abnormal.	ON	
16#A602	Internal error The CJC is abnormal.	ON	
16#A603	Internal error The factory correction is abnormal.	ON	
16#A800	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A801	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	OFF	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A802	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A803	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A804	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A805	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A806	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A807	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	OFF	

*With regard to the errors related to the input signals' exceeding the range of inputs which can be received by the hardware and the conversion values' exceeding the limits, whether the error code generated is within the range between 16#A000 and 16#A00F, within the range between 16#A400 and 16#A40F, or within the range between 16#A800~16#A80F depends on the LED indicator status defined by users.

AH02HC-5A/AH04HC-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A001	The linear accumulation in channel 0 exceeds the range.	Blinking	
16#A002	The scale set for channel 0 exceeds the range.	Blinking	
16#A003	The number of cycles set for channel 0 exceeds the range.	Blinking	
16#A004	The comparison value set for channel 0 exceeds the range.	Blinking	
16#A005	A limit value set for channel 0 is incorrect.	Blinking	
16#A006	The interrupt number set for channel 0 exceeds the range.	Blinking	
16#A011	The linear accumulation in channel 1 exceeds the range.	Blinking	
16#A012	The scale set for channel 1 exceeds the range.	Blinking	
16#A013	The number of cycles set for channel 1 exceeds the range.	Blinking	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A014	The comparison value set for channel 1 exceeds the range.	Blinking	
16#A015	A limit value set for channel 1 is incorrect.	Blinking	
16#A016	The interrupt number set for channel 1 exceeds the range.	Blinking	
16#A021	The linear accumulation in channel 2 exceeds the range.	Blinking	
16#A022	The scale set for channel 2 exceeds the range.	Blinking	
16#A023	The number of cycles set for channel 2 exceeds the range.	Blinking	
16#A024	The comparison value set for channel 2 exceeds the range.	Blinking	
16#A025	A limit value set for channel 2 is incorrect.	Blinking	
16#A026	The interrupt number set for channel 2 exceeds the range.	Blinking	
16#A031	The linear accumulation in channel 3 exceeds the range.	Blinking	
16#A032	The scale set for channel 3 exceeds the range.	Blinking	
16#A033	The number of cycles set for channel 3 exceeds the range.	Blinking	
16#A034	The comparison value set for channel 3 exceeds the range.	Blinking	
16#A035	A limit value set for channel 3 is incorrect.	Blinking	
16#A036	The interrupt number set for channel 3 exceeds the range.	Blinking	

AH05PM-5A/AH10PM-5A/AH15PM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	Error
16#A002	The subroutine has no data.	Blinking	
16#A003	CJ, CJN, and JMP have no matching pointers.	Blinking	
16#A004	There is a subroutine pointer in the main program.	Blinking	
16#A005	Lack of the subroutine	Blinking	
16#A006	The pointer is used repeatedly in the same program.	Blinking	
16#A007	The subroutine pointer is used repeatedly.	Blinking	
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	Blinking	
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	Blinking	
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	Blinking	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	Error
16#A00B	Target position (I) of the single speed is incorrect.	Blinking	
16#A00C	Target position (II) of the single-axis motion is incorrect.	Blinking	
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Blinking	
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	Blinking	
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Blinking	
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Blinking	
16#A011	The setting of the JOG speed is incorrect.	Blinking	
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	Blinking	
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	Blinking	
16#A014	The limit switch is reached.	Blinking	
16#A015	The device which is used exceeds the device range.	Blinking	
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Blinking	
16#A018	The conversion into the floating-point number is incorrect.	Blinking	
16#A019	The conversion into the binary-coded decimal number is incorrect.	Blinking	
16#A01A	Incorrect division operation (The divisor is 0.)	Blinking	
16#A01B	General program error	Blinking	
16#A01C	LD/LDI has been used more than nine times.	Blinking	
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Blinking	
16#A01E	SRET is used between RPT and RPE.	Blinking	
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Blinking	
16#A020	The wrong instruction is used, or the device used exceeds the range.	Blinking	

AH20MC-5A

Error code	Description	LED indicator status	
		CPU	Module

		BUS FAULT	ERROR
16#A002	The subroutine has no data.		Blinking
16#A003	CJ, CJN, and JMP have no matching pointers.		Blinking
16#A004	There is a subroutine pointer in the main program.		Blinking
16#A005	Lack of the subroutine		Blinking
16#A006	The pointer is used repeatedly in the same program.		Blinking
16#A007	The subroutine pointer is used repeatedly.		Blinking
16#A008	The pointer used in JMP is used repeatedly in different subroutines.		Blinking
16#A009	The pointer used in JMP is the same as the pointer used in CALL.		Blinking
16#A00B	Target position (I) of the single speed is incorrect.		Blinking
16#A00C	Target position (II) of the single-axis motion is incorrect.		Blinking
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.		Blinking
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.		Blinking
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.		Blinking
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.		Blinking
16#A011	The setting of the JOG speed is incorrect.		Blinking
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.		Blinking
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.		Blinking
16#A014	The limit switch is reached.		Blinking
16#A015	The device which is used exceeds the device range.		Blinking
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.		Blinking
16#A018	The conversion into the floating-point number is incorrect.		Blinking
16#A019	The conversion into the binary-coded decimal number is incorrect.		Blinking
16#A01A	Incorrect division operation (The divisor is 0.)		Blinking
16#A01B	General program error		Blinking
16#A01C	LD/LDI has been used more than nine times.		Blinking
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.		Blinking

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A01E	SRET is used between RPT and RPE.	Blinking	
16#A01F	Incorrect division operation (The divisor is 0.)	Blinking	
16#A020	The wrong instruction is used, or the device used exceeds the range.	Blinking	

AH10COPM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A0B0	AH10COPM-5A does not send a heartbeat message after a set period of time.	Blinking	The red light flashes twice.
16#A0B1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Blinking	OFF
16#A0B2	The master station selected does not send a node guarding message after a set period of time.	Blinking	The red light flashes twice.
16#A0E0	AH10COPM-5A receives an emergency message from a slave station.	Blinking	OFF
16#A0E1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Blinking	OFF
16#A0E2	AH10COPM-5A does not receive a PDO from a slave station.	Blinking	OFF
16#A0E3	An automatic SDO is not downloaded successfully.	Blinking	OFF
16#A0E4	A PDO parameter is not set successfully.	Blinking	OFF
16#A0E5	A key parameter is set incorrectly.	Blinking	OFF
16#A0E6	The actual network configuration is not the same as the network configuration set.	Blinking	OFF
16#A0E7	The control of the errors in a slave station is not sent after a set period of time.	Blinking	The red light flashes twice.
16#A0E8	The master station address is the same as a slave station address.	Blinking	OFF
16#A0F1	No slave station is added to the node list in CANopen builder.	Blinking	OFF
16#A0F3	An error occurs in AH10COPM-5A.	Blinking	OFF

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A0F4	The bus used is off.	Blinking	The red light is ON.
16#A0F5	The node address of AH10COPM-5A is set incorrectly.	Blinking	OFF
16#A0F6	Internal error: An error occurs in the manufacturing process in the factory.	Blinking	OFF
16#A0F7	Internal error: GPIO error	Blinking	OFF
16#A0F8	Hardware error	Blinking	OFF
16#A0F9	Low voltage	Blinking	OFF
16#A0FA	An error occurs in the firmware of AH10COPM-5A.	Blinking	OFF
16#A0FB	The transmission registers in AH10COPM-5A are full.	Blinking	OFF
16#A0FC	The reception registers in AH10COPM-5A are full.	Blinking	OFF

AH10SCM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A002	The setting of the UD Link is incorrect, or the communication fails.	Blinking	
16#A401	Hardware error	ON	
16#A804	The communication through the communication port is incorrect.	OFF	
16#A808	MODBUS communication error	OFF	

A5**A.5.2. Error Codes and Troubleshooting****AHxxEMC-5A**

You can get the corrective actions from the tables below according to the error codes.

ERROR Indicator ON

Error Code	Description	Corrective action
16#000B	The program in the PLC is damaged.	Download the program again.
16#000D	The CPU parameters are damaged.	Reset the CPU parameter, and download it.
16#0010	The access to the memory in the CPU is denied.	Download the program or parameters again. If the problem still occurs, please contact the manufacturer.
16#0011	The PLC ID is incorrect. (SM9)	Please check the PLC ID.

Error Code	Description	Corrective action
16#0012	The PLC password is incorrect. (SM9)	Please check the PLC password.
16#0014	The procedure of restoring the system can not be executed. (SM9)	The contents of the system backup file are incorrect, or the file does not exist in the path specified. If the file exists and the procedure of restoring the system can not be executed, please back up the system again. If the error still occurs, please contact the manufacturer. (You can refer to AH Motion – Operation Manual for more details about using memory cards)
16#0015	The module table is incorrect. (SM10)	The module table stored in the CPU module is incorrect. Compare the module table in HWCONFIG with the actual module configuration, and download the module table again.
16#0016	The module setting is incorrect. (SM10)	The module setting stored in the CPU module is incorrect. Check whether the version of the module inserted in the slot is the same as the version of the module in HWCONFIG. After the version of the module is updated, users can download the module setting again.
16#0017	The data register exceeds the device range. (SM10)	The data register stored in the CPU module exceeds the device range. Check whether the module parameter in HWCONFIG is correct, and download the module parameter again.
16#001B	Timed interrupt 0 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001C	Timed interrupt 1 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001D	Timed interrupt 2 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001E	Timed interrupt 3 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001F	The watchdog timer is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.

Error Code	Description	Corrective action
16#0020	The setting of the fixed scan time is incorrect.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#0021	The setting of the fixed scan time is incorrect.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#0022	The CPU parameter downloaded to the PLC is incorrect.	Download the CPU parameter again.
16#0023	CPU parameters setting error. The state of Y devices when the CPU is set from STOP to RUN is incorrect	Adjust the CPU parameters setting in HWCONFIG and download it to PLC again.
16#0026	The Communication Ratio box in the Communication Loading of Scan Time (%) section in the PLC Parameter Setting window is set incorrectly.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0027	The latching auxiliary relay range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0028	The latching data register range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0029	The latching timer range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#002A	The latching counter range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#002B	The latching 32-bit counter range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0050	The memories in the latched special auxiliary relays are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0051	The latched special data registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.

Error Code	Description	Corrective action
16#0052	The memories in the latched auxiliary relays are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0053	The latched timers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0054	The latched counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0055	The latched 32-bit counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0056	The memories in the latched timers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0057	The memories in the latched counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0058	The memories in the latched 32-bit counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0059	The latched data registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#005A	The latched working registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.

ERROR Indicator Blinking

Error Code	Description	Corrective action
16#000A	Scan timeout (SM8: The watchdog timer error)	<ol style="list-style-type: none"> 1. Check the setting of the watchdog timer in HWCONFIG. 2. Check whether the program causes the long scan time

Error Code	Description	Corrective action
16#000C	The program downloaded to the PLC is incorrect.	After users compile the program again, they can download the program again.
16#000E	The program or the parameter is being downloaded, and therefore the PLC can not run.	After the program or the parameter is downloaded to the PLC, users can try to run the PLC.
16#0018	The serial port is abnormal. (SM9)	Retry the connection. If the error still occurs, please contact the factory.
16#0019	The USB is abnormal. (SM9)	Retry the connection. If the error still occurs, please contact the factory.
16#001A	The contents of the system backup file (.dup file) are incorrect.	Create the system backup file again.
16#0033	The communication setting of COM1 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0034	The setting of the station address of COM1 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0035	The setting of the communication type of COM1 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0038	The communication setting of COM2 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0039	The setting of the station address of COM2 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#003A	The setting of the communication type of COM2 is incorrect. (SM9)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.

Error Code	Description	Corrective action
16#0066	An error occurs when the system is backed up.	1. Check whether the memory card is normal, and whether the capacity of the memory card is large enough. 2. Retry the backup procedure. If the error still occurs, please contact the factory.
16#0067	The size of the PLC parameters restored exceeds the size of the PLC parameters of the CPU module.	The error code is appeared to indicate alarm only.
16#2000	There is no END in the program in the PLC. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#2001	The program is incorrect. There is a syntax error.	Check the program, compile the program again, and download the program again.
16#2002	GOEND is used incorrectly. (SM5)	Check the program, compile the program again, and download the program again.
16#2003	The devices used in the program exceed the range. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2004	The part of the program specified by the label used in CJ/JMP is incorrect, or the label is used repeatedly. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2005	The N value used in MC is not the same as the corresponding N value used in MCR, or the number of N values used in MC is not the same as the number of N values used in MCR. (SM5)	Check the program, compile the program again, and download the program again.
16#2006	The N values used in MC do not start from 0, or the N values used in MC are not continuous. (SM5)	Check the program, compile the program again, and download the program again.
16#2007	The operands used in ZRST are not used properly. (SM5)	Check the program, compile the program again, and download the program again.

Error Code	Description	Corrective action
16#200A	Invalid instruction (SM5)	Check the program, compile the program again, and download the program again.
16#200B	The operand n or the other constant operands exceed the range. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200C	The operands overlap. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200D	An error occurs when the binary number is converted into the binary-coded decimal number. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200E	The string does not end with 0x00. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200F	The instruction does not support the modification by an index register. (SM5)	Check the program, compile the program again, and download the program again.
16#2010	<ol style="list-style-type: none"> 1. The instruction does not support the device. 2. Encoding error 3. The instruction is a 16-bit instruction, but the constant operand is a 32-bit code. (SM5) 	Check the program, compile the program again, and download the program again.
16#2011	The number of operands is incorrect. (SM5)	Check the program, compile the program again, and download the program again.
16#2012	Incorrect division operation (SM0/SM5).	Check the program, compile the program again, and download the program again.
16#2013	The value exceeds the range of values which can be represented by the floating-point numbers. (SM0/SM5)	Check the program, compile the program again, and download the program again.

Error Code	Description	Corrective action
16#2014	The task designated by TKON/TKOFF is incorrect, or exceeds the range. (SM5)	Check the program, compile the program again, and download the program again.
16#2015	There are more than 32 levels of nested program structures supported by CALL. (SM0)	Check the program, compile the program again, and download the program again.
16#2016	There are more than 32 levels of nested program structures supported by FOR/NEXT. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2017	The number of times FOR is used is different from the number of times NEXT is used. (SM5)	Check the program, compile the program again, and download the program again.
16#2018	There is a label after FEND, but there is no SRET. There is SRET, but there is no label. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#2019	The interrupt task is not after FEND. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#201A	IRET/SRET is not after FEND. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#201B	There is an interrupt task, but there is no IRET. There is IRET, but there is not interrupt task. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#201C	End is not at the end of the program. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.

A5

Error Code	Description	Corrective action
16#201D	There is CALL, but there is no MAR. (SM5)	<ol style="list-style-type: none"> 1. Compile the program again, and download the program again. 2. Reinstall ISPSOft, compile the program again, and download the program again.
16#201E	The function code used in MODRW is incorrect. (SM102/SM103)	Check the usage of the instruction and the setting of the operands. Please refer to the explanation of the instruction MODRW in AH500 Programming Manual for more information.
16#201F	The length of the data set in MODRW is incorrect. (SM102/SM103)	Check the usage of the instruction and the setting of the operands. Please refer to the explanation of the instruction MODRW in AH500 Programming Manual for more information.
16#2020	The communication command received by using MODRW is incorrect. (SM102/SM103)	Check whether the slave supports the function code and the specified operation.
16#2021	The checksum of the command received is incorrect. (SM102/SM103)	<ol style="list-style-type: none"> 1. Check whether there is noise, and retry the sending of the command. 2. Check whether the slave operates normally.
16#2022	The format of the command used in MODRW does not conform to the ASCII format. (SM102/SM103)	Make sure that the format of the command conforms to the ASCII format.
16#2023	There is a communication timeout when MODRW is executed. (SM120/SM103)	Check whether the slave operates normally, and whether the connection is normal.
16#2024	The setting value of the communication timeout is invalid. (SM120/SM103)	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#2025	There is a communication timeout when RS is executed. (SM120/SM103)	Check whether the slave operates normally, and whether the connection is normal.
16#2026	The interrupt number used in RS is incorrect. (SM102/104)	Check whether the the interrupt service routine used in RS is downloaded.

Error Code	Description	Corrective action
16#2027	The execution of FWD is abnormal.	Please refer to AH500 Programming Manual, and check the instruction FWD.
16#2028	The execution of REV is abnormal.	Please refer to AH500 Programming Manual, and check the instruction REV.
16#2029	The execution of STOP is abnormal.	Please refer to AH500 Programming Manual, and check the instruction STOP.
16#202A	The execution of RSDT is abnormal.	Please refer to AH500 Programming Manual, and check the instruction RSDT.
16#202B	The execution of RSTEF is abnormal.	Please refer to AH500 Programming Manual, and check the instruction RSTEF.
16#202C 16#204B	I/O interrupt service routine 0 does not exist. I/O interrupt service routine 31 does not exist.	Download I/O interrupt service routine 0 (I/O interrupt 0) Download I/O interrupt service routine 31 (I/O interrupt 31)
16#2054 16#2127	External interrupt service routine 40 does not exist. External interrupt service routine 251 does not exist.	Download external interrupt service routine 40 (external interrupt 40) Download external interrupt service routine 251 (external interrupt 251)
16#2128	An action in a sequential function chart is incorrectly assigned qualifiers related to time.(SM0/SM1)	Check whether qualifiers related to time are duplicated when setting SFC action.
16#2129	The modifier R is assigned to an action in a sequential function chart incorrectly. (SM0/SM1)	Check whether there are conflict settings between properties when setting SFC action.
16#3100	Input parameters exceed the available setting range.	Check whether the input parameters exceed the available setting range.
16#3102	An error occurs in a sub-function block inside the function block.	Re-execute the function block instruction.

Error Code	Description	Corrective action
16#3103	The distance between the detecting sensors used for identifying exceptional bags is a negative value.	Check whether the positions of the detecting sensors are correct.
16#3104	Phasing is executed again before the previous phasing is completed.	Cause: the instruction is executed again when <i>Done</i> is still False. Action: re-execute the instruction again.
16#3105	Superimposing is executed again before the previous superimposing is completed.	Cause: the instruction is executed again when <i>Done</i> is still False. Action: re-execute the instruction again.
16#3106	Chain position compensation is triggered before the previous compensation is completed.	Cause: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. Action: adjust all packaging related parameters according to the application requirements.
16#3107	Film axis position compensation is triggered before the previous compensation is completed.	Cause: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. Action: adjust all packaging related parameters according to the application requirements.
16#3108	Knife position compensation is triggered before the previous compensation is completed.	Cause: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. Action: adjust all packaging related parameters according to the application requirements..
16#3405	A negative value is given to <i>Velocity</i> .	Cause: the value given to <i>Velocity</i> is negative or 0. Action: set the velocity to a positive value and re-execute the instruction.
16#3407	The speed of returning to zero (V_{RT}) is incorrect.	Cause: V_{RT} is negative, 0, or higher than 2,000rpm. Action: set V_{RT} to a value of 1~2,000 and re-execute the instruction.
16#3408	The deceleration of returning to zero (V_{CR}) is incorrect.	Cause: V_{CR} is negative, 0, or higher than 500rpm. Action: set V_{CR} to a value of 1~500 and re-execute the instruction.

Error Code	Description	Corrective action
16#340A	Homing mode setting error.	Cause: homing mode is not set to a value between 1 and 35. Action: set homing mode to a value between 1 and 35 and re-execute the instruction.
16#340B	Target distance is 0.	Cause: target distance of this instruction is not set to 0. Action: set target distance to a positive value and re-execute the instruction.
16#3410	User unit setting error; or the output pulse type setting error.	Cause: user unit setting of this instruction is not set to 0~2. Action: set the user unit to 0~2 and re-execute the instruction.
16#3411	Velocity override factor setting error.	Cause: velocity override factor of this instruction is not set to 0~500. Action: set the velocity override factor to 0~500 and re-execute the instruction.
16#3412	Continuous interpolation velocity setting error.	Cause: continuous interpolation velocity of this instruction is set to a negative value or 0. Action: set the continuous interpolation velocity to a positive value and re-execute the instruction.
16#3419	Master axis position is negative value.	Cause: master axis position is set to a negative value or 0. Action: set the master axis position to a positive value and re-execute the instruction.
16#341B	Maximum speed setting error.	Cause: maximum speed is not set to 1~1,000,000. Action: set the maximum speed to 1~1,000,000 and re-execute the instruction.
16#3429	G-code compiling error.	Cause: non-supported G-codes exist in the G-code file; or errors occur in the G-code format. Action: remove non-supported G-code or fix G-code format errors in the G-code file, and download the G-code file again.
16#342A	G-code program source error.	Cause: the designated file name (<i>GcodeID</i>) does not exist in the SD card or the AH Motion CPU. Action: enter the correct file name of the G-code program to <i>GcodeID</i> and re-execute the instruction.

Error Code	Description	Corrective action
16#342B	G-code ID setting error.	Cause: <i>GcodeID</i> is not set to a value between 1 and 136. Action: set <i>GcodeID</i> to 1~136 and re-execute the instruction.
16#3431	Motion axis number is repeated in the same group in <i>DFB_GroupEnable</i> .	Cause: The given value to the inputs <i>AxisNumorder1</i> ~ <i>AxisNumorder6</i> is repeated. Action: adjust the axis numbers to avoid repeated number in the same group and re-execute the instruction.
16#3432	The specified group number does not exist.	Cause: The designated group in <i>GroupNum</i> is not enabled. Action: specify an enabled group number to the instruction.
16#3433	The number of axes is insufficient for the specified group axes motion.	Cause: The designated group in <i>GroupNum</i> does not have enough axes for performing the required group motion. Action: designate a group with sufficient axes for the required group motion, e.g. at least 2 axes are needed for the group to perform linear interpolation, and 3 axes to perform arc interpolation.
16#3434	<i>DFB_GroupDisable</i> is executed when group motion is in progress.	Execute <i>DFB_GroupReset</i> to clear the error status of the group.
16#3435	Motion axis number is repeated in different groups when <i>DFB_GroupEnable</i> is enabled.	Cause: The specified axis number in <i>AxisNumorder1</i> ~ <i>AxisNumorder6</i> is repeated in different groups. Action: adjust the axis numbers to avoid repeated number in different groups and re-execute the instruction.
16#3436	The axis number of the first order should be a positive number other than 0.	Cause: <i>AxisNumorder1</i> is not given a positive number. Action: set <i>AxisNumorder1</i> to a positive number and re-execute the instruction.
16#3437	The group number exceeds the setting range.	Cause: <i>GroupNum</i> is not given a value between 1 and 32. Action: set <i>GroupNum</i> 1 to 1~32 and re-execute the instruction.
16#3438	The designated group is in "ErrorStop"	Execute <i>DFB_GroupReset</i> to clear the error status of the group.

Error Code	Description	Corrective action
16#3461	The required communication parameters for PDO settings are not specified	<p>Cause: the required communication parameters for PDO settings are not specified when the function block is in execution.</p> <p>Action: re-execute ECAT Builder and specify the required parameters for the function block.</p>
16#3505	An error occurs when writing cam data.	<p>Cause: the cam data read is different from the cam data written, which is detected by the checking after DFB_CamWrite is executed.</p> <p>Action: re-execute the DFB_CamWrite instruction.</p>
16#3506	The axis is in "Coordinated"	<p>Cause: the axis is in "Coordinated" when MC_stop is executed.</p> <p>Action: confirm that <i>Execute</i> =False and <i>Done</i>=True in MC_Stop. Use DFB_GroupReset to reset the axis to "Standby" and use DFB_GroupDisable to disable the group motion.</p>
16#3507	The axis is in "ErrorStop"	<p>Cause: The axis is in "ErrorStop" when the instruction is executed.</p> <p>Action: use MC_Reset to reset the axis error status.</p>
16#3508	The axis is not in "Standstill"	<p>Cause: the axis is not in "Standstill" when the instruction is executed.</p> <p>Action: execute MC_Reset and confirm is the axis is in "Standstill"</p>
16#3600	The axis is currently in group motion	<p>Cause: the axis used by this instruction is currently in group motion.</p> <p>Action: execute DFB_GroupDisable to detach the axis from group motion.</p>
16#3601	The limit of the number of buffering instructions is reached	<p>Cause: the number of buffering instructions (with buffer mode enabled) reached 20.</p> <p>Action: 1. The error status will lead the axis to "ErrorStop". In this case, execute MC_Reset to set the axis back to "Standstill". 2. Make sure the total number of buffering instructions is less than 20 before executing current instruction.</p>

A5

Error Code	Description	Corrective action
16#3602	A multiple instructions which are not allowed to be executed at the same time are executed.	<p>Cause: the instruction is executed when another instruction is in execution at the same time. (Both do not support simultaneously execution)</p> <p>Action: use MC_Reset to clear the axis error, and set the axis state to “StandStill.”</p>
16#3603	Buffermode parameter setting error	<p>Cause: the set value in <i>Buffermode</i> is not valid.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3604	Errors occur on the motion direction of the function block	<p>Cause: the moving direction of the axis is not correct.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3605	P1 exceeds the available range	<p>Cause: the target position is not specified with an available value.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3606	P2 exceeds the available range	<p>Cause: the target position is not specified with an available value.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3607	V1 exceeds the available range	<p>Cause: the target velocity is not specified with an available value.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3608	V2 exceeds the available range	<p>Cause: the target velocity is not specified with an available value.</p> <p>Action: use MC_Reset to clear the axis error, and specify the input parameters again.</p>
16#3612	LSP is On in positive travel	<p>Cause: positive limit is reached.</p> <p>Action: use MC_Reset to clear the axis error, and move the position potively or negatively to the proper position.</p>
16#3613	LSN is On in negative travel	<p>Cause: negative limit is reached.</p> <p>Action: use MC_Reset to clear the axis error, and move the position potively or negatively to the proper position.</p>

Error Code	Description	Corrective action
16#3614	The servo limit is exceeded.	<p>Cause: the servo drive limit is reached.</p> <p>Action: use MC_Reset to clear the axis error, and move the position positively or negatively to the proper position.</p>
16#3615	SuperImposing is on-going	<p>Cause: MPG is executed while superimposing is on-going.</p> <p>Action: use MC_Reset to clear the axis error, and wait while the superimposing is finished.</p>
16#3800	Motion network disconnected during the execution of the instruction.	Check whether the network cable is detached or the network is disconnected.
16#3801	Axis error occurs on the motion network	<p>Cause: the motion axis reports an alarm or an error during the motion.</p> <p>Action: read the axis states and errors by using related function blocks, and reset the axis error by using MC_Reset.</p>
16#3900	Failed to re-connect to the motion network.	<p>Cause: After the motion network is reset, the CPU cannot re-connect to the motion network.</p> <p>Action: 1. Check whether the network cable is detached or the network is disconnected. 2. Check whether the connected servo drive is powered on.</p>
16#3902	The servo axis cannot be powered on. (ServoOn failed)	<p>Cause: the servo axis cannot be powered on when <i>Enable</i> of MC_Power changes to True.</p> <p>Action: Check whether the motor driven by the servo drive is powered on.</p>
16#3903	The servo axis cannot be powered off. (ServoOff failed)	<p>Cause: the servo axis cannot be powered off when <i>Enable</i> of MC_Power changes to False.</p> <p>Action: Check the parameters of the servo drive and disable the "Force On" settings, if it is specified.</p>
16#3904	Motion network master can not read Slave parameters via SDO.	Check whether the parameter reading settings of Group and Parameter matches the available range of the servo drive.
16#3905	Motion network master can not write Slave parameters via SDO.	<p>1. Check whether the parameter writing settings of Group and Parameter matches the available range of the servo drive.</p> <p>2. Check whether the specified values to be written are within the available setting range for the parameters.</p>

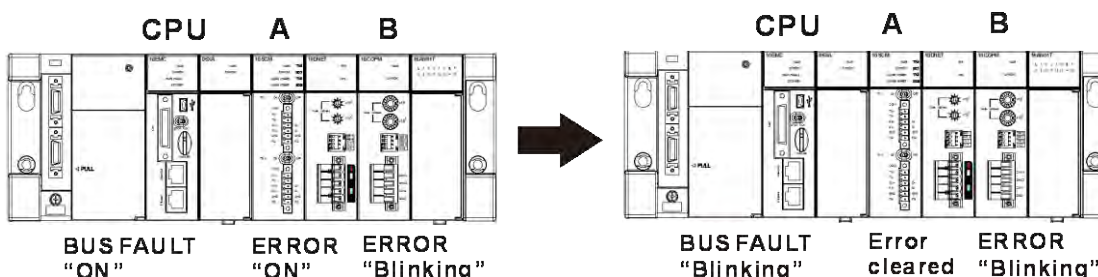
A5

Error Code	Description	Corrective action
16#3906	Torque limit setting error in MC_SetTorqueLimit	Cause: the specified value for <i>PositiveValue</i> or <i>NegativeValue</i> is invalid. Action: Check whether the specified value for <i>PositiveValue</i> or <i>NegativeValue</i> is within available setting range of the servo drive.
16#3909	The motion network is currently executing other network functions.	Check the read/write status of SDO to see if the motion network is executing other network functions.
16#3911	Software limit error	Cause: the axis reached the software limit. Action: use MC_Reset to clear the error and use MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity or DFB_MPG to move the axis back to the proper range.
16#3D00	EtherCAT ENI file does not match current hardware configuration.	Cause: EtherCAT ENI file in the system does not match current EtherCAT configuration. Action: download again the ENI file that matches current EtherCAT configuration.
16#3D01	Slave lost.in motoin network	Cause: slave lost during the motion network communication. Action: reconnect to the motion netowrk
16#6001	Illegal IP address (SM1107)	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6002	Illegal netmask address (SM1107)	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6003	Illegal gateway mask (SM1107)	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6004	The IP address filter is set incorrectly. (SM1108)	Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6006	The static ARP table is set incorrectly. (SM1108)	Set the Ethernet parameter for the CPU module in HWCONFIG again.

BUS FAULT Indicator ON

The BUS FAULT indicator on the CPU would be ON to indicate an error on CPU, or to indicate an error on I/O module together with the ERROR indicator on an I/O module. If an error occurs in an I/O module, the status of the BUS FAULT

indicator on the CPU will be the same as that of the ERROR indicator on the I/O module. If multiple errors occur in the I/O modules, the BUS FAULT indicator on the CPU will keep ON (not blinking). For example, if the ERROR indicator on module A is ON and the ERROR indicator on module B blinks, the BUS FAULT indicator will keep ON. When the error in I/O module A is cleared, module B will blink and the BUS FAULT indicator will blink as well. Refer to the section **A.5.1** for more information about the indicator behaviors of each module.



You can get the corrective actions for the CPU errors indicated by the BUS FAULT indicator from the table below. If the error code you obtained is not listed in the table below, you can check if an error occurs on the I/O modules. Refer to the following content of this section for more information about the troubleshooting for I/O modules.

Error Code	Description	Corrective action
16#0013	The I/O module can not run/stop. (SM10)	Check whether the setting of the parameter for the module is correct. If the setting is correct, please check whether the module breaks down. If the error still occurs, please contact the manufacturer.
16#0014	The procedure of restoring the system can not be executed. (SM9)	The contents of the system backup file are incorrect, or the file does not exist in the path specified. If the file exists and the procedure of restoring the system can not be executed, please backing up the system again. If the error still occurs, please contact the manufacturer.
16#1401	An error occurs when the data in the I/O module is accessed. (SM9)	Please contact the factory.
16#1402	The actual arrangement of the I/O modules is not consistent with the module table. (SM9)	Check whether the module table in HWCONFIG is consistent with the actual arrangement of the I/O modules.
16#1403	An error occurs when the data is read from the module. (SM9)	Check whether the module operates normally. If the error still occurs, please contact the factory.
16#1405	The setting parameter of the module is not found. (SM9)	Set the parameter in HWCONFIG again, and download it.
16#140B	The number of network modules exceeds the limit. (SM9)	Please decrease the number of network modules to the number supported by the system.

A5

Error Code	Description	Corrective action
16#140C	The checksum of the high-speed data exchange is incorrect	Please check the version of the firmware installed on the module, and contact the factory.
16#140D	The ID of the actual power supply module is not the same as the ID of the power supply module set in HWCONFIG. (SM9)	Check whether the power supply configuration in HWCONFIG is consistent with the actual arrangement of the power supply module.
16#140E	The amount of data exchanged at a high speed exceeds the maximum amount supported.	Check the firmware version and contact the supplier.
16#140F	High-speed data exchange error	Check the firmware version and contact the supplier.

BUS FAULT Indicator Blinking

If the BUS FAULT indicator blinks, check the operating state of the module. Refer to sections **A.5.1** for more information about the indicators behaviors of each module, and refer to the following content of this section for more information about the troubleshooting for I/O modules.

Others

Error Code	Description	Corrective action
16#000F	The original program in the PLC is damaged.	After users compile the program again, they can download the program again.
16#005E	The memory card is initialized incorrectly. (SM453)	Check whether the memory card breaks down.
16#005F	The file to be read does not exist in the memory card; or the file directory to write in a file does not exist. (SM453)	Check whether the file name and file directory is correct.
16#0061	The storage capacity of the memory card is not enough. (SM453)	Check whether the storage capacity of the memory card is enough, or whether the memory card breaks down.
16#0062	The memory card is write-protected. (SM453)	Check whether the memory card is write-protected.
16#0063	An error occurs when the data is written into the memory card. (SM453)	Check whether the file path is correct, or whether the memory card breaks down.

Error Code	Description	Corrective action
16#0064	The file in the memory card can not be read. (SM453)	Check whether the file path is correct, or whether the file is damaged.
16#0065	The file in the memory card is a read-only file. (SM453)	Users need to set the file so that the file is not a read-only file.
16#1801	There is no interrupt service routine in the CPU module.	Check whether a corresponding interrupt service routine is created in the PLC program (24V LV Detection)
16#3047	No SD card is detected; or the specified file does not exist in the SD card.	Cause: no SD is detected when powering on the AH Motion CPU, therefore an error occurs in the SD card related instructions. Action: power off the CPU, re-install the memory card, and power on again, and re-execute the instruction.
16#3400	Motion axis number is incorrect.	Cause: motion axis number is not between 1 and 32. Action: set the axis number to 1~32 and re-execute the instruction.
16#3401	SDO Data Type setting error in DFB_ECATServoWrite.	Cause: <i>Data Type</i> of DFB_ECATServoWrite is not 0 or 1. Action: set <i>Data Type</i> to 0 or 1 and re-execute the instruction.
16#3404	The number of the counting channel exceeds the available setting range.	Cause: <i>Channel</i> of the instruction exceeds the available setting range. Action: set <i>Channel</i> of the instruction to the available value and re-execute the instruction.
16#340E	Incorrect value given to the source of the comparison in DFB_Compare.	Cause: <i>Source</i> of DFB_Compare exceeds the value between 0 and 7. Action: set <i>Source</i> of DFB_Compare to 0~7 and re-execute the instruction.
16#3414	Pulse type setting error in DFB_HCnt.	Cause: <i>InputType</i> of DFB_HCnt exceeds the value between 0 and 3. Action: set <i>InputType</i> of DFB_HCnt to 0~3 and re-execute the instruction.
16#3415	Comparison condition setting error in DFB_Compare.	Cause: <i>CmpMode</i> of DFB_Compare exceeds the value between 0 and 2. Action: set <i>CmpMode</i> of DFB_Compare to 0~2 and re-execute the instruction.

Error Code	Description	Corrective action
16#3421	Device type setting error in DFB_SDDDevRead.	<p>Cause: <i>Device</i> of DFB_SDDDevRead exceeds the value between 0 and 7.</p> <p>Action: set <i>Device</i> of DFB_SDDDevRead to 0~7 and re-execute the instruction.</p>
16#3422	Output device setting error in DFB_Compare.	<p>Cause: <i>OutputDevice</i> of DFB_Compare exceeds the value between 0 and 7.</p> <p>Action: set <i>OutputDevice</i> of DFB_Compare to 0~7 and re-execute the instruction.</p>
16#6212	There is no response from the remote device after the timeout period.	Make sure that the remote device is connected.
16#6213	The data received exceeds the limit.	<ol style="list-style-type: none"> 1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6214	The remote device refuses the connection.	Make sure that the remote device operates normally.
16#6400	The number of TCP connections reaches the upper limit, or the flag which is related to the sending of the data is not set to ON.	<ol style="list-style-type: none"> 1. Check whether the flag which is related to the sending of the data in the program is modified. 2. Retry the setting of the flag and the sending of the packet.
16#6401	The remote device aborts the connection.	Check whether the remote device support the MODBUS port (502).
16#6402	There is no response from the remote device after the timeout period.	Check whether the remote device operate normally.
16#6403	The remote IP address used in the applied instruction is illegal.	Check whether the program is correct.
16#6404	The MODBUS function code not supported is received.	Check the command transmitted from the remote device.
16#6405	The number of data which will be received is not consistent with the actual length of the data.	Check the command transmitted from the remote device.

Error Code	Description	Corrective action
16#6501	The remote device involved in the data exchange does not respond after the timeout period. (SM828~SM955)	Check the device whose connection number corresponds to the error flag, and check whether it is connected normally.
16#6502	The remote device involved in the data exchange does not respond correctly. (SM828~SM955)	Check the device whose connection number corresponds to the error flag, and check whether it is connected normally.
16#6700	MODBUS TCP data exchange initialization error	Check the setting value and download the data again.
16#6701	MODBUS TCP data exchange timeout	Confirm if the device to be connected supports MODBUS communication protocol.
16#6702	MODBUS TCP data receiving error	Confirm if the device to be connected supports MODBUS communication protocol.
16#7002	CPU do not support this function	Check the CPU firmware version.
16#7203	Invalid access code	Check the content of the packet sent by the device to be connected.
16#7401	Function code error	Check the content of the packet sent by the device to be connected.
16#7402	The packet exceeds the max. data length.	Check the content of the packet sent by the device to be connected.
16#7407	Non-ASCII characters exist in the command.	Check the content of the packet sent by the device to be connected.
16#7408	PLC is in RUN mode	Data download for program or CPU parameters is not allowed when PLC is in RUN mode.
16#740B	The Clear or Reset operation is in progress.	The RST/CLR operation is in progress. Please try again later.
16#740E	Error occurs when clearing the memory	Please try again. If the error occurs again, contact the supplier.
16#740F	Communication timeout	Check if the device to be connected is in normal operation.

Error Code	Description	Corrective action
16#7410	The received Function Code doesn't match the current Function Code.	Check the packet content sent by the remote device.
16#7412	Data cannot be downloaded to CPU because SW1 is ON.	Confirm that SW1 is OFF.
16#757D	Remaining times for entering PLC password is 0	The password retry limit is reached. Please power on the PLC again.
16#757E	Incorrect PLC password	Check if the password is correct.
16#8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Download the program again.
16#8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Download the program again.
16#8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Download the program again.
16#8230	The CPU parameter downloaded is incorrect. The IP address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8231	The CPU parameter downloaded is incorrect. The netmask address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8232	The CPU parameter downloaded is incorrect. The gateway address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8233	The CPU parameter downloaded is incorrect. The IP address filter is set incorrectly.	Check the network related parameter which is downloaded to the CPU.

Error Code	Description	Corrective action
16#8235	The CPU parameter downloaded is incorrect. The static ARP table is set incorrectly.	1. Check the Ethernet parameters of the CPU in HWCONFIG. 2. Check if CPU firmware version matches the HWCONFIG version
16#8522	Auto scanning is in progress	Auto scanning of module configuration is in progress. Please try again later.
16#853B	An I/O module is not configured.(write error)	Check if the module configuration in HWCONFIG is correct.
16#853C	An I/O module does not exist. (write error)	Check if the module configuration in HWCONFIG is correct.
16#854B	An I/O module is not configured. (read error)	Check if the module configuration in HWCONFIG is correct.
16#854C	An I/O module does not exist. (read error)	Check if the module configuration in HWCONFIG is correct.
16#9B21	An error occurs when COM2 communicates with slave 1 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B22	An error occurs when COM2 communicates with slave 2 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B23	An error occurs when COM2 communicates with slave 3 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B24	An error occurs when COM2 communicates with slave 4 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B25	An error occurs when COM2 communicates with slave 5 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B26	An error occurs when COM2 communicates with slave 6 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B27	An error occurs when COM2 communicates with slave 7 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B28	An error occurs when COM2 communicates with slave 8 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B29	An error occurs when COM2 communicates with slave 9 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2A	An error occurs when COM2 communicates with slave 10 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2B	An error occurs when COM2 communicates with slave 11 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2C	An error occurs when COM2 communicates with slave 12 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2D	An error occurs when COM2 communicates with slave 13 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2E	An error occurs when COM2 communicates with slave 14 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2F	An error occurs when COM2 communicates with slave 15 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B30	An error occurs when COM2 communicates with slave 16 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B31	An error occurs when COM2 communicates with slave 17 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B32	An error occurs when COM2 communicates with slave 18 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B33	An error occurs when COM2 communicates with slave 19 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B34	An error occurs when COM2 communicates with slave 20 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B35	An error occurs when COM2 communicates with slave 21 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B36	An error occurs when COM2 communicates with slave 22 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B37	An error occurs when COM2 communicates with slave 23 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B38	An error occurs when COM2 communicates with slave 24 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B39	An error occurs when COM2 communicates with slave 25 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3A	An error occurs when COM2 communicates with slave 26 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

A5

Error Code	Description	Corrective action
16#9B3B	An error occurs when COM2 communicates with slave 27 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3C	An error occurs when COM2 communicates with slave 28 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3D	An error occurs when COM2 communicates with slave 29 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3E	An error occurs when COM2 communicates with slave 30 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3F	An error occurs when COM2 communicates with slave 31 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B40	An error occurs when COM2 communicates with slave 32 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B41	COM2 receives no response from slave 1 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B42	COM2 receives no response from slave 2 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B43	COM2 receives no response from slave 3 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B44	COM2 receives no response from slave 4 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B45	COM2 receives no response from slave 5 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B46	COM2 receives no response from slave 6 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B47	COM2 receives no response from slave 7 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B48	COM2 receives no response from slave 8 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B49	COM2 receives no response from slave 9 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4A	COM2 receives no response from slave 10 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4B	COM2 receives no response from slave 11 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4C	COM2 receives no response from slave 12 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4D	COM2 receives no response from slave 13 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4E	COM2 receives no response from slave 14 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

A5

Error Code	Description	Corrective action
16#9B4F	COM2 receives no response from slave 15 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B50	COM2 receives no response from slave 16 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B51	COM2 receives no response from slave 17 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B52	COM2 receives no response from slave 18 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B53	COM2 receives no response from slave 19 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B54	COM2 receives no response from slave 20 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B55	COM2 receives no response from slave 21 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B56	COM2 receives no response from slave 22 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B57	COM2 receives no response from slave 23 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B58	COM2 receives no response from slave 24 by MODBUS.	<ol style="list-style-type: none"> 1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B59	COM2 receives no response from slave 25 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5A	COM2 receives no response from slave 26 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5B	COM2 receives no response from slave 27 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5C	COM2 receives no response from slave 28 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5D	COM2 receives no response from slave 29 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5E	COM2 receives no response from slave 30 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5F	COM2 receives no response from slave 31 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B60	COM2 receives no response from slave 32 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

A5

Analog I/O Modules and Temperature Measurement Modules

Error code	Description	Corrective action
16#A000	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A001	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.

Error code	Description	Corrective action
16#A002	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A003	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.
16#A004	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A005	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 5 exceeds the range of inputs which can be received by the hardware.
16#A006	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A007	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.
16#A400	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A401	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.
16#A402	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A403	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.

Error code	Description	Corrective action
16#A404	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A405	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 5 exceeds the range of inputs which can be received by the hardware.
16#A406	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A407	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.
16#A600	Hardware failure	1. Check whether the backplane is normal. 2. Check whether the module operate normally.
16#A601	The external voltage is abnormal.	Check whether the external 24 V power supply to the module is normal.
16#A602	Internal error The CJC is abnormal.	Please contact the manufacturer.
16#A603	Internal error The factory correction is abnormal.	Please contact the manufacturer.
16#A800	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A801	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.
16#A802	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A803	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.

A5

Error code	Description	Corrective action
16#A804	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A805	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.
16#A806	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A807	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.

AH02HC-5A/AH04HC-5A

Error code	Description	Corrective action
16#A001	The linear accumulation in channel 1 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR0 to ON by means of FROM/TO.
16#A002	The scale set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 1 should be in the range of 0 to 32767.
16#A003	The number of cycles set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 1 should be in the range of 2 to 60.
16#A004	The comparison value set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 1 should be in the range of -999999999 to 999999999.
16#A005	A limit value set for channel 1 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 1 should be in the range of -200000 to 200000.
16#A006	The interrupt number set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 1 should be in the range of 0 to 31.
16#A011	The linear accumulation in channel 1 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR28 to ON by means of FROM/TO.
16#A012	The scale set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 2 should be in the range of 0 to 32767.

Error code	Description	Corrective action
16#A013	The number of cycles set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 2 should be in the range of 2 to 60.
16#A014	The comparison value set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 2 should be in the range of -999999999 to 999999999.
16#A015	A limit value set for channel 2 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 2 should be in the range of -200000 to 200000.
16#A016	The interrupt number set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 2 should be in the range of 0 to 31.
16#A021	The linear accumulation in channel 3 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR56 to ON by means of FROM/TO.
16#A022	The scale set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 3 should be in the range of 0 to 32767.
16#A023	The number of cycles set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 3 should be in the range of 2 to 60.
16#A024	The comparison value set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 3 should be in the range of -999999999 to 999999999.
16#A025	A limit value set for channel 3 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 3 should be in the range of -200000 to 200000.
16#A026	The interrupt number set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 3 should be in the range of 0 to 31.
16#A031	The linear accumulation in channel 4 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR84 to ON by means of FROM/TO.
16#A032	The scale set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 4 should be in the range of 0 to 32767.
16#A033	The number of cycles set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 4 should be in the range of 2 to 60.
16#A034	The comparison value set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 4 should be in the range of -999999999 to 999999999.

Error code	Description	Corrective action
16#A035	A limit value set for channel 4 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 4 should be in the range of -200000 to 200000.
16#A036	The interrupt number set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 4 should be in the range of 0 to 31.

AH05PM-5A/AH10PM-5A/AH15PM-5A

The programs and the setting which are mentioned in the table below are edited in PMSOFT version 2.02 or above.

Error code	Description	Corrective action
16#A002	The subroutine has no data.	A program should be written in the subroutine.
16#A003	CJ, CJN, and JMP have no matching pointers.	Write the pointers which match CJ, CJN, and JMP respectively.
16#A004	There is a subroutine pointer in the main program.	The subroutine pointer can not be in the main program.
16#A005	Lack of the subroutine	The nonexistent subroutine can not be called.
16#A006	The pointer is used repeatedly in the same program.	The pointer can not be used repeatedly in the same program.
16#A007	The subroutine pointer is used repeatedly.	The subroutine pointer can not be used repeatedly.
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	The pointer used in JMP can not be used repeatedly in different subroutines.
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	The pointer used in JMP can not be the same as the pointer used in CALL.
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	The pointer used in JMP can not be the same as a subroutine pointer.
16#A00B	Target position (I) of the single speed is incorrect.	The target position (I) of the single speed should be set correctly.
16#A00C	Target position (II) of the single-axis motion is incorrect.	Check whether target position (II) of the single-axis motion and target position (I) of the single-axis motion are in opposite directions.
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Set the speed of the single-axis motion.
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	The setting value can not be zero.
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Set the speed of returning to zero properly. (The setting value can not be zero.)
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Set the speed of returning to zero. The deceleration should be less than the speed of returning to zero. (The setting value can not be zero.)
16#A011	The setting of the JOG speed is incorrect.	The setting value can not be zero.
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.

Error code	Description	Corrective action
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A014	The limit switch is reached.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A015	The device which is used exceeds the device range.	Use the device which does not exceed the device range.
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Use the 16-bit index register/32-bit index register which does not exceed the device range.
16#A018	The conversion into the floating-point number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A019	The conversion into the binary-coded decimal number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A01A	Incorrect division operation (The divisor is 0.)	Modify the operation to prevent the divisor from being zero.
16#A01B	General program error	Modify the program to make the syntax correct.
16#A01C	LD/LDI has been used more than nine times.	Modify the program to prevent LD/LDI from being used more than nine times.
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Modify the program to prevent more than one level of nested program structure supported by RPT/RPE from being used.
16#A01E	SRET is used between RPT and RPE.	Modify the program to prevent SRET from being used between RPT and RPE.
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Modify the program so that there is M102 in the main program, or modify the program so that there is M2 in the motion program.
16#A020	The wrong instruction is used, or the device used exceeds the range.	Check and modify the program to prevent the wrong instruction from being used, or check whether the device used exceeds the device range.

AH20MC-5A

The programs and the setting which are mentioned in the table below are edited in PMSOFT version 2.02 or above.

Error code	Description	Corrective action
16#A002	The subroutine has no data.	A program should be written in the subroutine.
16#A003	CJ, CJN, and JMP have no matching pointers.	Write the pointers which match CJ, CJN, and JMP respectively.

Error code	Description	Corrective action
16#A004	There is a subroutine pointer in the main program.	The subroutine pointer can not be in the main program.
16#A005	Lack of the subroutine	The nonexistent subroutine can not be called.
16#A006	The pointer is used repeatedly in the same program.	The pointer can not be used repeatedly in the same program.
16#A007	The subroutine pointer is used repeatedly.	The subroutine pointer can not be used repeatedly.
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	The pointer used in JMP can not be used repeatedly in different subroutines.
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	The pointer used in JMP can not be the same as the pointer used in CALL.
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	The pointer used in JMP can not be the same as a subroutine pointer.
16#A00B	Target position (I) of the single speed is incorrect.	The target position (I) of the single speed should be set correctly.
16#A00C	Target position (II) of the single-axis motion is incorrect.	Check whether target position (II) of the single-axis motion and target position (I) of the single-axis motion are in opposite directions.
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Set the speed of the single-axis motion.
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	The setting value can not be zero.
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Set the speed of returning to zero properly. (The setting value can not be zero.)
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Set the speed of returning to zero. The deceleration should be less than the speed of returning to zero. (The setting value can not be zero.)
16#A011	The setting of the JOG speed is incorrect.	The setting value can not be zero.
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A014	The limit switch is reached.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.

Error code	Description	Corrective action
16#A015	The device which is used exceeds the device range.	Use the device which does not exceed the device range.
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Use the 16-bit index register/32-bit index register which does not exceed the device range.
16#A018	The conversion into the floating-point number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A019	The conversion into the binary-coded decimal number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A01A	Incorrect division operation (The divisor is 0.)	Modify the operation to prevent the divisor from being zero.
16#A01B	General program error	Modify the program to make the syntax correct.
16#A01C	LD/LDI has been used more than nine times.	Modify the program to prevent LD/LDI from being used more than nine times.
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Modify the program to prevent more than one level of nested program structure supported by RPT/RPE from being used.
16#A01E	SRET is used between RPT and RPE.	Modify the program to prevent SRET from being used between RPT and RPE.
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Modify the program so that there is M102 in the main program, or modify the program so that there is M2 in the motion program.
16#A020	The wrong instruction is used, or the device used exceeds the range.	Check and modify the program to prevent the wrong instruction from being used, or check whether the device used exceeds the device range.

AH10SCM-5A

Error code	Description	Corrective action
16#A002	The setting of the UD Link is incorrect, or the communication fails.	Check the setting in SCMSOFT, and download the setting again.
16#A401	Hardware error	Please contact the manufacturer.
16#A804	The communication through the communication port is incorrect.	<ol style="list-style-type: none"> 1. Check whether the communication cable is connected well. 2. Check the parameter in HWCONFIG, and the parameter. Download the parameter again.
16#A808	MODBUS communication error	<ol style="list-style-type: none"> 1. Check whether the communication cable is connected well. 2. Check the parameter in HWCONFIG, and the parameter. Download the parameter again.

AH10COPM-5A

Error code	Description	Corrective action
16#A0B0	AH10COPM-5A does not send a heartbeat message after a set period of time.	Check whether the bus cable on the CANopen network created is connected correctly.
16#A0B1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Set the length of the PDO in the slave station again, and then download the setting to AH10COPM-5A.
16#A0B2	The master station selected does not send a node guarding message after a set period of time.	Check whether the bus cable on the CANopen network created is connected correctly.
16#A0E0	AH10COPM-5A receives an emergency message from a slave station.	Use the function block CANopen_EMCY to read relevant information.
16#A0E1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Set the length of the PDO in the slave station again, and then download the setting to AH10COPM-5A.
16#A0E2	AH10COPM-5A does not receive a PDO from a slave station.	Make sure that the PDOs in the slave station are set correctly.
16#A0E3	An automatic SDO is not downloaded successfully.	Make sure that the automatic SDO is set correctly.
16#A0E4	A PDO parameter is not set successfully.	Make sure that the setting of the PDO parameter is legal.
16#A0E5	A key parameter is set incorrectly.	Make sure that the slave stations connected are the same as the slave stations set.
16#A0E6	The actual network configuration is not the same as the network configuration set.	Make sure that the power supplied to the slave stations connected is normal and the network created is connected correctly.
16#A0E7	The control of the errors in a slave station is not sent after a set period of time.	
16#A0E8	The master station address is the same as a slave station address.	Set the master station address or the slave station address again, and make sure the new station address is not the same as a slave station address.
16#A0F1	No slave station is added to the node list in CANopen builder.	Add slave stations to the node list, and download the configuration to AH10COPM-5A.
16#A0F3	An error occurs in AH10COPM-5A.	Download parameters again. If the error still occurs, please replace AH10COPM-5A.
16#A0F4	The bus used is off.	Please check whether the bus cable on the CANopen

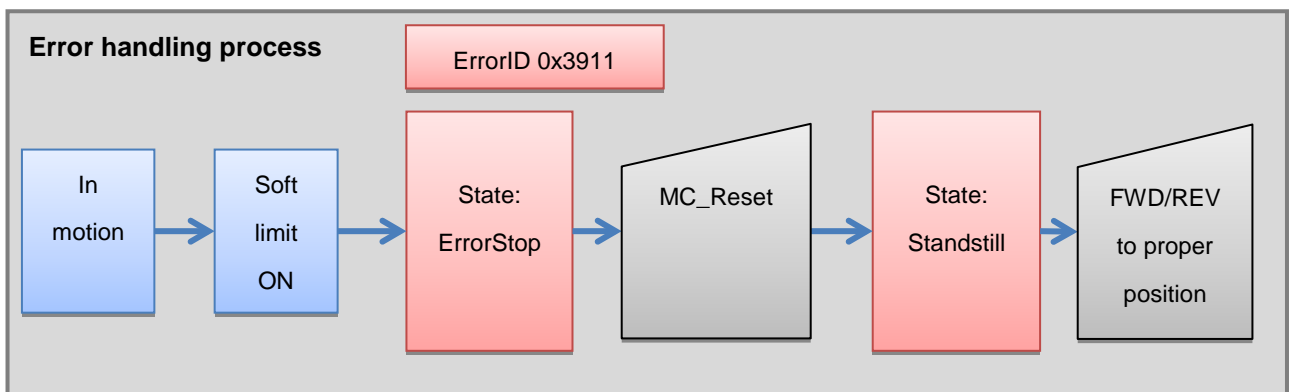
Error code	Description	Corrective action
		network created is connected correctly, make sure that the serial transmission speeds of all the nodes on the network are the same, and power AH10COPM-5A again.
16#A0F5	The node address of AH10COPM-5A is set incorrectly.	The node address of AH10COPM-5A must be in the range of 1 to 127.
16#A0F6	Internal error: An error occurs in the manufacturing process in the factory.	Power AH10COPM-5A again. If the error still occurs, please replace AH10COPM-5A.
16#A0F7	Internal error: GPIO error	
16#A0F8	Hardware error	
16#A0F9	Low voltage	Make sure that the power supplied to AH10COPM-5A is normal.
16#A0FA	An error occurs in the firmware of AH10COPM-5A.	Power AH10COPM-5A again.
16#A0FB	The transmission registers in AH10COPM-5A are full.	Please make sure that the bus cable on the CANopen network created is connected correctly, and power AH10COPM-5A again.
16#A0FC	The reception registers in AH10COPM-5A are full.	Please make sure that the bus cable on the CANopen network created is connected correctly, and power AH10COPM-5A again.

A5

A.5.3. Troubleshooting for Limitation Errors

Troubleshooting for the software limit errors

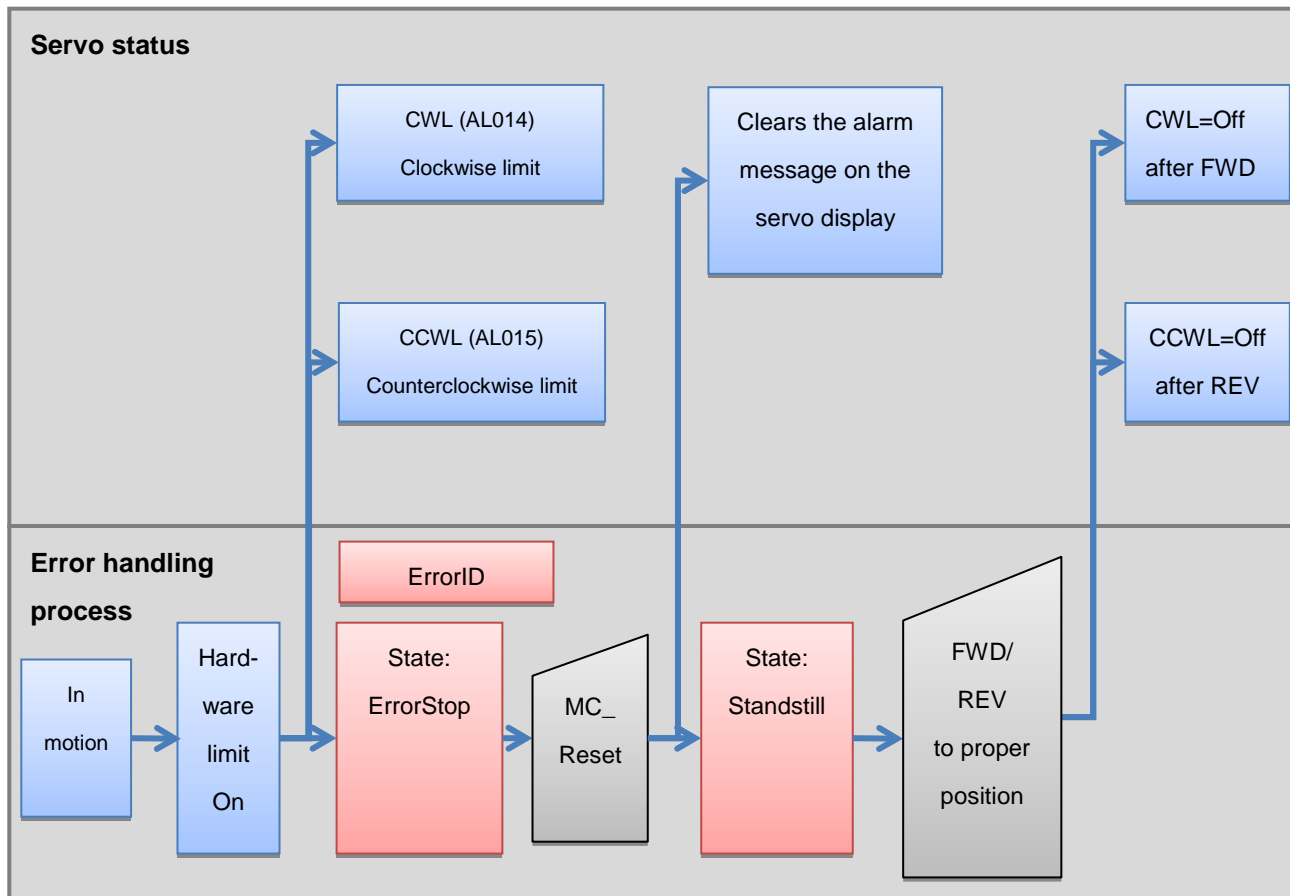
The controller system checks the software limits before or during the motion by the error code 0x3911. When the operation exceeds the software limits, the error code will be indicated and the axis will enter “ErrorStop”. Note that servo drive will not report this error since the error handling in this case is controlled by the controller.



Troubleshooting for the hardware limit errors

When the servo drive is driving a motion, the servo will stop when CWL(Clockwise limit) or CCWL(Counterclockwise limit)

is On, no matter it's running forward or reversely. AL014(CWL) or AL015(CCWL) will indicate such error.



A.6. Table of Data Type Unit (DUT): Enum

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_OUTTYPE	0: UD 1: PD 2: AB	Setting pulse output type 0: clockwise / counterclockwise pulse output(counting up/down) 1: Pulse+Direction 2: A/B-phase	DFB_AxisSetting2 Interface: <i>OutputType</i>
eDFB_UNIT	0: Motor 1: Machine 2: Compound	Unit setting of the coordinate system 0: motot unit 1: mechanical system 2: compound unit	FB: DFB_AxisSetting2 Interface: <i>Unit</i>
eDFB_MODE	4096: AxisIdle 256: AxisStopping 4353: AbsSeg1 4354: RelSeg1 4355: AbsSeg2 4356: RelSeg2 4357: TrSeg1 4358: Jog 4359: Mpg 4362: GearIn 4363: CamIn 4608: GcodeStopping 4609: GcodeRun 4864: InterpolationStopping	0x000: axis indling 0x100: axis stopping 0x101: absolute single-speed motion 0x102: relative single-speed motion 0x103: absolute two-speed motion 0x104: relative two-speed motion 0x105: triggering single-speed motion 0x107: Jog motion 0x108: manual pulse generator 0x10A: electronic gear 0x10B: electronic cam 0x200: G-code stopping 0x201: G-code running 0x300: interpolation stopping	DFB_AxisStatus Interface: <i>Mode</i>
eDFB_SDODataType	0: mc16bits 1: mc32bits	0: writing in 16-bit data 1: writing in 32-bit data	DFB_ECATServoWrite Interface: <i>DataType</i>

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_SELECT_DEV	0: M_DEV 5: D_DEV 6: W_DEV 7: ALL	0: reading M devices from SD card 5: reading D devices from SD card 6: reading W devices from SD card 7: reading (M/D/W) devices from SD card	DFB_SDDevRead Interface: <i>Device</i>
eDFB_ENGAGE_TYPE	0: ByCapture0 1: ByCapture1 2: ByCapture2 3: ByCapture3 4: ByCapture4 5: ByCapture5 6: ByCapture6 7: ByCapture7 -1: Direct	0: engaging when Capture 0 is triggered 1: engaging when Capture 1 is triggered 2: engaging when Capture 2 is triggered 3: engaging when Capture 3 is triggered 4: engaging when Capture 4 is triggered 5: engaging when Capture 5 is triggered 6: engaging when Capture 6 is triggered 7: engaging when Capture 7 is triggered -1: engaging directly	DFB_GearIn2/DFB_CamIn2 Interface: <i>extTrgCAPno</i>
eDFB_ACC_CURVE	0: Polynomial_0order 1: Polynomial_1order 2: SingleHypot 3: Cycloid	Acceleration curve type 0: 0-order polynomial (constant) curve 1: 1st order polynomial curve 2: single hypotenuse curve 3: cycloid curve	DFB_CamCurve/ DFB_CamCurve2/ DFB_FlyCut2/ DFB_HorizontalFlowWrapper Interface: <i>AccCurve</i>

A6

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_GEN_CURVE	0: leftCAM 5: rightCAM 1: midCAMall 9: midCAMzero 7: midCAMbegin 8: midCAMend	Cam curve type 0: left cam 5: right cam 1: middle cam 9: middle cam zero 7: middle cam begins 8: middle cam ends	DFB_CamCurve / DFB_CamCurve2/ DFB_FlyCut2 Interface: <i>eCamCurve</i>
eDFB_HCNT	0: AC0 1: AC4 2: AC8 3: AC12 4: AC16 6: AC20	High speed counters for motion control 0: high speed counter 1 1: high speed counter 2 2: high speed counter 3 3: high speed counter 4 4: high speed counter 5 6: high speed counter 6	DFB_HCnt Interface: <i>Channel</i>
eDFB_HCNT_INTYPE	0: UD 1: PD 2: AB 3: AB4	Setting pulse input type 0: clockwise / counterclockwise pulse output(counting up/down) 1: Pulse+Direction 2: A/B-phase 3: 4A/B-phase	DFB_HCnt Interface: <i>InputType</i>
eDFB_HTMR	0: AC0 1: AC4 2: AC8 3: AC12	High speed timers for motion control 0: high speed timer 1 1: high speed timer 2 2: high speed timer 3 3: high speed timer 4	DFB_HTmr Interface: <i>Channel</i>

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_COMP	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7	0: channel 0 1: channel 1 2: channel 2 3: channel 3 4: channel 4 5: channel 5 6: channel 6 7: channel 7	DFB_Compare Interface: <i>Channel</i>
eDFB_COMP_SOURCE	0: Axis1 1: Axis2 2: Axis3 3: Axis4 4: AC0 5: AC4 6: AC8 7: AC12 8: AC16	0: Axis 1 1: Axis 2 2: Axis 3 3: Axis 4 4: high speed counter 1 5: high speed counter 2 6: high speed counter 3 7: high speed counter 4 8: high speed counter 5	DFB_Compare Interface: <i>Source</i>
eDFB_COMP_MODE	0: Equal 1: Bigger_Equal 2: Smaller_Equal	0: equal 1: bigger or equal 2: smaller or equal	DFB_Compare Interface: <i>Mode</i>
eDFB_COMP_OUTDEV	0: SetY08 1: SetY09 2: SetY10 3: SetY11 4: RstAC0 5: RstAC4 6: RstAC8 7: RstAC12	0: set Y0.8 1: set Y0.9 2: set Y0.10 3: set Y0.11 4: reset the value of high speed counter 1 5: reset the value of high speed counter 2 6: reset the value of high speed counter 3 7: reset the value of high speed counter 4	DFB_Compare Interface: <i>OutPutDevice</i>

A6

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_CAP	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7	0: channel 0 1: channel 1 2: channel 2 3: channel 3 4: channel 4 5: channel 5 6: channel 6 7: channel 7	DFB_Capture/ DFB_Capture2 Interface: <i>Source</i>
eDFB_CAP_TRIG_DEV	0: X0p0 1: X0p1 2: X0p2 3: X0p3 8: X0p8 9: X0p9 10: X0p11 11: X0p11 12: X0p12 13: X0p13 14: X0p14	0: trigger by X0.0 signal 1: trigger by X0.1 signal 2: trigger by X0.2 signal 3: trigger by X0.3 signal 8: trigger by X0.8 signal 9: trigger by X0.9 signal 10: trigger by X0.10 signal 11: trigger by X0.11 signal 12: trigger by X0.12 signal 13: trigger by X0.13 signal 14: trigger by X0.14 signal	DFB_Capture/ DFB_Capture2 Interface: <i>TriggerDevice</i>
eDFB_CAP_SOURCE	0: Axis1 1: Axis2 2: Axis3 3: Axis4 4: AC0 5: AC4 6: AC8 7: AC12 8: AC16	0: capture axis 1 1: capture axis 2 2: capture axis 3 3: capture axis 4 4: capture high speed counter 1 5: capture high speed counter 2 6: capture high speed counter 3 7: capture high speed counter 4 8: capture high speed counter 5	DFB_Capture/ DFB_Capture2 Interface: <i>Source</i>

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_HALT_CLK_SOURCE	0: slaveEOP 1: masterEOP 2: extern	0: end point of slave cam 1: end point of master cam 2: external input of the function block	DFB_FlyCut2 Interface: <i>Halt_ClkSource</i>
eMC_STATE_MACHINE	0: Unknown 1: ErrorStop 2: Disabled 3: Standstill 4: Homing 5: Stopping 6: ContinuousMotion 7: SynchronizedMotion 8: DiscreteMotion 9: Coordinated 10: CoordinatedHalt 11: CoordinatedStop	0: Unknown 1: ErrorStop 2: Disabled 3: Standstill 4: Homing 5: Stopping 6: ContinuousMotion 7: SynchronizedMotion 8: DiscreteMotion 9: Coordinated 10: CoordinatedHalt 11: CoordinatedStop	-
eMC_GROUP_STATE_MACHINE	0: GroupDisable 256: GroupStandby 512: GroupStopping 576: GroupMotion 768: GroupErrorStop	0: GroupDisable 256: GroupStandby 512: GroupStopping 576: GroupMotion 768: GroupErrorStop	
eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0: aborting the ongoing motion 1: buffering when ongoing motion is done 2: blending with the lowest velocity 3: blending with the velocity of the previous motion 4: blending with the velocity of the next motion 5: blending with the highest velocity	Interface: <i>BufferMode</i>

A6

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eMC_DIRECTION	1: mcPositiveDirection 2: mcShortestWay 3: mcNegativeDirection 4: mcCurrentDirection	1: positive direction 2: shortest way 3: negative direction 4: current direction	Interface: <i>Direction</i>
eMC_SOURCE	0: mcCommandedValue 1: mcSetValue 2: mcActualValue	0: command value 1: set value 2: actual value	MC_ReadMotionState Interface: <i>Source</i> MC_CamIn/ MC_GearIn/ MC_GearInPos Interface: <i>MasterValueSource</i> MC_CombineAxes Interface: <i>MasterValueSourceM1/</i> <i>MasterValueSourceM2</i> MC_DigitalCamSwitch Interface: <i>ValueSource</i>
eMC_SYNC_MODE	1: mcRampIn_Shortest 2: mcRampIn_Positive 3: mcRampIn_Negative	1: (reserved) 2: (reserved) 3: (reserved)	MC_GearInPos Interface: <i>SyncMode</i>
eMC_START_MODE	0: mcJump 1: mcRampIn_Shortest 2: mcRampIn_Positive 3: mcRampIn_Negative 4: mcAbsolute 5: mcRelative	0: jump in immediately 1: shortest path 2: positive path 3: negative path 4: (reserved) 5: (reserved)	MC_CamIn Interface: <i>StartMode</i>
eMC_COMBINE_MODE	0: mcAddAxes 1: mcSubAxes	0: adding 1: subtracting	MC_CombineAxes Interface: <i>CombineMode</i>
eMC_SERVOOFF_MOD E	0: mcAborting 1: mcBuffered	0: Aborting 1: mcBuffered	MC_Power Interface: <i>Mode</i>

MEMO

A6