MITSUBISHI ELECTRIC

PROGRAMMABLE CONTROLLERS

MELSEC iQ-F series

# MELSEC iQ-F
# FX5 Programming Manual
# (Instructions, Standard Functions/Function Blocks)

# SAFETY PRECAUTIONS

(Read these precautions before use.)
Before using the MELSEC iQ-F series PLCs, please read the manual supplied with each product and the relevant manuals introduced in that manual carefully and pay full attention to safety to handle the product correctly.
Store this manual in a safe place so that it can be taken out and read whenever necessary. Always forward it to the end user.

# INTRODUCTION

This manual describes the instructions and functions/function blocks required for programming MELSEC iQ-F series systems. This manual and the related manuals should be read and the functions and performance of the MELSEC iQ-F series PLC should be understood before attempting to use the unit.
However, before using a program example introduced in this manual to the actual system, always confirm that it poses no problem for control of the target system.

## Regarding use of this product

- This product has been manufactured as a general-purpose part for general industries, and has not been designed or manufactured to be incorporated in a device or system used in purposes related to human life.
- Before using the product for special purposes such as nuclear power, electric power, aerospace, medicine or passenger movement vehicles, consult with Mitsubishi Electric.
- This product has been manufactured under strict quality control. However when installing the product where major accidents or losses could occur if the product fails, install appropriate backup or failsafe functions in the system.

## Note

- If in doubt at any stage during the installation of the product, always consult a professional electrical engineer who is qualified and trained in the local and national standards. If in doubt about the operation or use, please consult the nearest Mitsubishi Electric representative.
- Since the examples indicated by this manual, technical bulletin, catalog, etc. are used as a reference, please use it after confirming the function and safety of the equipment and system. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.
- This manual content, specification etc. may be changed without a notice for improvement.
- The information in this manual has been carefully checked and is believed to be accurate; however, if you have noticed a doubtful point, an error, etc., please contact the nearest Mitsubishi Electric representative. When doing so, please provide the manual number given at the end of this manual.

# CONTENTS

# PART 3 CPU MODULE INSTRUCTIONS

# PART 4 MODULE DEDICATED INSTRUCTION

# RELEVANT MANUALS

## User's manuals for the applicable modules

| Manual name <manual number> | Description |
|---|---|
| MELSEC iQ-F FX5 User's Manual (Startup)<br><JY997D58201> | Performance specifications, procedures before operation, and troubleshooting of the CPU module. |
| MELSEC iQ-F FX5U User's Manual (Hardware)<br><JY997D55301> | Describes the details of hardware of the FX5U CPU module, including input/output specifications, wiring, installation, and maintenance. |
| MELSEC iQ-F FX5UC User's Manual (Hardware)<br><JY997D61401> | Describes the details of hardware of the FX5UC CPU module, including input/output specifications, wiring, installation, and maintenance. |
| MELSEC iQ-F FX5 User's Manual (Application)<br><JY997D55401> | Describes basic knowledge required for program design, functions of the CPU module, devices/labels, and parameters. |
| MELSEC iQ-F FX5 Programming Manual (Program Design)<br><JY997D55701> | Describes specifications of ladders, ST, FBD/LD, and other programs and labels. |
| MELSEC iQ-F FX5 Programming Manual (Instructions, Standard Functions/Function Blocks)<br><JY997D55801> (This manual) | Describes specifications of instructions and functions that can be used in programs. |
| MELSEC iQ-F FX5 User's Manual (Serial Communication)<br><JY997D55901> | Describes N:N network, MELSEC Communication protocol, inverter communication, non-protocol communication, and predefined protocol support. |
| MELSEC iQ-F FX5 User's Manual (MELSEC Communication Protocol)<br><JY997D60801> | Explains methods for the device that is communicating with the CPU module by MC protocol to read and write the data of the CPU module. |
| MELSEC iQ-F FX5 User's Manual (MODBUS Communication)<br><JY997D56101> | Describes MODBUS serial communication. |
| MELSEC iQ-F FX5 User's Manual (Ethernet Communication)<br><JY997D56201> | Describes the functions of the built-in Ethernet port communication function. |
| MELSEC iQ-F FX5 User's Manual (SLMP)<br><JY997D56001> | Explains methods for the device that is communicating with the CPU module by SLMP to read and write the data of the CPU module. |
| MELSEC iQ-F FX5 User's Manual (Positioning Control)<br><JY997D56301> | Describes the built-in positioning function. |
| MELSEC iQ-F FX5 User's Manual (Analog Control)<br><JY997D60501> | Describes the analog function. |
| GX Works3 Operating Manual<br><SH-081215ENG> | System configuration, parameter settings, and online operations of GX Works3. |

# TERMS

Unless otherwise specified, this manual uses the following terms.

- ☐ indicates a variable part to collectively call multiple models or versions.

(Example) FX5U-32MR/ES, FX5U-32MT/ES ⇨ FX5U-32M☐/ES

- For details of the FX3 devices that can be connected with the FX5, refer to the User's Manual (Hardware) of the CPU module to be used.

| Terms | Description |
|---|---|
| ■Devices | |
| FX5 | Generic term for FX5U and FX5UC PLCs |
| FX3 | Generic term for FX3S, FX3G, FX3GC, FX3U, and FX3UC PLCs |
| FX5 CPU module | Generic term for FX5U CPU module and FX5UC CPU module |
| FX5U CPU module | Generic term for FX5U-32MR/ES, FX5U-32MT/ES, FX5U-32MT/ESS, FX5U-64MR/ES, FX5U-64MT/ES, FX5U-64MT/ESS, FX5U-80MR/ES, FX5U-80MT/ES, and FX5U-80MT/ESS |
| FX5UC CPU module | Generic term for FX5UC-32MT/D and FX5UC-32MT/DSS |
| Extension module | Generic term for FX5 extension modules and FX3 function modules |
| • FX5 extension module | Generic term for I/O modules, FX5 extension power supply module, and FX5 intelligent function module |
| • FX3 extension module | Generic term for FX3 extension power supply module and FX3 intelligent function module |
| Extension module (extension cable type) | Input modules (extension cable type), Output modules (extension cable type), Bus conversion module (extension cable type), and Intelligent function modules |
| Extension module (extension connector type) | Input modules (extension connector type), Output modules (extension connector type), Input/output modules, Bus conversion module (extension connector type), and Connector conversion module (extension connector type) |

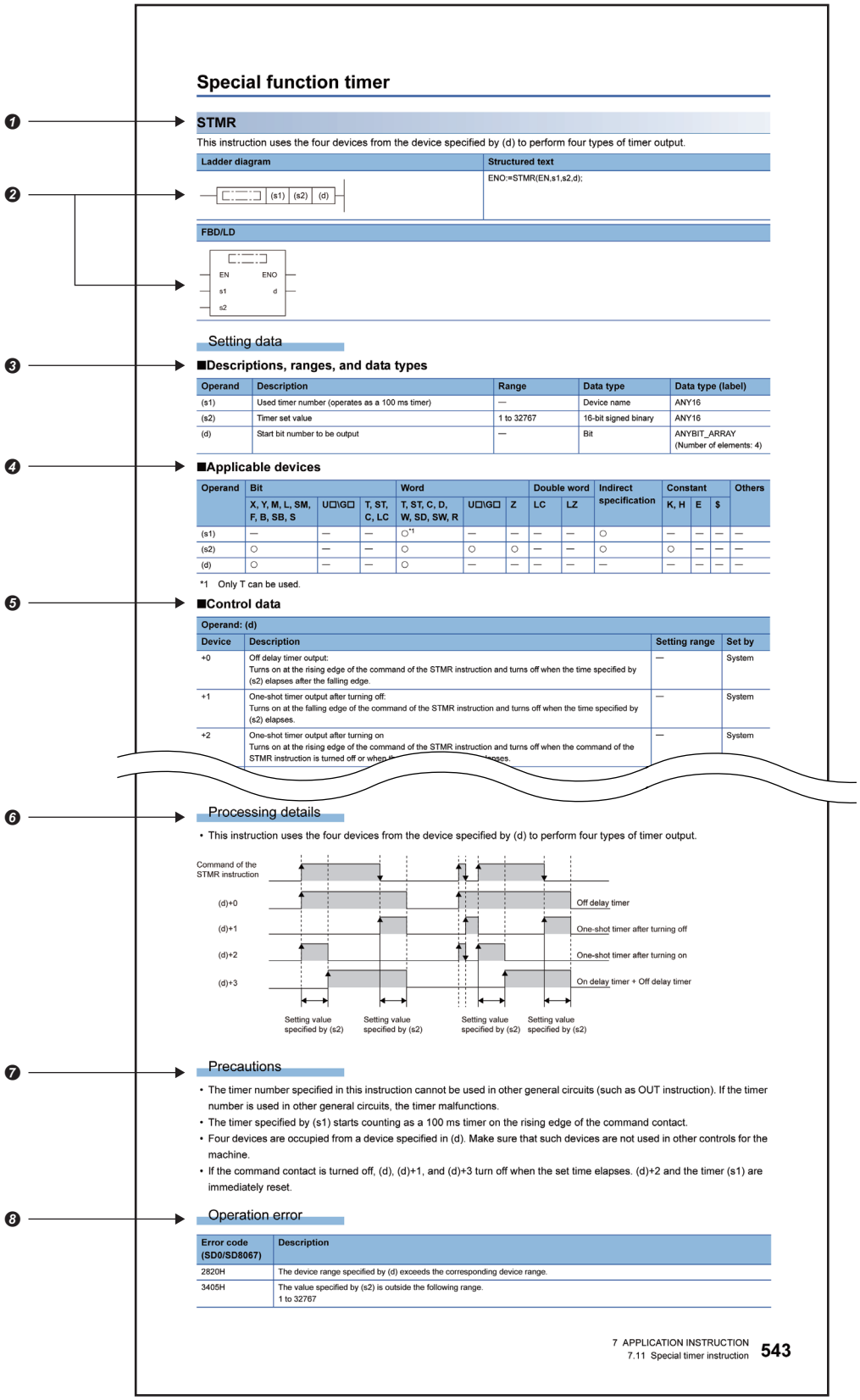
| Terms | Description |
|---|---|
| I/O module | Generic term for input modules, output modules, Input/output modules, and powered input/output modules |
| Input module | Generic term for Input modules (extension cable type) and Input modules (extension connector type) |
| • Input module (extension cable type) | Generic term for FX5-8EX/ES and FX5-16EX/ES |
| • Input module (extension connector type) | Generic term for FX5-C32EX/D and FX5-C32EX/DS |
| Output module | Generic term for output modules (extension cable type) and output modules (extension connector type) |
| • Output module (extension cable type) | Generic term for FX5-8EYR/ES, FX5-8EYT/ES, FX5-8EYT/ESS, FX5-16EYR/ES, FX5-16EYT/ES, and FX5-16EYT/ESS |
| • Output module (extension connector type) | Generic term for FX5-C32EYT/D and FX5-C32EYT/DSS |
| Input/output modules | Generic term for FX5-C32ET/D and FX5-C32ET/DSS |
| Powered input/output module | Generic term for FX5-32ER/ES, FX5-32ET/ES, and FX5-32ET/ESS |
| Extension power supply module | Generic term for FX5 extension power supply module and FX3 extension power supply module |
| • FX5 extension power supply module | Different name for FX5-1PSU-5V |
| • FX3 extension power supply module | Different name for FX3U-1PSU-5V |
| Intelligent module | The abbreviation for intelligent function modules |
| Intelligent function module | Generic term for FX5 intelligent function modules and FX3 intelligent function modules |
| • FX5 intelligent function module | Generic term for FX5 intelligent function modules |
| • FX3 intelligent function module | Different name for FX3 special function blocks |
| Simple motion module | Different name for FX5-40SSC-S |
| Expansion board | Generic term for board for FX5U CPU module |
| • Communication board | Generic term for FX5-232-BD, FX5-485-BD, and FX5-422-BD-GOT |
| Expansion adapter | Generic term for adapter for FX5 CPU module |
| • Communication adapter | Generic term for FX5-232ADP and FX5-485ADP |
| • Analog adapter | Generic term for FX5-4AD-ADP and FX5-4DA-ADP |
| Bus conversion module | Generic term for Bus conversion module (extension cable type) and Bus conversion module (extension connector type) |
| • Bus conversion module (extension cable type) | Different name for FX5-CNV-BUS |
| • Bus conversion module (extension connector type) | Different name for FX5-CNV-BUSC |
| Battery | Different name for FX3U-32BL |
| SD memory card | Generic term for NZ1MEM-2GBSD, NZ1MEM-4GBSD, L1MEM-2GBSD and L1MEM-4GBSD SD memory cards<br>Abbreviation of Secure Digital Memory Card. Device that stores data using flash memory. |
| Peripheral device | Generic term for engineering tools and GOTs |
| GOT | Generic term for Mitsubishi Graphic Operation Terminal GOT1000 and GOT2000 series |
| ■Software packages | |
| Engineering tool | The product name of the software package for the MELSEC programmable controllers |
| GX Works3 | The product name of the software package, SWnDND-GXW3, for the MELSEC programmable controllers (The 'n' represents a version.) |

# HOW TO READ THIS MANUAL

The following describes the page layout and symbols used in this manual.

## How to read PART 3 and PART 4

The contents described in this section are provided only for explaining how to read this manual. Thus, the actual description may differ.



### Special function timer

❶ **STMR**

This instruction uses the four devices from the device specified by (d) to perform four types of timer output.

| Ladder diagram | Structured text |
|---|---|
| (s1) (s2) (d) | ENO:=STMR(EN,s1,s2,d); |

FBD/LD

EN ENO
s1 d
s2

❸ Setting data

■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Used timer number (operates as a 100 ms timer) | — | Device name | ANY16 |
| (s2) | Timer set value | 1 to 32767 | 16-bit signed binary | ANY16 |
| (d) | Start bit number to be output | — | Bit | ANYBIT_ARRAY (Number of elements: 4) |

❹ ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

*1 Only T can be used.

❺ ■Control data

| Operand: (d) | | | |
|---|---|---|---|
| Device | Description | Setting range | Set by |
| +0 | Off delay timer output:<br>Turns on at the rising edge of the command of the STMR instruction and turns off when the time specified by (s2) elapses after the falling edge. | — | System |
| +1 | One-shot timer output after turning off:<br>Turns on at the falling edge of the command of the STMR instruction and turns off when the time specified by (s2) elapses. | — | System |
| +2 | One-shot timer output after turning on<br>Turns on at the rising edge of the command of the STMR instruction and turns off when the command of the STMR instruction is turned off or when it [illegible] elapses. | — | System |

❻ Processing details

• This instruction uses the four devices from the device specified by (d) to perform four types of timer output.




❼ Precautions

• The timer number specified in this instruction cannot be used in other general circuits (such as OUT instruction). If the timer number is used in other general circuits, the timer malfunctions.
• The timer specified by (s1) starts counting as a 100 ms timer on the rising edge of the command contact.
• Four devices are occupied from a device specified in (d). Make sure that such devices are not used in other controls for the machine.
• If the command contact is turned off, (d), (d)+1, and (d)+3 turn off when the set time elapses. (d)+2 and the timer (s1) are immediately reset.

❽ Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (s2) is outside the following range.<br>1 to 32767 |

7 APPLICATION INSTRUCTION
7.11 Special timer instruction 543

❶Indicates the instruction symbol.

- The instruction symbol with brackets means multiple instructions. For example, "GRY(P)(_U)" means the GRY, GRYP, GRY_U, and GRYP_U instructions.

| Instruction symbol | Description of symbol |
|---|---|
| Instruction symbol with "(P)" | The instruction is executed on the rising edge. |
| Instruction symbol with "(_U)" | The instruction handles 16-bit or 32-bit unsigned binary data. |

- The instruction symbol with "□" means multiple instructions. For example, "LDDT□" means the LDDT=, LDDT<>, LDDT>, LDDT<=, LDDT<, and LDDT>= instructions.

❷Indicates the description format of the ladder diagram, FBD/LD language and ST (structured text) language

Instruction symbols are input in each corresponding place surrounded in a square in the ladder diagram.

❸Indicates the description, setting range, data type, and data type (label) of each operand.

- For the data type, refer to the following.

MELSEC iQ-F FX5 Programming Manual (Program Design)

❹Indicates the applicable devices for each operand. The following table describes the usage classification.

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others *5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X*2, Y*2, M*2, L*2, SM*2, F*2, B*2, SB*2, S*2 | U□¥G□ | T, ST, C, LC | T, ST, C, D*3,W*3,SD*3, SW*3, R*3 | U□¥G□ | Z | LC | LZ | | K, H | E | $ | |
| Applicable devices*1 | X, Y, M, L, SM, F, B, SB, S | U□¥G□ | T*4, ST*4, C*4, LC*4 | T, ST, C, D, W, SD, SW, R | U□¥G□ | Z | LC | LZ | @□<br>@□.□ | K, H | E | $ | P, I, U, N |

*1 For the description of each device, refer to the following.
MELSEC iQ-F FX5 User's Manual (Application)

*2 "○" is described in positions where bit devices or nibble specification of bit devices is available.

*3 "○" is described in positions where word device or bit specification of word device is available.

*4 When T, ST, C, and LC are used with an instruction other than the following instructions, they can be used only as word data. They cannot be used as bit data.
[Instruction which can be used as bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, BKRST, MOVB(P), CMLB(P)

*5 Devices which can be set are described in the "Others" column.

❺Depending on the instruction, the control data to set the operation of the instruction exists. When the "Set by" column is "User", the value must be specified according to the setting range.

❻Indicates the function details of the instruction. When no details are described, the following programs correspond to "Interrupt program".

- Interrupt program using the interrupt pointer (I)
- Event execution type program which is triggered by an interrupt by the interrupt pointer (I)

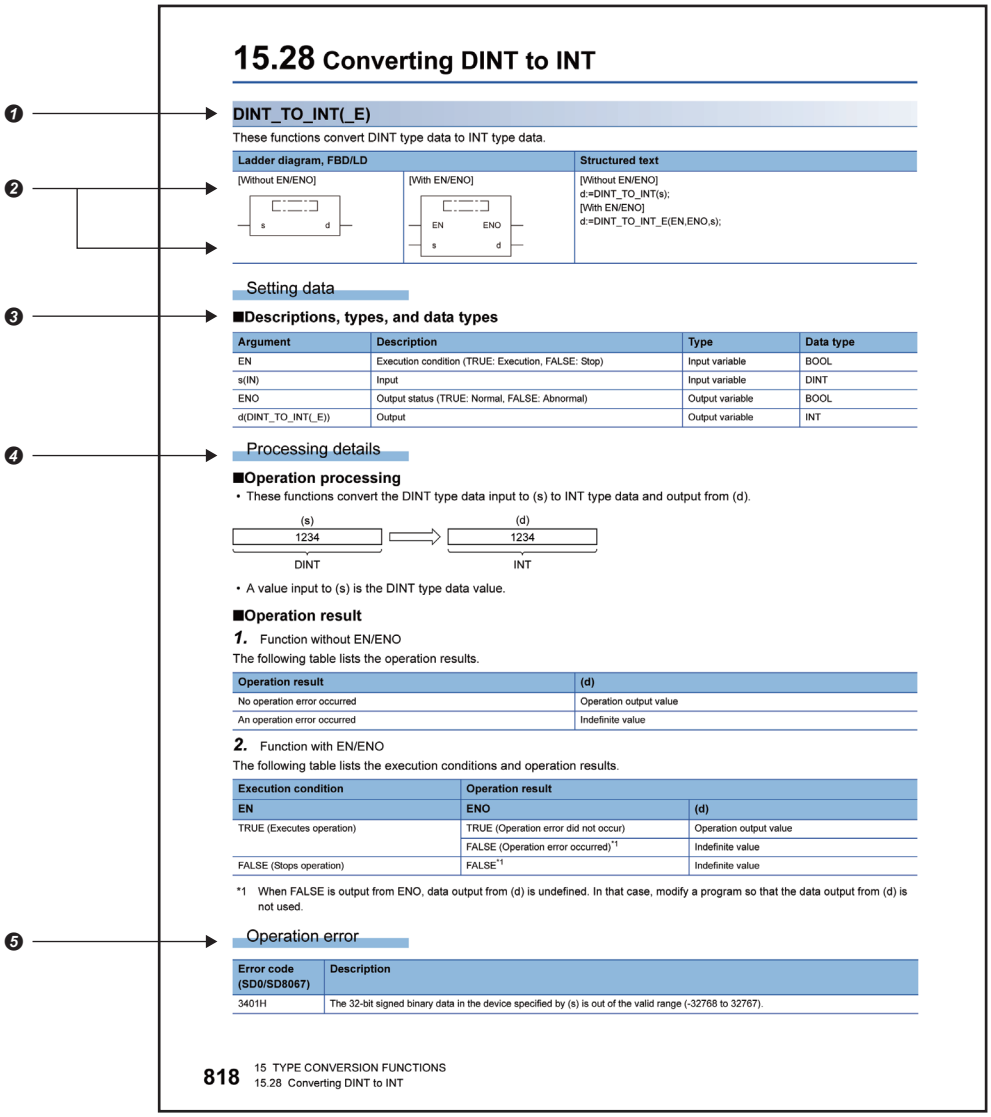
❼Indicates the cautions.

❽Indicates an error code (hexadecimal) which occurs at the execution and the error description when the instruction has a specific operation error.

- A device in which an error code is stored is described in the error code column. When an error code is stored in SD0/SD8067, the error flag (SM0, SM1, SM56, SM8067) turns on.

## How to read PART 5 and PART 6

The contents described in this section are provided only for explaining how to read this manual. Thus, the actual description may differ.



### 15.28 Converting DINT to INT

❶ **DINT_TO_INT(_E)**

These functions convert DINT type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DINT_TO_INT(s);<br>[With EN/ENO]<br>d:=DINT_TO_INT_E(EN,ENO,s); |

❷

❸ Setting data

**■Descriptions, types, and data types**

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_INT(_E)) | Output | Output variable | INT |

❹ Processing details

**■Operation processing**

• These functions convert the DINT type data input to (s) to INT type data and output from (d).

(s) 1234 (DINT) ⇒ (d) 1234 (INT)

• A value input to (s) is the DINT type data value.

**■Operation result**

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

❺ Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The 32-bit signed binary data in the device specified by (s) is out of the valid range (-32768 to 32767). |



❶Indicates function symbols.

When character strings in brackets are added to the end of the function symbol for standard functions and function blocks, the function symbol indicates multiple functions. For example, "DINT_TO_INT(_E)" means "DINT_TO_INT" and "DINT_TO_INT_E".

| Function symbol | Description of symbol |
|---|---|
| Function symbol to which "(_E)" is added. | Indicates that the description format with EN/ENO can be used in the standard function and function block. |

❷Indicates the description format of the ladder diagram, FBD/LD language and ST (structured text) language.

In the square , either of the following symbol should be described.

- Standard function: Function symbol
- Standard function block: Instance name and function block symbol

The sign of return value of the standard function of FBD/LD is not displayed.

❸Indicates the description, type and data type of each argument.

❹Indicates the functions of each standard function or function block.

❺Indicates an error code which occurs at the execution and the error description when the standard function or the function block has a specific operation error.

A device in which an error code is stored is described in the error code column. When an error code is stored in SD0, the error flag SM0 turns on.

# PART 1 OVERVIEW

Part 1 consists of the following chapter.

# 1 OVERVIEW

## 1.1 Instruction Configuration

Many instructions available for CPU module are each divided into the instruction part and device part.
The instruction part and device part are used as follows.

- Instruction part: Indicates the function of the relevant instruction.
- Device part: Indicates the data used for the instruction.

The device part is further classified to source data, destination data, and numerical data.

### Source (s)

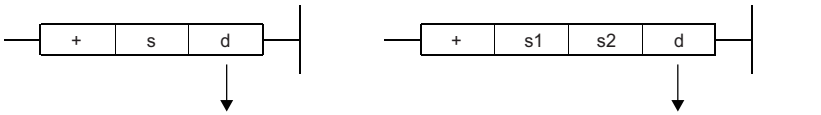
Source is the data used in the operation.
Depending on the label or device specified in each instruction, the source becomes as follows.

| Type | Description |
|---|---|
| Constant | The constant specifies a numerical value used in the operation.<br>It is set during program creation and cannot be changed during program execution. |
| Bit device<br>Word device | The user specifies the device where the data to be used in the operation is stored.<br>Necessary data must be thus stored in the specified device before operation execution.<br>By changing the data to be stored in the specified device during program execution, the data to be used by the instruction can be changed. |

### Destination (d)

Data after operation is stored in the destination area.
However, some instructions require the data to be used in the operation to be stored before the operation.

Ex.
Binary 16-bit data addition instruction

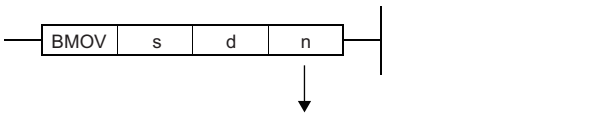



A label or device to store data must be set for the destination.

### Numerical values (n)

In an instruction which uses multiple devices or an instruction which specifies the number of repetitions, data to be processed, and character strings, use numerical values to specify the number of devices, transfers, data, and character strings.

Ex.
Block transfer instruction




A numerical value from 0 to 65535 or 0 to 4294967295 can be set for the size such as the number of devices, transfers, or characters.*1
Note, however, that when the size specification such as the number of devices, transfers, or characters is 0, the relevant instruction results in non-processing.

*1 The setting range varies depending on the instruction. For details, refer to the description of each instruction.

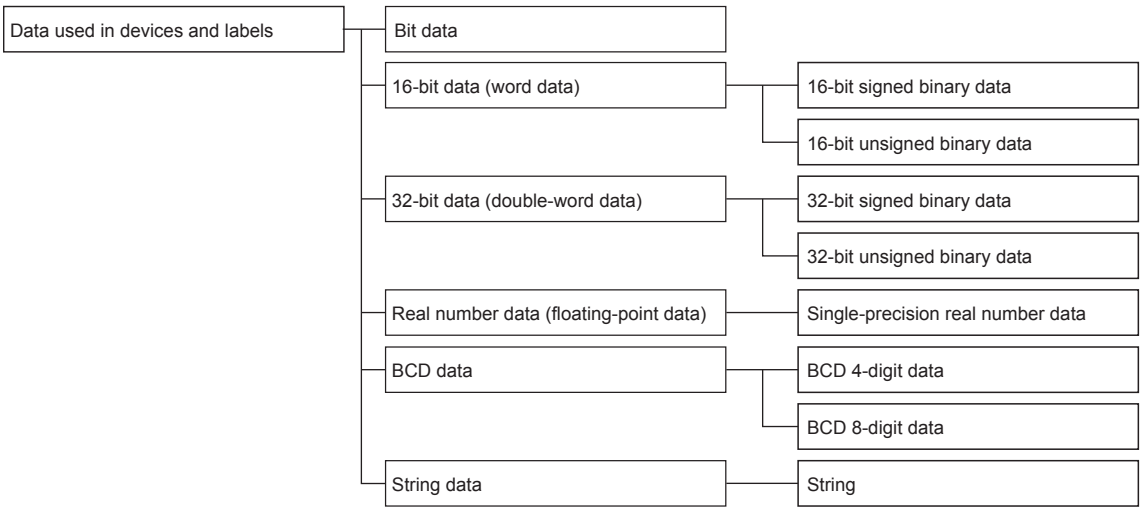
**Point**

Be careful when a large numerical value is used such as for the number of transfers. It delays the scan time.

# 1.2 Data Specification Method



The following table lists the types of data that can be used for instructions in CPU modules.




## Device data

| Data type | Description | Specifiable device/constant*1 |
|---|---|---|
| Bit | Bit data can be handled.<br>☞ Page 26 Bit data | • Bit device<br>• Bit specification of word device |
| Word | Word data can be handled.<br>☞ Page 27 16-bit data (word data) | • Word device<br>• Nibble specification of bit devices (K1 to K4)*2<br>• Decimal constant<br>• Hexadecimal constant |
| 16-bit signed binary | 16-bit data can be handled.<br>The value range varies depending on whether the value is signed or unsigned.<br>☞ Page 27 16-bit data (word data) | |
| 16-bit unsigned binary | | |
| Double word | Double-word data can be handled.<br>☞ Page 29 32-bit data (double word data) | • Word device<br>• Double-word device<br>• Nibble specification of bit devices (K1 to K8)*2<br>• Decimal constant<br>• Hexadecimal constant |
| 32-bit signed binary | Two consecutive sets of 32-bit data or 16-bit data can be handled.<br>The value range varies depending on whether the value is signed or unsigned.<br>☞ Page 29 32-bit data (double word data) | |
| 32-bit unsigned binary | | |
| BCD 4-digit | BCD 4-digit data can be handled.<br>16-bit data is divided by 4 digits and each digit is specified in 0 to 9. | • Word device<br>• Nibble specification of bit devices (K1 to K4)*2<br>• Decimal constant<br>• Hexadecimal constant |
| BCD 8-digit | BCD 8-digit data can be handled.<br>32-bit data is divided by 8 digits and each digit is specified in 0 to 9. | • Word device<br>• Double-word device<br>• Nibble specification of bit devices (K1 to K8)*2<br>• Decimal constant<br>• Hexadecimal constant |
| Single-precision real number | Single-precision real number data (single-precision floating-point data) can be handled.<br>☞ Page 32 Configuration of single-precision real number data | • Word device<br>• Double-word device<br>• Real constant |
| Character string | ASCII code and Shift JIS code character string data can be handled.<br>☞ Page 34 Character string data | • Word device<br>• Character string constant |

*1 A constant can be used in the data specified for the source (s) or numerical data (n) by an instruction.

*2 For the specification method, refer to the detail page of each data type.

## Label data

### ■Primitive data type

| Data type (label) | Specifiable label |
|---|---|
| Bit<br>(BOOL) | • Bit type label<br>• Bit-specified word [unsigned]/bit string [16 bits] type label<br>• Bit-specified word [signed] type label<br>• Timer/retentive timer/long timer/long retentive timer type label contact/coil<br>• Counter/ long counter type label contact/coil |
| Word [unsigned]/bit string [16 bits]<br>(WORD) | • Word [unsigned]/bit string [16 bits] type label<br>• Nibble specified bit type label (K1 to K4)<br>• Current value of timer/retentive timer type label<br>• Current value of counter type label |
| Double word [unsigned]/bit string [32 bits]<br>(DWORD) | • Double word [unsigned]/bit string [32 bits] type label<br>• Nibble specified bit type label (K1 to K8)<br>• Current value of long timer/long retentive timer type label<br>• Current value of long counter type label |
| Word [signed]<br>(INT) | • Word [signed] type label<br>• Nibble specified bit type label (K1 to K4)<br>• Current value of timer/retentive timer type label<br>• Current value of counter type label |
| Double word [signed]<br>(DINT) | • Double word [signed] type label<br>• Nibble specified bit type label (K1 to K8)<br>• Current value of long timer/long retentive timer type label<br>• Current value of long counter type label |
| Single-precision real number<br>(REAL) | • Single-precision real data type label |
| Time<br>(TIME) | • Time type label |
| Character string<br>(STRING) | • Character string type label |
| Timer<br>(TIMER) | • Timer type label |
| Retentive timer<br>(RETENTIVETIMER) | • Retentive timer type label |
| Counter<br>(COUNTER) | • Counter type label |
| Long counter<br>(LCOUNTER) | • Long counter type label |
| Pointer<br>(POINTER) | • Pointer type label |

## ■Generic data type

| Data type (label) | Specifiable label |
|---|---|
| ANY[*1] | Bit, word [signed], double word [signed], word [unsigned]/bit string [16 bits], double word [unsigned]/bit string[32 bits], single-precision real number, hour, character string, structure |
| ANY_BITADDR[*1] | Bit |
| ANY_BOOL | Bit |
| ANY_ELEMENTARY | Bit, word [signed], double word [signed], word [unsigned]/bit string [16 bits], double word [unsigned]/bit string[32 bits], single-precision real number, hour, character string |
| ANY_WORDADDR | Word [signed], double word [signed], word [unsigned]/bit string [16 bits], double word [unsigned]/bit string[32 bits], single-precision real number, hour, character string |
| Any 16-bit data (ANY16) | Word [signed], word [unsigned]/bit string [16 bits] |
| ANY16_S | Word [signed] |
| ANY16_U | Word [unsigned]/bit string [16 bits] |
| Any 32-bit data (ANY32) | Double word [signed], double word [unsigned]/bit string [32 bits], hour |
| ANY32_S | Double word [signed], hour |
| ANY32_U | Double word [unsigned]/bit string [32 bits] |
| ANY_REAL | Single-precision real number |
| ANYREAL_32 | Single-precision real number |
| ANY_STRING | Character string |
| ANYSTRING_SINGLE | Character string |
| ANY_STRUCT[*1] | Structures |
| ANY_DT | Word [signed], word [unsigned]/bit string [16 bits] |
| ANY_TM | Word [signed], word [unsigned]/bit string [16 bits] |
| STRUCT | Structures |
| ANY16_OR_STRING_SINGLE | Word [signed], word [unsigned]/bit string [16 bits], character string |

*1 Can also be used as an array.

## ■Generic data type (array)

For the following generic data type, define the number of array elements.

| Data type (label) | Specifiable label |
|---|---|
| ANYBIT_ARRAY | Bit |
| ANYWORD_ARRAY | Word [signed], double word [signed], word [unsigned]/bit string [16 bits], double word [unsigned]/bit string[32 bits], single-precision real number, hour, character string |
| ANY16_ARRAY | Word [signed], word [unsigned]/bit string [16 bits] |
| ANY16_S_ARRAY | Word [signed] |
| ANY16_U_ARRAY | Word [unsigned]/bit string [16 bits] |
| ANY32_ARRAY | Double word [signed], double word [unsigned]/bit string [32 bits] |
| ANY32_S_ARRAY | Double word [signed] |
| ANY32_U_ARRAY | Double word [unsigned]/bit string [32 bits] |
| ANY_REAL_ARRAY | Single-precision real number |
| ANY_REAL_32_ARRAY | Single-precision real number |
| ANY_STRING_ARRAY | Character string |
| ANY_STRING_SINGLE_ARRAY | Character string |
| STRUCT_ARRAY | Structures |

# Bit data

## Data size and data range

Bit data is handled in increments of bits such as contacts and coils.

| Data name | Data size | Value range |
|---|---|---|
| Bit data | 1 bit | 0, 1 |

## Handling bit data with bit devices and labels

Bit data of one point per point can be handled.

## Handling bit data with bit word devices

By specifying a bit number for a word device, bit data of the specified bit number can be handled.

The notation for bit number specification is as follows.

Word device number . Bit number

A bit number can be specified in hexadecimal in the range from 0 to F.

For example, bit 5 (b5) of D0 is specified as D0.5, and bit 10 (b10) of D0 is specified as D0.A.

The following word devices support bit specification.

| Item | Device |
|---|---|
| Word devices which support bit specification | • Data register (D)<br>• Link register (W)<br>• Link special register (SW)<br>• Special register (SD)<br>• Module access device (U□\G)<br>• File register (R) |

## Handling bit data with word type labels

By specifying a bit number for a word [unsigned]/bit string [16 bits] type label or word [signed] type label, bit data of the specified bit number can be handled.

The notation for bit number specification is as follows.

Label name . Bit number

# 16-bit data (word data)



## Data size and data range

16-bit data includes signed and unsigned 16-bit data.

In signed 16-bit data, a negative number is represented in two's complement.

| Data name | Data size | Value range | |
|---|---|---|---|
| | | Decimal notation | Hexadecimal notation |
| Signed 16-bit data | 16 bits (1 word) | -32768 to 32767 | 0000H to FFFFH |
| Unsigned 16-bit data | | 0 to 65535 | |

## Handling 16-bit data with bit devices

A bit device can be handled as 16-bit data by performing nibble specification.

| Item | Notation | Example |
|---|---|---|
| Bit device | K□ Bit device start number<br>Number of digits: Specify the number within the range from 1 to 4. | K4X10<br>K2M113 |

## Handling 16-bit data with bit type array labels

A bit type array label can be handled as 16-bit data by performing nibble specification.

The following table shows the notation for handling a bit type array label as 16-bit data by nibble specification.

| Item | Notation | Example |
|---|---|---|
| Bit type array label | K□ Label name<br>Number of digits: Specify a number within the range of 1 to 4. | K1L_BOOL |

## Nibble specification range

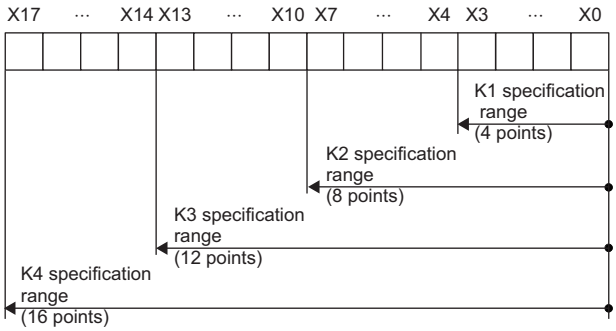
The following table lists the range of 16-bit data for each nibble specification.

| Nibble specification | Decimal notation | Hexadecimal notation |
|---|---|---|
| K1 | 0 to 15 | 0H to FH |
| K2 | 0 to 255 | 00H to FFH |
| K3 | 0 to 4095 | 000H to FFFH |
| K4 | Signed 16-bit data: -32768 to 32767<br>Unsigned 16-bit data: 0 to 65535 | 0000H to FFFFH |

Ex.

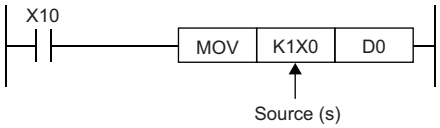
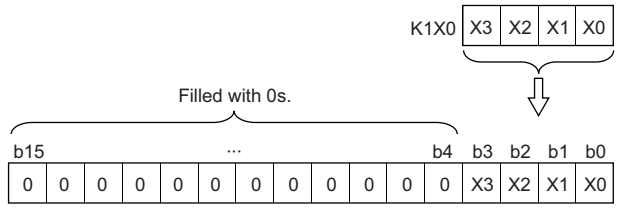
When nibble specification is made for X0, the applicable number of points is as follows.

- K1X0→4 points from X0 to X3
- K2X0→8 points from X0 to X7
- K3X0→12 points from X0 to X13
- K4X0→16 points from X0 to X17




### ■Specifying a bit device with nibble specification in the source (s)

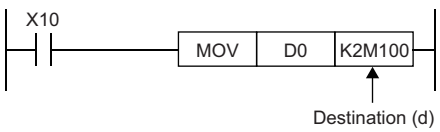
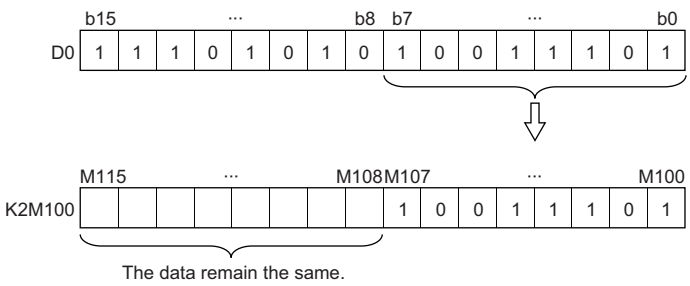
When a bit device with nibble specification is specified in the source of an instruction, 0 is stored in the bits, which follow the bit for which nibble specification is made in the source, in the word device of the destination.

| Ladder example | Processing |
|---|---|
| • 16-bit data instruction<br>  |  |

### ■Specifying a bit device with nibble specification in the destination (d)

When a nibble specification is made in the destination of an instruction, the number of points by the nibble specification is applicable in the destination.

The bit devices after the number of points specified by nibble remain unchanged.

| Ladder example | Processing |
|---|---|
| • When the source data is a word device<br>  |  |

## Handling 16-bit data with word devices/labels

### ■Word device

One point of word device can handle 16-bit data.

### ■Word type label

One point of word type label can handle 16-bit data.

# 32-bit data (double word data)

## Data size and data range

32-bit data includes signed and unsigned 32-bit data.

In signed 32-bit data, a negative number is represented in two's complement.

| Data name | Data size | Value range | |
|---|---|---|---|
| | | Decimal notation | Hexadecimal notation |
| Signed 32-bit data | 32 bits (2 word) | -2147483648 to 2147483647 | 00000000H to FFFFFFFFH |
| Unsigned 32-bit data | | 0 to 4294967295 | |

## Handling 32-bit data with bit devices

A bit device can be handled as 32-bit data by performing nibble specification.

| Item | Notation | Example |
|---|---|---|
| Bit device | K□ Bit device start number<br>Number of digits: Specify the number within the range from 1 to 8. | K8X80<br>K6B018 |

## Handling 32-bit data with bit type array labels

A bit type array label can be handled as 32-bit data by performing nibble specification.

The following table shows the notation for handling a bit type array label as 32-bit data by nibble specification.

| Item | Notation | Example |
|---|---|---|
| Bit type array label | K□ Label name<br>Number of digits: Specify a number within the range of 1 to 8. | K8L_BOOL |

## Nibble specification range

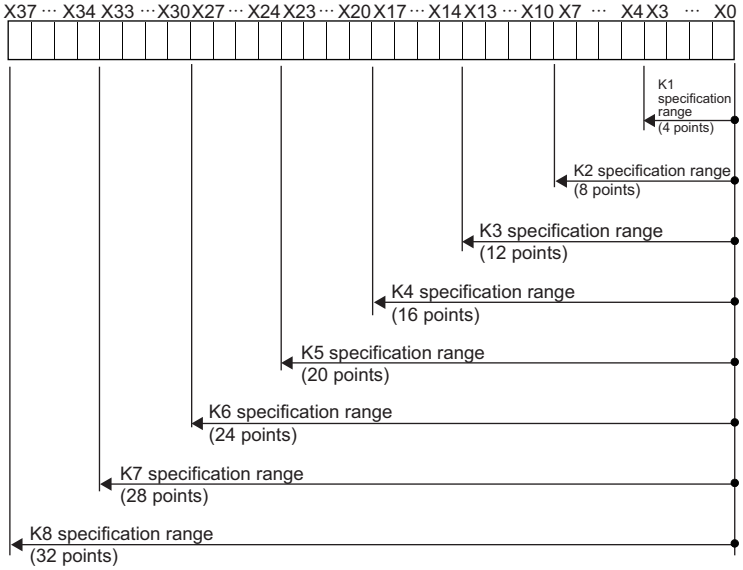
The following table lists the range of 32-bit data for each nibble specification.

| Nibble specification | Decimal notation | Hexadecimal notation |
|---|---|---|
| K1 | 0 to 15 | 0H to FH |
| K2 | 0 to 255 | 00H to FFH |
| K3 | 0 to 4095 | 000H to FFFH |
| K4 | 0 to 65535 | 0000H to FFFFH |
| K5 | 0 to 1048575 | 00000H to FFFFFH |
| K6 | 0 to 16777215 | 000000H to FFFFFFH |
| K7 | 0 to 268435455 | 0000000H to FFFFFFFH |
| K8 | Signed 32-bit data: -2147483648 to 2147483647<br>Unsigned 32-bit data: 0 to 4294967295 | 00000000H to FFFFFFFFH |

**Ex.**

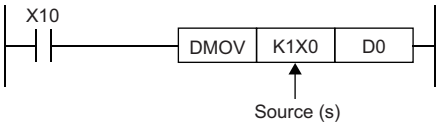
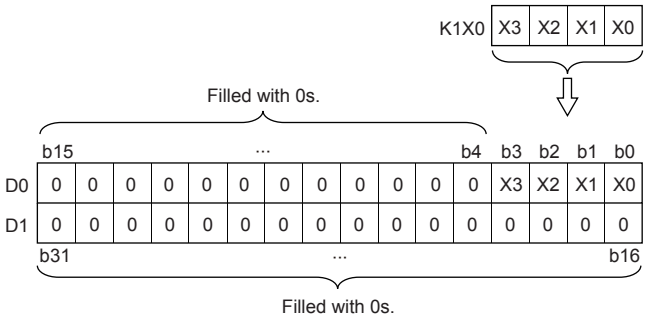
When nibble specification is made for X0, the applicable number of points is as follows.

- K1X0→4 points from X0 to X3
- K2X0→8 points from X0 to X7
- K3X0→12 points from X0 to X13
- K4X0→16 points from X0 to X17
- K5X0→20 points from X0 to X23
- K6X0→24 points from X0 to X27
- K7X0→28 points from X0 to X33
- K8X0→32 points from X0 to X37



### ■Specifying a bit device with nibble specification in the source (s)

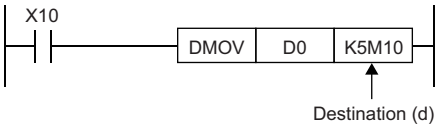
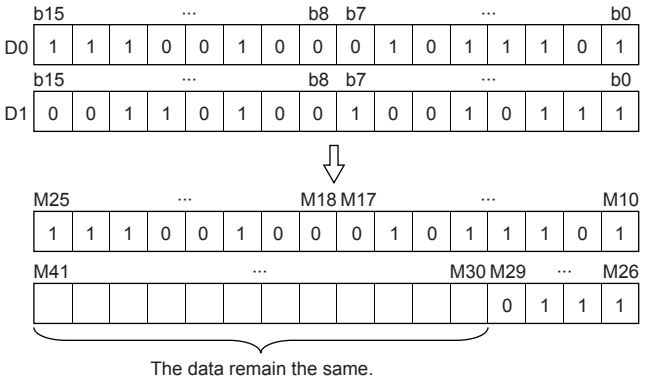
When a bit device with nibble specification is specified in the source of an instruction, 0 is stored in the bits, which follow the bit for which nibble specification is made in the source, in the word device of the destination.

| Ladder example | Processing |
|---|---|
| • 32-bit data instruction<br>  |  |

### ■Specifying a bit device with nibble specification in the destination (d)

When a nibble specification is made in the destination of an instruction, the number of points by the nibble specification is applicable in the destination.

The bit devices after the number of points specified by nibble remain unchanged.

| Ladder example | Processing |
|---|---|
| • When the source data is a word device<br>  |  |

## Handling 32-bit data with word devices/labels

### ■Word device

Two points of word device can handle 32-bit data.

Note, however, that one point of the following devices can handle 32-bit data.

- Long counter (LC)
- Long index register (LZ)

### ■Double word type label

One point of double word device can handle 32-bit data.

# Real number data (floating-point data)

## Data size and data range

Real number data includes single-precision 32-bit real number data.

Real number data can be stored only in devices other than bit devices or in single-precision real data type labels.

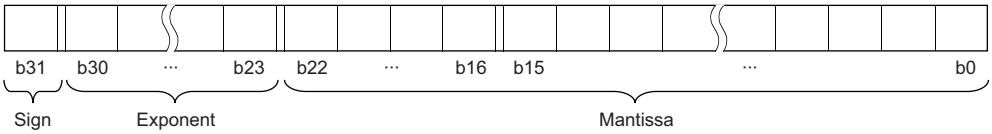
| Data name | | Data size | Value range |
|---|---|---|---|
| Single-precision real number data (single-precision floating-point data) | Positive number | 32 bits (2 word) | $2^{-126} \leq$ real number $< 2^{128}$ |
| | Zero | | 0 |
| | Negative number | | $-2^{128} <$ real number $\leq -2^{-126}$ |

## Configuration of single-precision real number data

Single-precision real number data consists of a sign, mantissa, and exponent, and is expressed as shown below.

Sign 1. Mantissa × 2 Exponent

The following figure shows the bit configuration of the internal expression of single-precision real number data and the meaning of each part.




### ■Sign (1 bit)

This bit represents the positive or negative sign of a numerical value. "0" indicates a positive number or 0. "1" Indicates a negative number.

### ■Mantissa (23 bits)

A mantissa means XXXXX··· of 1.XXXXX···$\times 2^N$ representing a single-precision real number in binary.

### ■Exponent (8 bits)

An exponent means N of 1.XXXXX···$\times 2^N$ representing a single-precision real number in binary. The following table shows the relationships between the exponent value and N of a single-precision real number.

| Exponent (b24 to b30) | FFH | FEH | FDH | ··· | 81H | 80H | 7FH | 7EH | ··· | 02H | 01H | 00H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | Not used | 127 | 126 | ··· | 2 | 1 | 0 | -1 | ··· | -125 | -126 | Not used |

## Precautions

### ■When setting an input value of single-precision real number from the engineering tool

The number of significant digits is about 7 because the engineering tool processes single precision real number data in 32-bit single precision.

When the input value of single-precision real number data exceeds 7 digits, the 8th digit is rounded off.

Therefore, if the rounded-off value goes out of the range from -2147483648 to 2147483647, it will not be an intended value.

Ex.

When "2147483647" is set as an input value, it is handled as "2147484000" because 8th digit "6" is rounded off.

Ex.

When "E1.1754943562" is set as an input value, it is handled as "E1.175494" because 8th digit "3" is rounded off.

**Point**

The monitor function of the engineering tool can monitor real number data of CPU modules.
To represent "0" in real number data, set all numbers in each of the following range to 0.

- Single-precision real number data: b0 to b31

The setting range of real number data is as follows.

- Single precision real number data: $-2^{128}$<single precision real number data]≤$-2^{-126}$, 0, $2^{-126}$≤[single precision real number data]<$2^{128}$

Do not specify "-0" (only the most significant bit is 1) in real number data. Performing a real number operation using -0 results in an operation error.

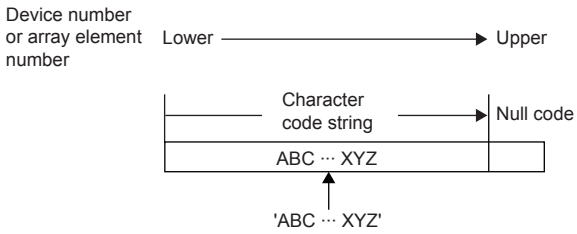
# Character string data

## Format of character string data

The following table lists the types of character string data, each of which ends with a NULL code to be handled as a character string.

| Type | Character code | Last character |
|---|---|---|
| Character string | ASCII code | NULL(00H) |

Character string data is stored in devices or an array in ascending order of device numbers or array element numbers.




## Data range

The following table summarizes the ranges of character string data.

| Type | Maximum number of character strings*1 | Maximum number of character strings that can be handled in the program |
|---|---|---|
| Character string | 255 single-byte characters (excluding the last NULL character) | 16383 characters (excluding the last NULL character) |

*1 When specifying a character string in the program, enclose it in single quotes (').

## Number of words required for storing data

Character string data can be stored in word devices.

The following table lists the numbers of words required for storing character string data.

| Number of character string bytes | Number of words required for storing character strings |
|---|---|
| 0 byte | 1 [word] |
| Odd number of bytes | (Number of character string bytes+1)÷ 2 [words] |
| Even number of bytes | (Number of character string bytes÷2) +1 [words] |

## Character string data storage location

An image of the character string data storage location is shown below.

### ■Character strings

In each character string storage image, "NULL" indicates a NULL code (00H).

| Character string to be stored | Image of storing character string data from D0 | Image of storing character string data from word type label array arrayA[0] |
|---|---|---|
| ' ' (null character string) | D0: NULL \| NULL | arrayA[0]: NULL \| NULL |
| 'ABC' | D0: B \| A<br>D1: NULL \| C | arrayA[0]: B \| A<br>arrayA[1]: NULL \| C |
| 'ABCD' | D0: B \| A<br>D1: D \| C<br>D2: NULL \| NULL | arrayA[0]: B \| A<br>arrayA[1]: D \| C<br>arrayA[2]: NULL \| NULL |

# 1.3 Execution Condition



## Types of execution conditions

The following are the five types of execution conditions of the instructions and functions of CPU module.

### ■On

An instruction is executed during on. It is executed only while the precondition of the instruction is on. When the precondition is off, the instruction is not executed.

### ■Rising edge

An instruction is executed one time when turned on. It is executed only once on the rising edge (off to on) of the precondition of the instruction and is no longer executed later even when the condition turns on.

### ■Off

An instruction is executed during off. It is executed only while the precondition of the instruction is off. When the precondition is on, the instruction is not executed.

### ■Falling edge

An instruction is executed one time when turned off. It is executed only once on the falling edge (on to off) of the precondition of the instruction and is no longer executed later even when the condition turns off.

### ■Always

An instruction is always executed regardless of whether the precondition of the instruction is on or off. When the precondition is off, the instruction performs off processing.

## Execution condition of each instruction

The execution condition varies depending on the instruction. The following table lists the execution conditions of individual instructions.

| Execution condition | Applicable instruction |
|---|---|
| On | All instructions except for the following |
| Rising edge | • Instruction followed by symbol (P)<br>• PLS |
| Off | - |
| Falling edge | PLF |
| Always | LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, ANB, ORB, MPS, MRD, MPP, INV, MEP, MEF, OUT, OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST, OUTHS ST, OUT C, OUT LC, MC, MCR, FEND, END, LD□, AND□, OR□, LD□_U, AND□_U, OR□_U, LDD□, ANDD□, ORD□, LDD□_U, ANDD□_U, ORD□_U, JMP, DI, EI, IMASK, SIMASK, IRET, FOR, NEXT, RET, LD$□, AND$□, OR$□, LDE□, ANDE□, ORE□, STMR, LDDT□, ANDDT□, ORDT□, LDTM□, ANDTM□, ORTM□ |

# 2 PRECAUTIONS ON PROGRAMMING

## 2.1 Errors Common to Instructions

The following table lists the conditions under which an error occurs when the instruction is executed.

| Error content*1 | Error code (SD0/SD8067) |
|---|---|
| An I/O number which corresponds to no module is specified. | 2801H |
| • An I/O number which is out of range (0 to 1777(Octal number)) is specified.<br>• The device or label specified by the instruction exceeds the available range. | 2820H |
| The range of the buffer memory of the module specified by the instruction is exceeded. | 2823H |

*1 For a contact instruction, an error is not detected but the operation result becomes no continuity.

## 2.2 Checking the Ranges of Instruction Runtime Devices and Labels

### Checking the ranges of devices and labels

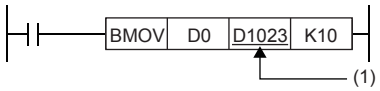
When a device or label is specified in an instruction, range check is performed. If a range exceeding that of the relevant device or label is specified, an error occurs.

The same applies when a label assigned to a device is specified in an instruction in the program.

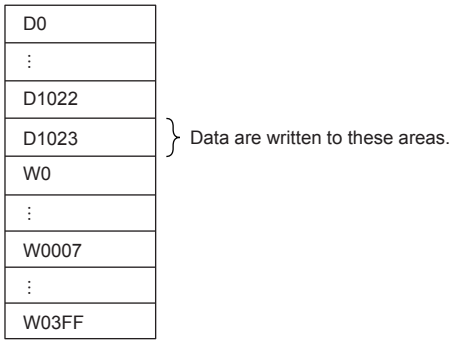
Create such a program that the operation result falls within the range of the relevant device or label.

Ex.

When a global device is specified




Device assignment image in the device/label memory




(1) The transfer destination is in the range corresponding to D1023 to D1032. Because D1024 to D1032 do not exist, the data are written only to D1023.

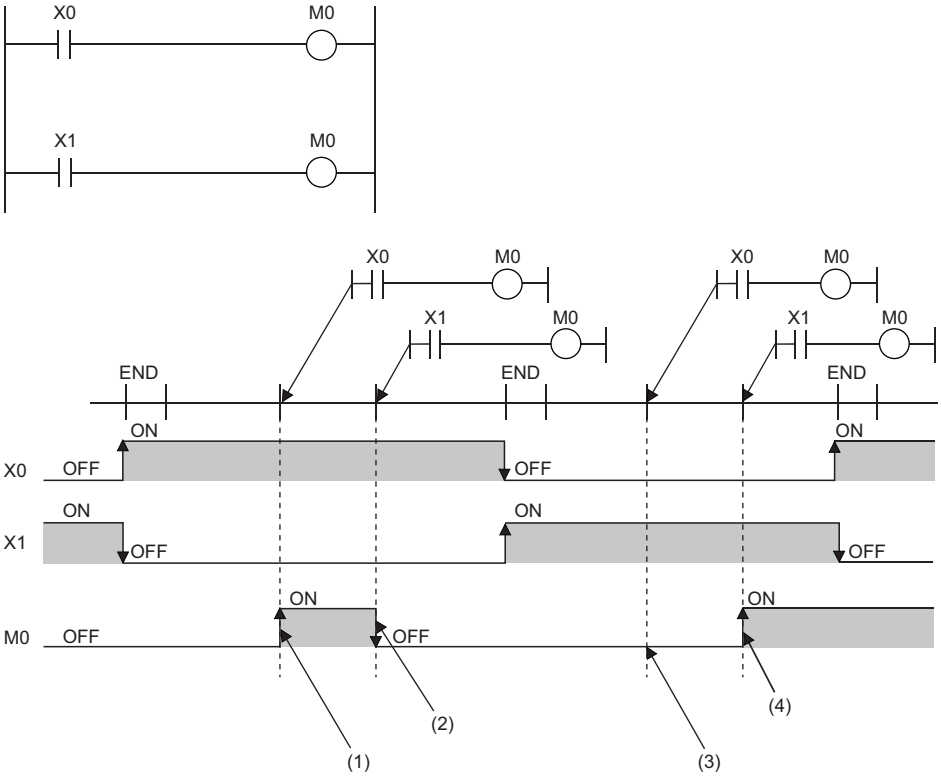
# 2.3 Operations Arising when the OUT, SET/RST, and PLS/PLF Instructions of the Same Device are Used



If two or more OUT, SET/RST, and PLS/PLF instructions are executed using the same device during one scan, they operate as described in this section.

## For OUT instructions of the same device

More than one OUT instruction of the same device must not be issued during one scan.
Otherwise, the specified device turns on or off, depending on the operation result up to each OUT instruction while it is in execution.
In this case, the device may turn on/off during one scan because the on/off state of the specified device is determined during execution of each OUT instruction.
The following figure shows the behavior arising when a circuit turning on/off the same internal relay (M0) is created with input X0 and X1.




(1) Since X0 is on, M0 turns on.
(2) Since X1 is off, M0 turns off.
(3) Since X1 is off, M0 remains off.
(4) Since X1 is on, M0 turns on.

If output (Y) is specified using an OUT instruction, the on/off state of the last OUT instruction executed during the one scan will be output.

## If SET/RST instructions of the same device are used

### ■For SET instructions

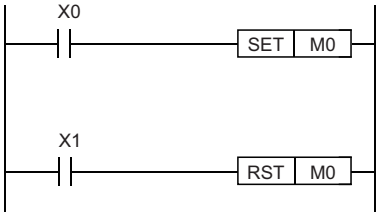
The SET instruction turns on the specified device if the execution command is on, and causes no operation if it is off.
Thus, if two or more SET instructions of the same device are executed during one scan, the specified device turns on even if one execution command is on.

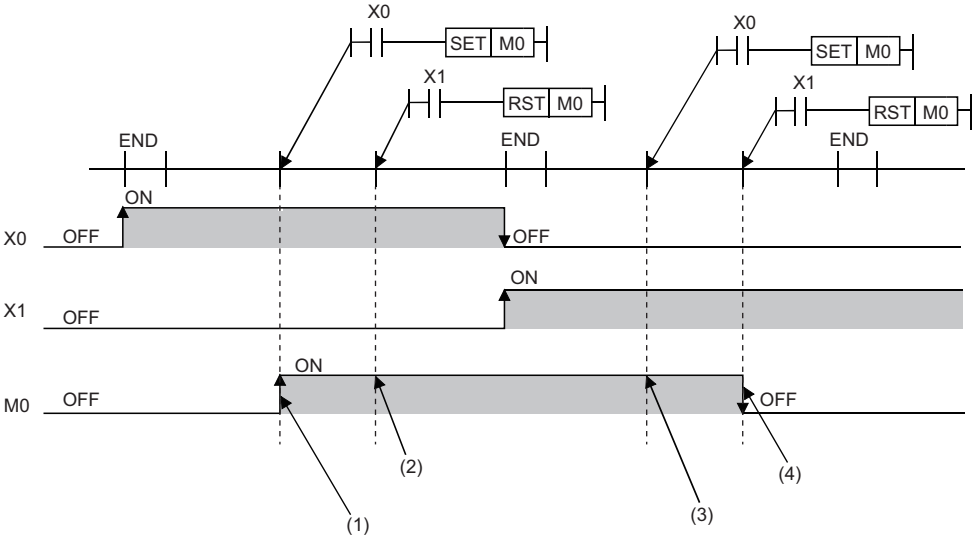
### ■For RST instructions

The RST instruction turns on the specified device if the execution command is off, and causes no operation if it is off.
Thus, if two or more RST instructions of the same device are executed during one scan, the specified device turns on even if one execution command is off.

### ■If the SET and RST instructions of the same device exist in one scan

If the SET and RST instructions of the same device exist in one scan, the SET instruction turns on the specified device if the execution command is on, and turns off the specified device if it is on.
If both the SET and RST instructions are off, the on/off state of the specified device will be unchanged.



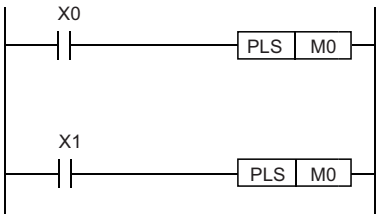




(1) Since X0 is on, M0 turns on.
(2) Since X1 is off, M0 remains on. (The RST instruction results in non-processing.)
(3) Since X0 is off, M0 remains on. (The SET instruction results in non-processing.)
(4) Since X1 is on, M0 turns off.

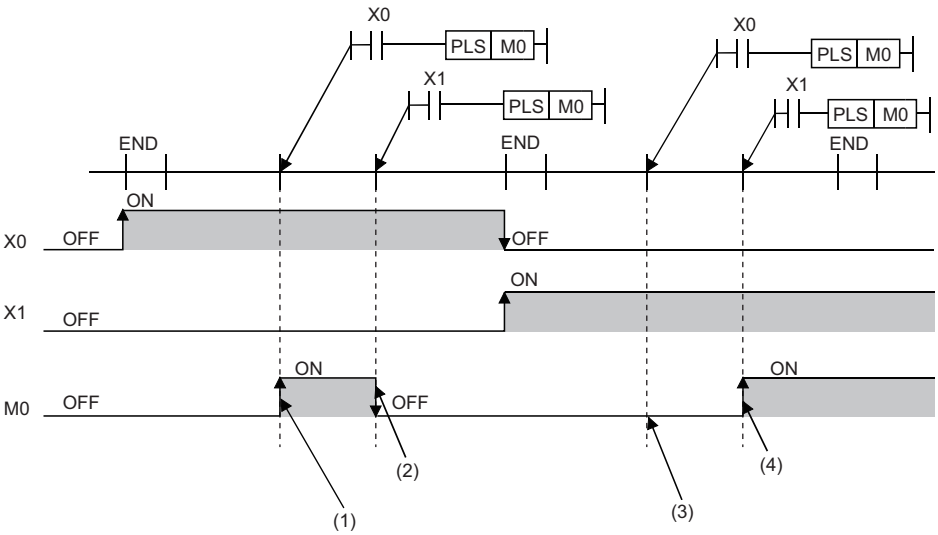
If output (Y) is specified using a SET/RST instruction, the on/off state of the last SET/RST instruction executed during the one scan will be output.

## If PLS instructions of the same device are used

The PLS instruction turns on the specified device when the execution command specifies an off-to-on change. The specified device is turned off unless the execution command specifies an off-to-on change (i.e. off to off, on to on, on to off).
Thus, if two or more PLS instructions of the same device are issued during one scan, the specified device is turned on when the execution command of each PLS instruction specifies an off-to-on change. The specified device is turned off unless the execution command specifies an off-to-on change.
Thus, if two or more PLS instructions are issued during one scan, the device turned on by a PLS instruction may not turn on for one scan.

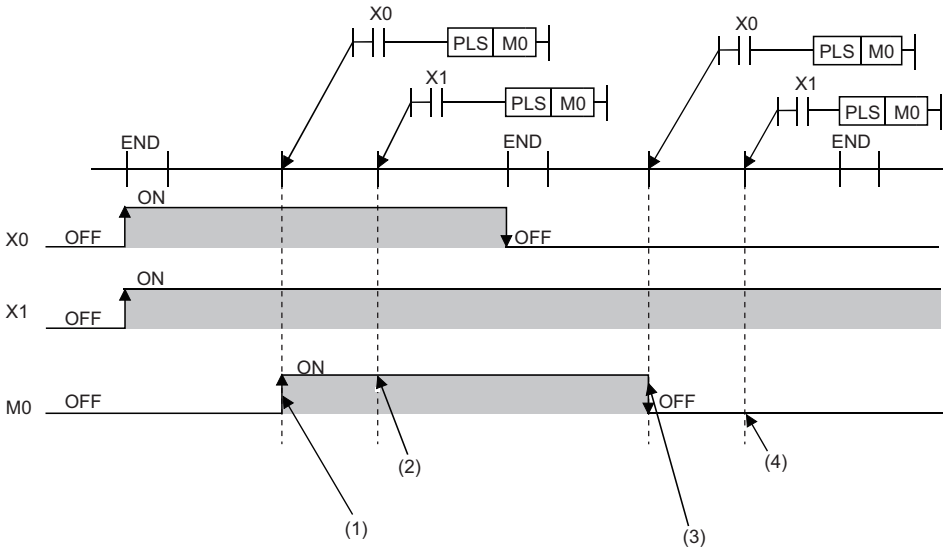



• If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)




(1) Since X0 turns on, M0 turns on.
(2) Since X1 is other than turning on, M0 turns off.
(3) Since X0 is other than turning on, M0 remains off.
(4) Since X1 turns on, M0 turns on.

• If the off-to-on changes of X0 and X1 are at the same timing

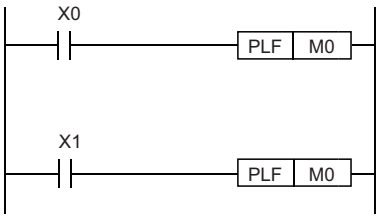



(1) Since X0 turns on, M0 turns on.
(2) Since X1 turns on, M0 remains on.
(3) Since X0 is other than turning on, M0 turns off.
(4) Since X1 is other than turning on, M0 remains off.

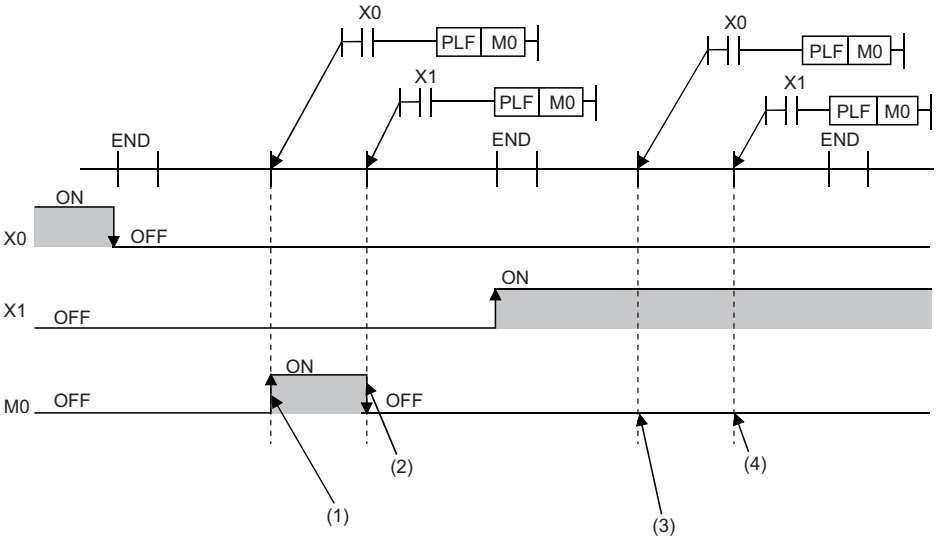
If output (Y) is specified using a PLS instruction, the on/off state of the last PLS instruction executed during the one scan will be output.

## If PLF instructions of the same device are used

The PLF instruction turns on the specified device when the execution command specifies an off-to-on change. The specified device is turned off unless the execution command specifies an on-to-off change (i.e. off to off, off to on, on to on).
Thus, if two or more PLF instructions of the same device are issued during one scan, the specified device is turned on when the execution command of each PLF instruction specifies an on-to-off change. The specified device is turned off unless the execution command specifies an on-to-off change.
Thus, if two or more PLF instructions are issued during one scan, the device turned on by a PLF instruction may not turn on for one scan.

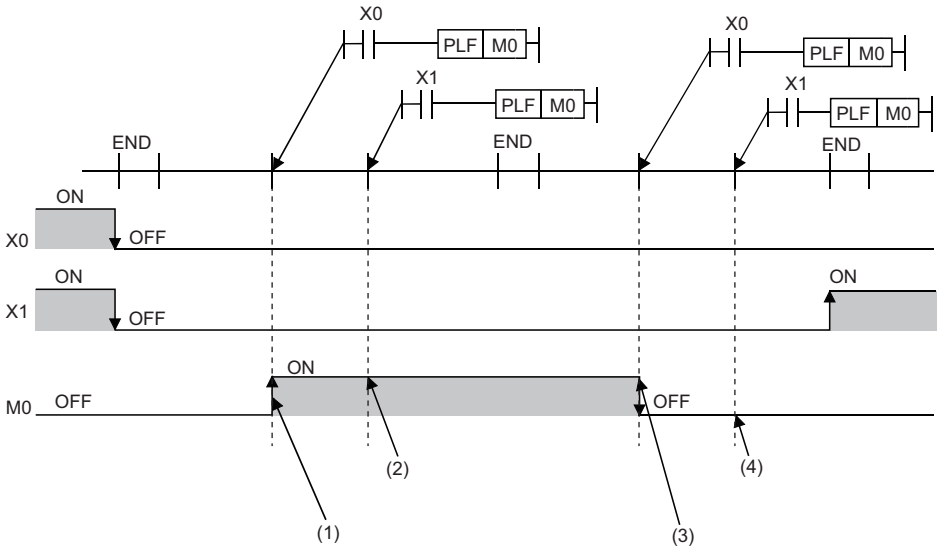



• If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)




(1) Since X0 turns off, M0 turns on.
(2) Since X1 is other than turning off, M0 turns off.
(3) Since X0 is other than turning off, M0 remains off.
(4) Since X1 is other than turning off, M0 remains off.

• If the on-to-off changes of X0 and X1 are at the same timing



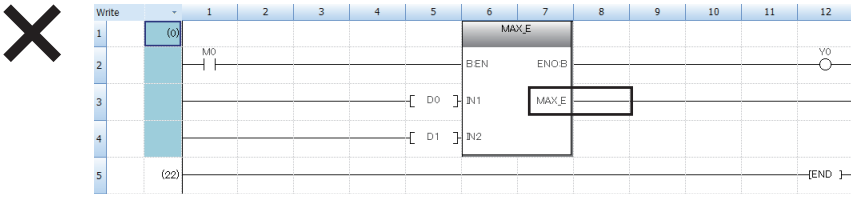

(1) Since X0 turns off, M0 turns on.
(2) Since X1 turns off, M0 remains on.
(3) Since X0 is other than turning off, M0 turns off.
(4) Since X1 is other than turning off, M0 remains off.

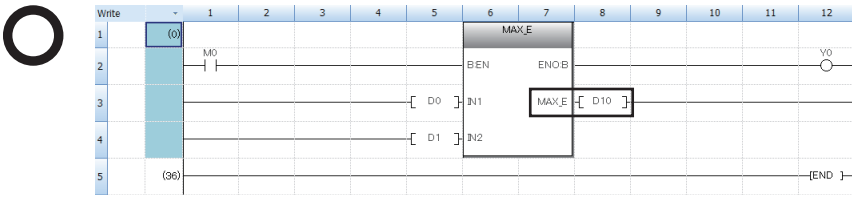
If output (Y) is specified using a PLF instruction, the on/off state of the last PLF instruction executed during the one scan will be output.

# 2.4 Standard Functions/Function Blocks Return Values

When standard functions/function blocks are used, always set a device or label for the output variable, and make sure to receive the return value.

If the output variable is not set, an error could occur after writing to the PLC.



# PART 2 INSTRUCTION/ FUNCTION LIST

This part consists of the following chapters.

# 3 CPU MODULE INSTRUCTION

## 3.1 Sequence Instruction

### Contact instruction

#### ■Operation start, series connection, parallel connection

| Instruction symbol | Description | Reference |
|---|---|---|
| LD | Starts logical operation (Starts NO contact logical operation) | Page 100 |
| LDI | Starts logical NOT operation (Starts NC contact logical operation) | |
| AND | Logical AND (NO contact series connection) | |
| ANI | Logical NAND (NC contact series connection) | |
| OR | Logical OR (NO contact parallel connection) | |
| ORI | Logical NOR (NC contact parallel connection) | |

#### ■Pulse operation start, pulse series connection, pulse parallel connection

| Instruction symbol | Description | Reference |
|---|---|---|
| LDP | Starts rising edge pulse operation | Page 102 |
| LDF | Starts falling edge pulse operation | |
| ANDP | Rising edge pulse series connection | |
| ANDF | Falling edge pulse series connection | |
| ORP | Rising edge pulse parallel connection | |
| ORF | Falling edge pulse parallel connection | |

#### ■Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection

| Instruction symbol | Description | Reference |
|---|---|---|
| LDPI | Starts rising edge pulse NOT operation | Page 104 |
| LDFI | Starts falling edge pulse NOT operation | |
| ANDPI | Rising edge pulse NOT series connection | |
| ANDFI | Falling edge pulse NOT series connection | |
| ORPI | Rising edge pulse NOT parallel connection | |
| ORFI | Falling edge pulse NOT parallel connection | |

### Association instruction

#### ■Ladder block series/parallel connection

| Instruction symbol | Description | Reference |
|---|---|---|
| ANB | AND between logical blocks (series connection between logical blocks) | Page 106 |
| ORB | OR between logical blocks (parallel connection between logical blocks) | |

#### ■Storing/reading/clearing the operation result

| Instruction symbol | Description | Reference |
|---|---|---|
| MPS | Stores the operation result | Page 107 |
| MRD | Reads the operation result stored by MPS | |
| MPP | Reads and resets of the operation result stored by MPS | |

#### ■Inverting the operation result

| Instruction symbol | Description | Reference |
|---|---|---|
| INV | Inversion of the operation result | Page 108 |

### ■Converting the operation result into a pulse

| Instruction symbol | Description | Reference |
|---|---|---|
| MEP | Conversion of operation result to rising edge pulse | Page 109 |
| MEF | Conversion of operation result to falling edge pulse | |

## Output instruction

### ■Out (excluding the timer, counter and annunciator)

| Instruction symbol | Description | Reference |
|---|---|---|
| OUT | Device output | Page 110 |

### ■Timer (low-speed, high-speed, low-speed retentive, high-speed retentive)

| Instruction symbol | Description | Reference |
|---|---|---|
| OUT T | Low-speed timer | Page 111 |
| OUTH T | Timer | |
| OUTHS T | High-speed timer | |
| OUT ST | Low-speed retentive timer | |
| OUTH ST | Retentive timer | |
| OUTHS ST | High-speed retentive timer | |

### ■Counter, long counter

| Instruction symbol | Description | Reference |
|---|---|---|
| OUT C | Counter | Page 113 |
| OUT LC | Long counter | Page 114 |

### ■Annunciator

| Instruction symbol | Description | Reference |
|---|---|---|
| OUT F | Annunciator | Page 115 |

### ■Setting devices (excluding annunciator)

| Instruction symbol | Description | Reference |
|---|---|---|
| SET | Sets devices | Page 116 |

### ■Resetting devices (excluding annunciator)

| Instruction symbol | Description | Reference |
|---|---|---|
| RST | Resets devices | Page 118 |

### ■Setting/resetting annunciator

| Instruction symbol | Description | Reference |
|---|---|---|
| SET F | Sets annunciator | Page 120 |
| RST F | Resets annunciator | Page 122 |
| ANS | Sets annunciator (with evaluation time) | Page 124 |
| ANR | Resets annunciator (smallest number reset) | Page 125 |
| ANRP | | |

### ■Rising/falling edge output

| Instruction symbol | Description | Reference |
|---|---|---|
| PLS | Generates a pulse for 1 cycle of a program at the rising edge of the input signal. | Page 126 |
| PLF | Generates a pulse for 1 cycle of a program at the falling edge of the input signal. | Page 128 |

### ■Inverting the bit device output

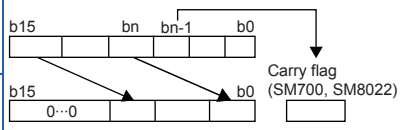
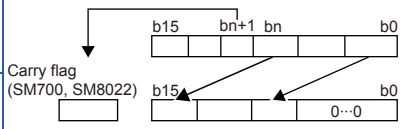
| Instruction symbol | Description | Reference |
|---|---|---|
| FF | Inversion of device output | Page 130 |
| ALT | | Page 131 |
| ALTP | | |

## Shift instruction

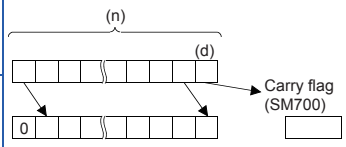
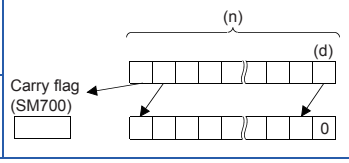
### ■Shifting bit devices

| Instruction symbol | Description | Reference |
|---|---|---|
| SFT | 1 bit shift of the device | Page 132 |
| SFTP | | |

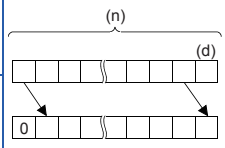
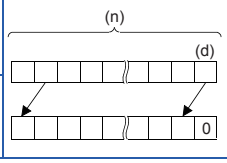
### ■Shifting 16-bit data to the right/left by n bit (s)

| Instruction symbol | Description | Reference |
|---|---|---|
| SFR |  | Page 134 |
| SFRP | | |
| SFL |  | Page 136 |
| SFLP | | |

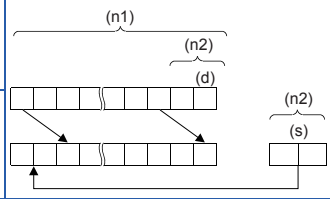
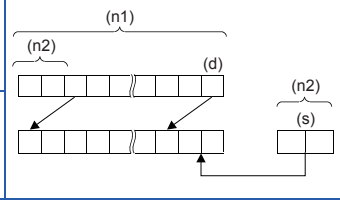
### ■Shifting n-bit data to the right/left by 1 bit

| Instruction symbol | Description | Reference |
|---|---|---|
| BSFR |  | Page 138 |
| BSFRP | | |
| BSFL |  | Page 139 |
| BSFLP | | |

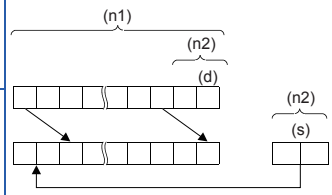
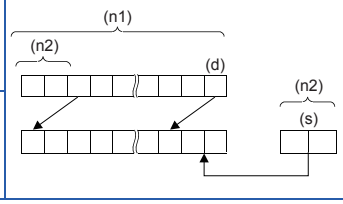
### ■Shifting n-word data to the right/left by 1 word

| Instruction symbol | Description | Reference |
|---|---|---|
| DSFR |  | Page 140 |
| DSFRP | | |
| DSFL |  | Page 141 |
| DSFLP | | |

### ■Shifting n-bit data to the right/left by n bit (s)

| Instruction symbol | Description | Reference |
|---|---|---|
| SFTR<br>SFTRP |  | Page 142 |
| SFTL<br>SFTLP |  | Page 144 |

### ■Shifting n-word data to the right/left by n word (s)

| Instruction symbol | Description | Reference |
|---|---|---|
| WSFR<br>WSFRP |  | Page 146 |
| WSFL<br>WSFLP |  | Page 148 |

## Master control instruction

### ■Setting/resetting the master control

| Instruction symbol | Description | Reference |
|---|---|---|
| MC | Starts master control | Page 150 |
| MCR | Releases master control | |

## Termination instruction

### ■Ending the main routine program

| Instruction symbol | Description | Reference |
|---|---|---|
| FEND | Ends the main routine program | Page 154 |

### ■Ending the sequence program

| Instruction symbol | Description | Reference |
|---|---|---|
| END | Ends the sequence program | Page 155 |

## Stop instruction

### ■Stopping the sequence program

| Instruction symbol | Description | Reference |
|---|---|---|
| STOP | Stops the sequence operation after input conditions are met.<br>Executes the sequence program, upon setting the RUN/STOP/RESET switch to RUN again. | Page 157 |

# 3.2 Basic Instruction

## Comparison operation instruction

### ■Comparing 16-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| LD=, AND=, OR= | (s1)=(s2): Conductive<br>(s1)≠(s2): Non-Conductive | Page 158 |
| LD=_U, AND=_U, OR=_U | | |
| LD<>, AND<>, OR<> | (s1)≠(s2): Conductive<br>(s1)=(s2): Non-Conductive | |
| LD<>_U, AND<>_U, OR<>_U | | |
| LD>, AND>, OR> | (s1)>(s2): Conductive<br>(s1)≤(s2): Non-Conductive | |
| LD>_U, AND>_U, OR>_U | | |
| LD<=, AND<=, OR<= | (s1)≤(s2): Conductive<br>(s1)>(s2): Non-Conductive | |
| LD<=_U, AND<=_U, OR<=_U | | |
| LD<, AND<, OR< | (s1)<(s2): Conductive<br>(s1)≥(s2): Non-Conductive | |
| LD<_U, AND<_U, OR<_U | | |
| LD>=, AND>=, OR>= | (s1)≥(s2): Conductive<br>(s1)<(s2): Non-Conductive | |
| LD>=_U, AND>=_U, OR>=_U | | |

### ■Comparing 32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| LDD=, ANDD=, ORD= | [(s1)+1, (s1)] = [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≠ [(s2)+1, (s2)]: Non-Conductive | Page 160 |
| LDD=_U, ANDD=_U, ORD=_U | | |
| LDD<>, ANDD<>, ORD<> | [(s1)+1, (s1)] ≠ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] = [(s2)+1, (s2)]: Non-Conductive | |
| LDD<>_U, ANDD<>_U, ORD<>_U | | |
| LDD>, ANDD>, ORD> | [(s1)+1, (s1)] > [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≤ [(s2)+1, (s2)]: Non-Conductive | |
| LDD>_U, ANDD>_U, ORD>_U | | |
| LDD<=, ANDD<=, ORD<= | [(s1)+1, (s1)] ≤ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] > [(s2)+1, (s2)]: Non-Conductive | |
| LDD<=_U, ANDD<=_U, ORD<=_U | | |
| LDD<, ANDD<, ORD< | [(s1)+1, (s1)] < [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≥ [(s2)+1, (s2)]: Non-Conductive | |
| LDD<_U, ANDD<_U, ORD<_U | | |
| LDD>=, ANDD>=, ORD>= | [(s1)+1, (s1)] ≥ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] < [(s2)+1, (s2)]: Non-Conductive | |
| LDD>=_U, ANDD>=_U, ORD>=_U | | |

### ■Comparison output 16-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| CMP | (s1)>(s2): (d) is on<br>(s1)=(s2): (d) +1 is on<br>(s1)<(s2): (d) +2 is on | Page 162 |
| CMPP | | |
| CMP_U | | |
| CMPP_U | | |

### ■Comparison output 32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| DCMP | [(s1)+1, (s1)] > [(s2)+1, (s2)]: (d) is on<br>[(s1)+1, (s1)] = [(s2)+1, (s2)]: (d) + 1 is on<br>[(s1)+1, (s1)] < [(s2)+1, (s2)]: (d) + 2 is on | Page 164 |
| DCMPP | | |
| DCMP_U | | |
| DCMPP_U | | |

## ■Comparing 16-bit binary data band

| Instruction symbol | Description | Reference |
|---|---|---|
| ZCP | (s1)>(s3): (d) is on<br>(s1)≤(s3)≤(s2): (d) + 1 is on<br>(s3)>(s2): (d) + 2 is on | Page 166 |
| ZCPP | | |
| ZCP_U | | |
| ZCPP_U | | |

## ■Comparing 32-bit binary data band

| Instruction symbol | Description | Reference |
|---|---|---|
| DZCP | [(s1)+1, (s1)] > [(s3)+1, (s3)]: (d) is on<br>[(s1)+1, (s1)]≤[(s3)+1, (s3)]≤[(s2)+1, (s2)]: (d) + 1 is on<br>[(s3)+1, (s3)]>[(s2)+1, (s2)]: (d) + 2 is on | Page 168 |
| DZCPP | | |
| DZCP_U | | |
| DZCPP_U | | |

## ■Comparing 16-bit binary block data

| Instruction symbol | Description | Reference |
|---|---|---|
| BKCMP=, BKCMP<>, BKCMP>, BKCMP<=, BKCMP<, BKCMP>= | Compares the 16-bit binary data in the device area ((n) points) from (s1) with the 16-bit binary data in the device area ((n) points) from (s2), and stores the result in the device area ((n) points) from (d). | Page 170 |
| BKCMP=P, BKCMP<>P, BKCMP>P, BKCMP<=P, BKCMP<P, BKCMP>=P | | |
| BKCMP=_U, BKCMP<>_U, BKCMP>_U, BKCMP<=_U, BKCMP<_U, BKCMP>=_U | | |
| BKCMP=P_U, BKCMP<>P_U, BKCMP>P_U, BKCMP<=P_U, BKCMP<P_U, BKCMP>=P_U | | |

## ■Comparing 32-bit binary block data

| Instruction symbol | Description | Reference |
|---|---|---|
| DBKCMP=, DBKCMP<>, DBKCMP>, DBKCMP<=, DBKCMP<, DBKCMP>= | Compares the 32-bit binary data in the device area ((n) points) from (s1) with the 32-bit binary data in the device area ((n) points) from (s2), and stores the result in the device area ((n) points) from (d). | Page 172 |
| DBKCMP=P, DBKCMP<>P, DBKCMP>P, DBKCMP<=P, DBKCMP<P, DBKCMP>=P | | |
| DBKCMP=_U, DBKCMP<>_U, DBKCMP>_U, DBKCMP<=_U, DBKCMP<_U, DBKCMP>=_U | | |
| DBKCMP=P_U, DBKCMP<>P_U, DBKCMP>P_U, DBKCMP<=P_U, DBKCMP<P_U, DBKCMP>=P_U | | |

## Arithmetic operation instruction

### ■Adding/subtracting 16-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| + | (d)+(s) → (d) | Page 175 |
| +P | | |
| +_U | | |
| +P_U | | |
| + | (s1)+(s2) → (d) | Page 176 |
| +P | | |
| +_U | | |
| +P_U | | |
| ADD | (s1)+(s2) → (d) | Page 177 |
| ADDP | | |
| ADD_U | | |
| ADDP_U | | |
| - | (d)-(s) → (d) | Page 179 |
| -P | | |
| -_U | | |
| -P_U | | |
| - | (s1)-(s2) → (d) | Page 180 |
| -P | | |
| -_U | | |
| -P_U | | |
| SUB | (s1)-(s2) → (d) | Page 182 |
| SUBP | | |
| SUB_U | | |
| SUBP_U | | |

## ■Adding/subtracting 32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| D+ | [(d)+1, (d)] + [(s)+1, (s)] → [(d)+1, (d)] | Page 184 |
| D+P | | |
| D+_U | | |
| D+P_U | | |
| D+ | [(s1)+1, (s1)] + [(s2)+1, (s2)] → [(d)+1, (d)] | Page 185 |
| D+P | | |
| D+_U | | |
| D+P_U | | |
| DADD | [(s1)+1, (s1)] + [(s2)+1, (s2)] → [(d)+1, (d)] | Page 186 |
| DADDP | | |
| DADD_U | | |
| DADDP_U | | |
| D- | [(d)+1, (d)] - [(s)+1, (s)] → [(d)+1, (d)] | Page 188 |
| D-P | | |
| D-_U | | |
| D-P_U | | |
| D- | [(s1)+1, (s1)] - [(s2)+1, (s2)] → [(d)+1, (d)] | Page 189 |
| D-P | | |
| D-_U | | |
| D-P_U | | |
| DSUB | [(s1)+1, (s1)] - [(s2)+1, (s2)] → [(d)+1, (d)] | Page 191 |
| DSUBP | | |
| DSUB_U | | |
| DSUBP_U | | |

## ■Multiplying/dividing 16-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| * | (s1) × (s2) → [(d)+1, (d)] | Page 193 |
| *P | | |
| *_U | | |
| *P_U | | |
| MUL | (s1) × (s2) → [(d)+1, (d)] | Page 195 |
| MULP | | |
| MUL_U | | |
| MULP_U | | |
| / | (s1) ÷ (s2) → quotient (d), remainder (d)+1 | Page 197 |
| /P | | |
| /_U | | |
| /P_U | | |
| DIV | (s1) ÷ (s2) → quotient (d), remainder (d)+1 | Page 199 |
| DIVP | | |
| DIV_U | | |
| DIVP_U | | |

## ■Multiplying/dividing 32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| D* | [(s1)+1, (s1)] × [(s2)+1, (s2)] → [(d)+3, (d)+2, (d)+1, (d)] | Page 201 |
| D*P | | |
| D*_U | | |
| D*P_U | | |
| DMUL | [(s1)+1, (s1)] × [(s2)+1, (s2)] → [(d)+3, (d)+2, (d)+1, (d)] | Page 203 |
| DMULP | | |
| DMUL_U | | |
| DMULP_U | | |
| D/ | [(s1)+1, (s1)] ÷ [(s2)+1, (s2)] → quotient [(d)+1, (d)], remainder [(d)+3, (d)+2] | Page 205 |
| D/P | | |
| D/_U | | |
| D/P_U | | |
| DDIV | [(s1)+1, (s1)] ÷ [(s2)+1, (s2)] → quotient [(d)+1, (d)], remainder [(d)+3, (d)+2] | Page 207 |
| DDIVP | | |
| DDIV_U | | |
| DDIVP_U | | |

## ■Adding/subtracting BCD 4-digit data

| Instruction symbol | Description | Reference |
|---|---|---|
| B+ | (d) + (s) → (d) | Page 209 |
| B+P | | |
| B+ | (s1) + (s2) → (d) | Page 210 |
| B+P | | |
| B- | (d) - (s) → (d) | Page 211 |
| B-P | | |
| B- | (s1) - (s2) → (d) | Page 212 |
| B-P | | |

## ■Adding/subtracting BCD 8-digit data

| Instruction symbol | Description | Reference |
|---|---|---|
| DB+ | [(d)+1, (d)] + [(s)+1, (s)] → [(d)+1, (d)] | Page 213 |
| DB+P | | |
| DB+ | [(s1)+1, (s1)] + [(s2)+1, (s2)] → [(d)+1, (d)] | Page 214 |
| DB+P | | |
| DB- | [(d)+1, (d)] - [(s)+1, (s)] → [(d)+1, (d)] | Page 215 |
| DB-P | | |
| DB- | [(s1)+1, (s1)] - [(s2)+1, (s2)] → [(d)+1, (d)] | Page 216 |
| DB-P | | |

## ■Multiplying/dividing BCD 4-digit data

| Instruction symbol | Description | Reference |
|---|---|---|
| B* | (s1) × (s2) → [(d)+1, (d)] | Page 217 |
| B*P | | |
| B/ | (s1) ÷ (s2) → quotient (d), remainder (d)+1 | Page 218 |
| B/P | | |

## ■Multiplying/dividing BCD 8-digit data

| Instruction symbol | Description | Reference |
|---|---|---|
| DB* | [(s1)+1, (s1)] × [(s2)+1, (s2)] → [(d)+3, (d)+2, (d)+1, (d)] | Page 220 |
| DB*P | | |
| DB/ | [(s1)+1, (s1)] ÷ [(s2)+1, (s2)] → quotient [(d)+1, (d)], remainder [(d)+3, (d)+2] | Page 222 |
| DB/P | | |

## ■Adding/subtracting 16-bit binary block data

| Instruction symbol | Description | Reference |
|---|---|---|
| BK+ | Adds the 16-bit binary bit data in the device area ((n) points) from (s1) and the data or constants in the device area ((n) points) from (s2) at once, and stores the result in the device area ((n) points) from (d). | Page 224 |
| BK+P | | |
| BK+_U | | |
| BK+P_U | | |
| BK- | Subtracts the 16-bit binary bit data in the device area ((n) points) from (s1) and the data or constants in the device area ((n) points) from (s2) at once, and stores the result in the device area ((n) points) from (d). | Page 226 |
| BK-P | | |
| BK-_U | | |
| BK-P_U | | |

## ■Adding/subtracting 32-bit binary block data

| Instruction symbol | Description | Reference |
|---|---|---|
| DBK+ | Adds the 32-bit binary bit data in the device area ((n) points) from (s1) and the 32-bit data or constants in the device area ((n) points) from (s2), and stores the result in the device area specified by (d) and later. | Page 228 |
| DBK+P | | |
| DBK+_U | | |
| DBK+P_U | | |
| DBK- | Subtracts the 32-bit binary bit data in the device area ((n) points) from (s1) and the 32-bit data or constants in the device area ((n) points) from (s2) and later, and stores the result in the device area specified by (d) and later. | Page 231 |
| DBK-P | | |
| DBK-_U | | |
| DBK-P_U | | |

## ■Incrementing/decrementing 16-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| INC | (d) + 1 → (d) | Page 233 |
| INCP | | |
| INC_U | | |
| INCP_U | | |
| DEC | (d) - 1 → (d) | Page 234 |
| DECP | | |
| DEC_U | | |
| DECP_U | | |

## ■Incrementing/decrementing 32-bit binary data

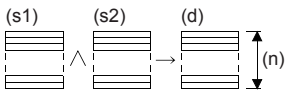
| Instruction symbol | Description | Reference |
|---|---|---|
| DINC | [(d)+1, (d)] + 1 → [(d)+1, (d)] | Page 235 |
| DINCP | | |
| DINC_U | | |
| DINCP_U | | |
| DDEC | [(d)+1, (d)] -1 → [(d)+1, (d)] | Page 236 |
| DDECP | | |
| DDEC_U | | |
| DDECP_U | | |

## Logical operation instruction

### ■Performing an AND operation on 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| WAND | (d)∧(s) → (d) | Page 237 |
| WANDP | | |
| WAND | (s1)∧ (s2) → (d) | Page 238 |
| WANDP | | |
| DAND | [(d)+1, (d)] ∧[(s)+1, (s)] → [(d)+1, (d)] | Page 239 |
| DANDP | | |
| DAND | [(s1)+1, (s1)]∧ [(s2)+1, (s2)] → [(d)+1, (d)] | Page 240 |
| DANDP | | |

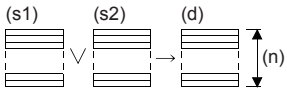
### ■Performing an AND operation on 16-bit block data

| Instruction symbol | Description | Reference |
|---|---|---|
| BKAND |  | Page 241 |
| BKANDP | | |

### ■Performing an OR operation on 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| WOR | (d)∨(s) → (d) | Page 243 |
| WORP | | |
| WOR | (s1) ∨(s2) → (d) | Page 244 |
| WORP | | |
| DOR | [(d)+1, (d)] ∨ [(s)+1, (s)] → [(d)+1, (d)] | Page 245 |
| DORP | | |
| DOR | [(s1)+1, (s1)] ∨ [(s2)+1, (s2)] → [(d)+1, (d)] | Page 246 |
| DORP | | |

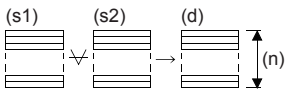
### ■Performing an OR operation on 16-bit block data

| Instruction symbol | Description | Reference |
|---|---|---|
| BKOR |  | Page 247 |
| BKORP | | |

### ■Performing an XOR operation on 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| WXOR | (d) ∀ (s) → (d) | Page 249 |
| WXORP | | |
| WXOR | (s1) ∀ (s2) → (d) | Page 250 |
| WXORP | | |
| DXOR | [(d)+1, (d)] ∀ [(s)+1, (s)] → [(d)+1, (d)] | Page 251 |
| DXORP | | |
| DXOR | [(s1)+1, (s1)] ∀ [(s2)+1, (s2)] → [(d)+1, (d)] | Page 252 |
| DXORP | | |

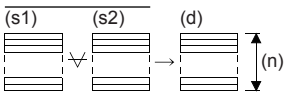
### ■Performing an XOR operation on 16-bit block data

| Instruction symbol | Description | Reference |
|---|---|---|
| BKXOR |  | Page 253 |
| BKXORP | | |

### ■Performing an XNOR operation on 16-bit/32-bit data

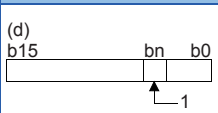
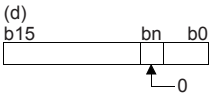
| Instruction symbol | Description | Reference |
|---|---|---|
| WXNR<br>WXNRP | $\overline{(d) \veebar (s)} \rightarrow (d)$ | Page 255 |
| WXNR<br>WXNRP | $\overline{(s1) \veebar (s2)} \rightarrow (d)$ | Page 256 |
| DXNR<br>DXNRP | $\overline{[(d)+1, (d)] \veebar [(s)+1, (s)]} \rightarrow [(d)+1, (d)]$ | Page 257 |
| DXNR<br>DXNRP | $\overline{[(s1)+1, (s1)] \veebar [(s2)+1, (s2)]} \rightarrow [(d)+1, (d)]$ | Page 258 |

### ■Performing an XNOR operation on 16-bit block data

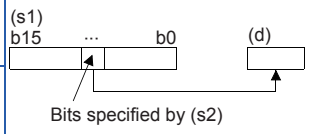
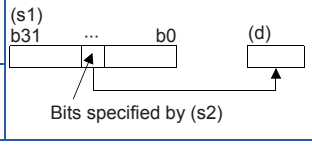
| Instruction symbol | Description | Reference |
|---|---|---|
| BKXNR<br>BKXNRP |  | Page 259 |

## Bit processing instruction

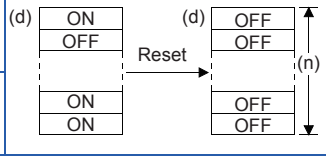
### ■Setting/resetting a bit in the word device

| Instruction symbol | Description | Reference |
|---|---|---|
| BSET<br>BSETP |  | Page 261 |
| BRST<br>BRSTP |  | Page 262 |

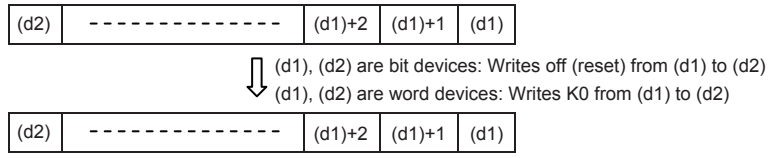
### ■Performing a bit test

| Instruction symbol | Description | Reference |
|---|---|---|
| TEST<br>TESTP |  | Page 263 |
| DTEST<br>DTESTP |  | Page 264 |

### ■Batch-resetting bit devices

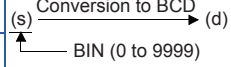
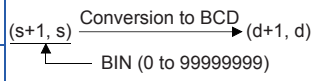
| Instruction symbol | Description | Reference |
|---|---|---|
| BKRST<br>BKRSTP |  | Page 265 |

### ■Batch-resetting devices

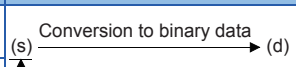
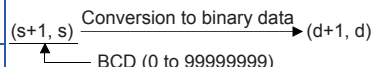
| Instruction symbol | Description | Reference |
|---|---|---|
| ZRST<br>ZRSTP |  | Page 266 |

## Data conversion instruction

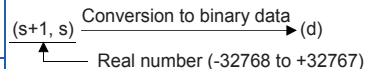
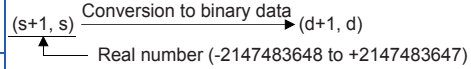
### ■Converting binary data to BCD 4-digit/8-digit data

| Instruction symbol | Description | Reference |
|---|---|---|
| BCD<br>BCDP |  | Page 268 |
| DBCD<br>DBCDP |  | Page 270 |

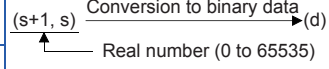
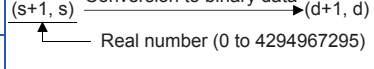
### ■Converting BCD 4-digit/8-digit data to binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| BIN<br>BINP |  | Page 272 |
| DBIN<br>DBINP |  | Page 274 |

### ■Converting single-precision real number to 16-bit/32-bit signed binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| FLT2INT<br>FLT2INTP |  | Page 276 |
| FLT2DINT<br>FLT2DINTP |  | Page 278 |

### ■Converting single-precision real number to 16-bit/32-bit unsigned binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| FLT2UINT<br>FLT2UINTP |  | Page 277 |
| FLT2UDINT<br>FLT2UDINTP |  | Page 279 |

### ■Converting 16-bit signed binary data to 16-bit/32-bit unsigned binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| INT2UINT<br>INT2UINTP | Converts 16-bit signed data in the device specified by (s) to 16-bit unsigned data, and stores the converted data in the device specified by (d). | Page 280 |
| INT2UDINT<br>INT2UDINTP | Converts 16-bit signed data in the device specified by (s) to 32-bit unsigned data, and stores the converted data in the device specified by (d). | Page 282 |

### ■Converting 16-bit signed binary data to 32-bit signed binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| INT2DINT | Converts 16-bit signed data in the device specified by (s) to 32-bit signed data, and stores the converted data in the device specified by (d). | Page 281 |
| INT2DINTP | | |

### ■Converting 16-bit unsigned binary data to 16-bit/32-bit signed binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| UINT2INT | Converts 16-bit unsigned data in the device specified by (s) to 16-bit signed data, and stores the converted data in the device specified by (d). | Page 283 |
| UINT2INTP | | |
| UINT2DINT | Converts 16-bit unsigned data in the device specified by (s) to 32-bit signed data, and stores the converted data in the device specified by (d). | Page 284 |
| UINT2DINTP | | |

### ■Converting 16-bit unsigned binary data to 32-bit unsigned binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| UINT2UDINT | Converts 16-bit unsigned data in the device specified by (s) to 32-bit unsigned data, and stores the converted data in the device specified by (d). | Page 285 |
| UINT2UDINTP | | |

### ■Converting 32-bit signed binary data to 16-bit signed binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| DINT2INT | Converts 32-bit signed data in the device specified by (s) to 16-bit signed data, and stores the converted data in the device specified by (d). | Page 286 |
| DINT2INTP | | |

### ■Converting 32-bit signed binary data to 16-bit/32-bit unsigned binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| DINT2UINT | Converts 32-bit signed data in the device specified by (s) to 16-bit unsigned data, and stores the converted data in the device specified by (d). | Page 287 |
| DINT2UINTP | | |
| DINT2UDINT | Converts 32-bit signed data in the device specified by (s) to 32-bit unsigned data, and stores the converted data in the device specified by (d). | Page 288 |
| DINT2UDINTP | | |

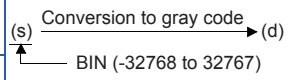
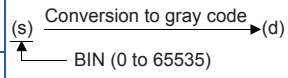
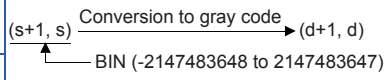
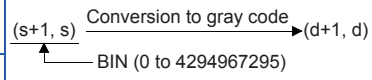
### ■Converting 32-bit unsigned binary data to 16-bit/32-bit signed binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| UDINT2INT | Converts 32-bit unsigned data in the device specified by (s) to 16-bit signed data, and stores the converted data in the device specified by (d). | Page 289 |
| UDINT2INTP | | |
| UDINT2DINT | Converts 32-bit unsigned data in the device specified by (s) to 32-bit signed data, and stores the converted data in the device specified by (d). | Page 291 |
| UDINT2DINTP | | |

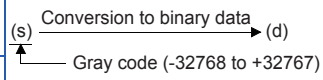
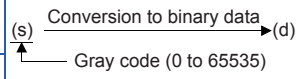
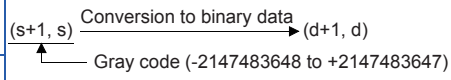
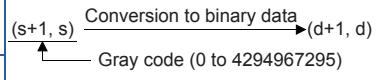
### ■Converting 32-bit unsigned binary data to 16-bit unsigned binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| UDINT2UINT | Converts 32-bit unsigned data in the device specified by (s) to 16-bit unsigned data, and stores the converted data in the device specified by (d). | Page 290 |
| UDINT2UINTP | | |

## ■Converting 16-bit/32-bit binary data to Gray code

| Instruction symbol | Description | Reference |
|---|---|---|
| GRY<br>GRYP |  | Page 292 |
| GRY_U<br>GRYP_U |  | |
| DGRY<br>DGRYP |  | Page 293 |
| DGRY_U<br>DGRYP_U |  | |

## ■Converting Gray code to 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| GBIN<br>GBINP |  | Page 294 |
| GBIN_U<br>GBINP_U |  | |
| DGBIN<br>DGBINP |  | Page 295 |
| DGBIN_U<br>DGBINP_U |  | |

## ■Converting decimal ASCII to 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| DABIN<br>DABINP<br>DABIN_U<br>DABINP_U | Converts a 5-digit decimal ASCII value in the device specified by (s) to a 1 word binary value, and stores the converted data in the word device number specified by (d). | Page 296 |
| DDABIN<br>DDABINP<br>DDABIN_U<br>DDABINP_U | Converts a 10-digit decimal ASCII value in the device specified by (s) to a 2 word binary value, and stores the converted data in the word device number specified by (d). | Page 298 |

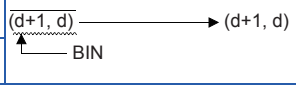
## ■Converting ASCII to HEX

| Instruction symbol | Description | Reference |
|---|---|---|
| HEXA<br>HEXAP | Converts the ASCII data stored in the number of characters specified by (n) starting from device specified in (s), and stores the converted data in the device specified by (d) onwards. | Page 300 |

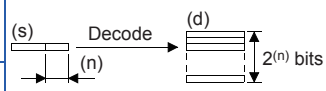
## ■Converting character string to 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| VAL<br>VALP<br>VAL_U<br>VALP_U | Converts a character string including decimal point in the device specified by (s) to a 1 word binary value and number of decimal fraction digits, and stores the converted data in the devices specified by (d1) and (d2). | Page 303 |
| DVAL<br>DVALP<br>DVAL_U<br>DVALP_U | Converts a character string including decimal point in the device specified by (s) to a 2 words binary value and number of decimal fraction digits, and stores the converted data in the devices specified by (d1) and (d2). | Page 306 |

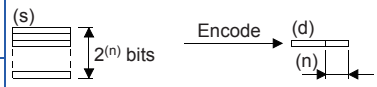
## ■Two's complement of 16-bit/32-bit binary data (sign inversion)

| Instruction symbol | Description | Reference |
|---|---|---|
| NEG<br>NEGP |  | Page 309 |
| DNEG<br>DNEGP |  | Page 310 |

## ■Decoding from 8 to 256 bits

| Instruction symbol | Description | Reference |
|---|---|---|
| DECO<br>DECOP |  | Page 311 |

## ■Encoding from 256 to 8 bits

| Instruction symbol | Description | Reference |
|---|---|---|
| ENCO<br>ENCOP |  | Page 313 |

## ■Seven-segment decoding

| Instruction symbol | Description | Reference |
|---|---|---|
| SEGD<br>SEGDP | Decoded to data for the seven-segment display unit in the device specified by (s),and stores in the device specified by (d). | Page 314 |

## ■Seven-segment with latch

| Instruction symbol | Description | Reference |
|---|---|---|
| SEGL | The 4-digit numeric value stored in (s) is converted into BCD data, and each digit is output to the seven-segment display unit with the BCD decoder by the time division method. | Page 316 |

## ■Separating 4 bits from 16-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| DIS<br>DISP | Separates the 16-bit data specified by (s) into 4-bit units and stores in the lower 4 bits of (n) points from (d). (n < 4) | Page 318 |

## ■Connecting 4 bits to 16-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| UNI<br>UNIP | Connects the lower 4 bits of (n) points from the device specified by (s), and stores the result in the device specified by (d). (n < 4) | Page 319 |

### ■Separating/connecting the specified number of bits

| Instruction symbol | Description | Reference |
|---|---|---|
| NDIS | Separates the data in the devices starting from the one specified by (s1) into bits specified by the devices from (s2), and stores them to the devices starting from the one specified by (d). | Page 320 |
| NDISP | | |
| NUNI | Connects the data in the devices starting from the one specified by (s1) with bits specified by the devices from (s2), and stores them to the devices starting from the one specified by (d). | Page 322 |
| NUNIP | | |

### ■Separating/connecting data in byte units

| Instruction symbol | Description | Reference |
|---|---|---|
| WTOB | Breaks (n) points of 16 bit data from the device specified by (s) into 8-bit units, and stores in the devices starting from the one specified by (d). | Page 324 |
| WTOBP | | |
| BTOW | Connects the lower 8 bits of 16-bit data of (n) points from the device specified by (s) into 16-bit units, and stores in the devices starting from the one specified by (d). | Page 326 |
| BTOWP | | |

## Digital Switch

| Instruction symbol | Description | Reference |
|---|---|---|
| DSW | The value of the (n) sets of digital switches connected to (s), and stored for (d2). | Page 328 |

## Data transfer instruction

### ■Transferring 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| MOV | (s) ⟶ (d) | Page 330 |
| MOVP | | |
| DMOV | (s+1, s) ⟶ (d+1, d) | Page 331 |
| DMOVP | | |

### ■Inverting and transferring 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| CML | $\overline{(s)}$ ⟶ (d) | Page 332 |
| CMLP | | |
| DCML | $\overline{(s+1, s)}$ ⟶ (d+1, d) | Page 333 |
| DCMLP | | |

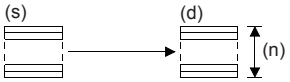
### ■Shift move

| Instruction symbol | Description | Reference |
|---|---|---|
| SMOV | Shifts the specified no. of digits from the word device specified by (s), and store in (d). | Page 334 |
| SMOVP | | |

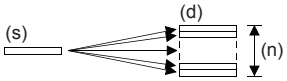
### ■Inverting and transferring 1-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| CMLB | Inverts the bit data specified by (s), and store in (d). | Page 336 |
| CMLBP | | |

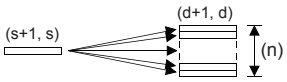
### ■Transferring 16-bit block data (65535 points maximum)

| Instruction symbol | Description | Reference |
|---|---|---|
| BMOV | (s) (d)  (n) | Page 337 |
| BMOVP | (n) = 1 to 65535 | |

## ■Transferring identical 16-bit block data (65535 points maximum)

| Instruction symbol | Description | Reference |
|---|---|---|
| FMOV<br>FMOVP |  <br>(n) = 1 to 65535 | Page 339 |

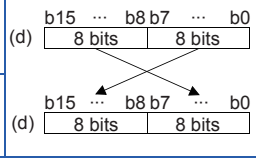
## ■Transferring identical 32-bit block data (65535 points maximum)

| Instruction symbol | Description | Reference |
|---|---|---|
| DFMOV<br>DFMOVP |  <br>(n) = 1 to 65535 | Page 340 |

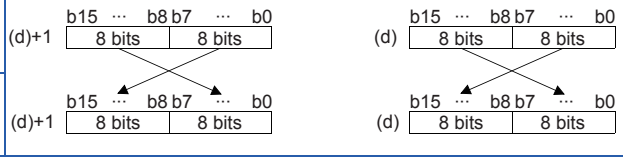
## ■Exchanging 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| XCH<br>XCHP | (d1) ⟷ (d2) | Page 341 |
| DXCH<br>DXCHP | (d1+1, d1) ⟷ (d2+1, d2) | Page 342 |

## ■Exchanging the upper and lower bytes of 16-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| SWAP<br>SWAPP |  | Page 343 |

## ■Exchanging the upper and lower bytes of 32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| DSWAP<br>DSWAPP |  | Page 344 |

## ■Transferring 1-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| MOVB<br>MOVBP | Stores the bit data specified by (s) in (d). | Page 345 |

## ■Parallel run (octal mode) (16-bit data)

| Instruction symbol | Description | Reference |
|---|---|---|
| PRUN<br>PRUNP | Handles device number specified by (s) in nibble specification and (d) as octal, and stores into (d) from (s). | Page 346 |

## ■Parallel run (octal mode) (32-bit data)

| Instruction symbol | Description | Reference |
|---|---|---|
| DPRUN<br>DPRUNP | Handles device number specified by (s) in nubble specification and (d) as octal, and stores into (d) from (s). | Page 348 |

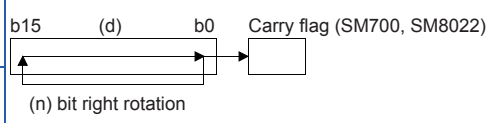
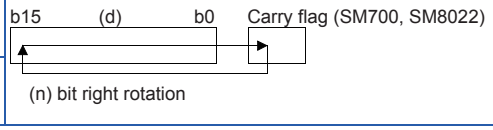
## ■Transferring n-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| BLKMOVB<br>BLKMOVBP | Block transfers bit data for (n) points from (s) to bit data for (n) points from (d). | Page 350 |

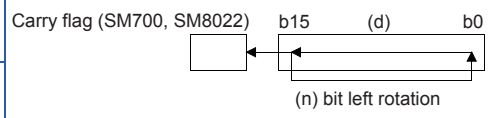
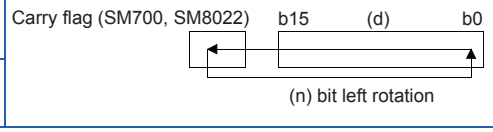
# 3.3 Application Instruction

## Rotation instruction

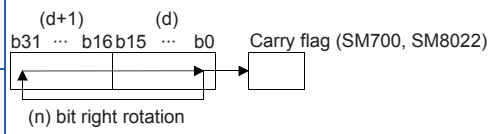
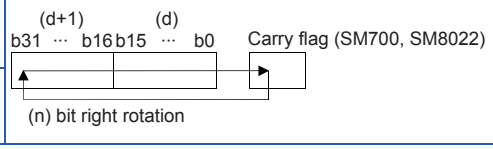
### ■Rotating 16-bit data to the right

| Instruction symbol | Description | Reference |
|---|---|---|
| ROR<br>RORP |  | Page 352 |
| RCR<br>RCRP |  | |

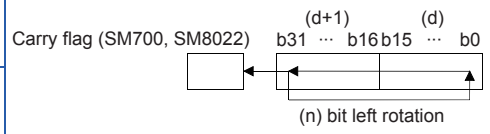
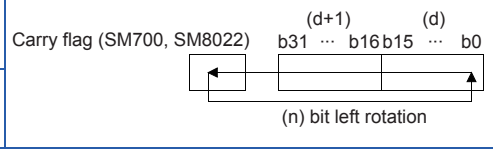
### ■Rotating 16-bit data to the left

| Instruction symbol | Description | Reference |
|---|---|---|
| ROL<br>ROLP |  | Page 355 |
| RCL<br>RCLP |  | |

### ■Rotating 32-bit data to the right

| Instruction symbol | Description | Reference |
|---|---|---|
| DROR<br>DRORP |  | Page 358 |
| DRCR<br>DRCRP |  | |

### ■Rotating 32-bit data to the left

| Instruction symbol | Description | Reference |
|---|---|---|
| DROL<br>DROLP |  | Page 360 |
| DRCL<br>DRCLP |  | |

## Program branch instruction

### ■Pointer branch

| Instruction symbol | Description | Reference |
|---|---|---|
| CJ | When the input condition is met, jump to pointer (P) | Page 362 |
| CJP | | |

### ■Jumping to END

| Instruction symbol | Description | Reference |
|---|---|---|
| GOEND | When the input condition is met, jump to END instruction | Page 365 |

## Program execution control instruction

### ■Disabling/enabling interrupt programs

| Instruction symbol | Description | Reference |
|---|---|---|
| DI | Disables the execution of interrupt programs. | Page 366 |
| EI | Releases the execution disabled state of interrupt program. | |

### ■Disabling the interrupt program with specified priority or lower

| Instruction symbol | Description | Reference |
|---|---|---|
| DI | Disables the execution of the interrupt program with a priority specified by (s) or lower until the EI instruction is executed. | Page 368 |

### ■Interrupt program mask

| Instruction symbol | Description | Reference |
|---|---|---|
| IMASK | Interrupt disable/enable settings | Page 371 |

### ■Disabling/enabling the specified interrupt pointer

| Instruction symbol | Description | Reference |
|---|---|---|
| SIMASK | Disables/enables the interrupt pointer specified by (I) | Page 373 |

### ■Returning from the interrupt program

| Instruction symbol | Description | Reference |
|---|---|---|
| IRET | Returns from the interrupt program to the sequence program | Page 374 |

### ■Resetting the watchdog timer

| Instruction symbol | Description | Reference |
|---|---|---|
| WDT | Resets the watchdog timer (WDT) in the program | Page 375 |
| WDTP | | |

## Structuring instruction

### ■Performing the FOR to NEXT instruction loop

| Instruction symbol | Description | Reference |
|---|---|---|
| FOR | Execute the instructions between FOR instruction and NEXT instruction (n) times | Page 376 |
| NEXT | | |

### ■Forcibly terminating the FOR to NEXT instruction loop

| Instruction symbol | Description | Reference |
|---|---|---|
| BREAK | Forcibly end execution between FOR instruction and NEXT instruction, and jump to pointer (P) | Page 378 |
| BREAKP | | |

### ■Calling a subroutine program

| Instruction symbol | Description | Reference |
|---|---|---|
| CALL | Executes a subroutine program specified by (P) when the input condition is met. | Page 380 |
| CALLP | | |

### ■Returning from the subroutine program

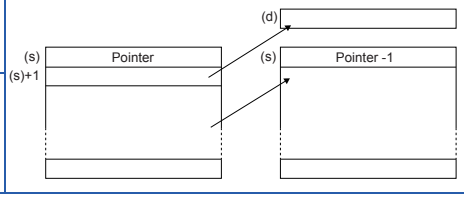
| Instruction symbol | Description | Reference |
|---|---|---|
| RET | Returns from the subroutine program. | Page 384 |
| SRET | | |

### ■Calling a subroutine program

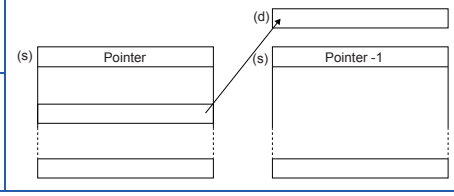
| Instruction symbol | Description | Reference |
|---|---|---|
| XCALL | Executes a subroutine program specified by (P) when the input condition is met.<br>Carry out non-execution processing for the subroutine program (P), when input conditions are not met. | Page 385 |

## Data table operation instruction

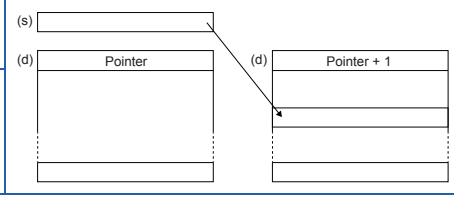
### ■Reading the oldest data from the data table

| Instruction symbol | Description | Reference |
|---|---|---|
| SFRD |  | Page 387 |
| SFRDP | | |

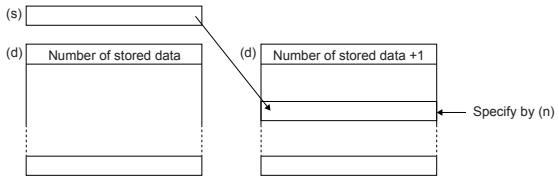
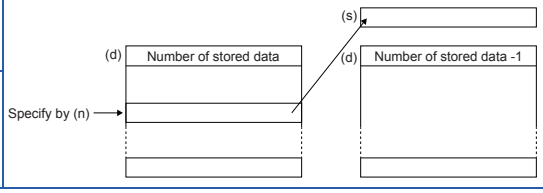
### ■Reading the newest data from the data table

| Instruction symbol | Description | Reference |
|---|---|---|
| POP |  | Page 389 |
| POPP | | |

### ■Writing data to the data table

| Instruction symbol | Description | Reference |
|---|---|---|
| SFWR |  | Page 391 |
| SFWRP | | |

### ■Deleting/inserting data from/to the data table

| Instruction symbol | Description | Reference |
|---|---|---|
| FINS<br>FINSP |  | Page 393 |
| FDEL<br>FDELP |  | Page 395 |

## Character string operation instruction

### ■Comparing character strings

| Instruction symbol | Description | Reference |
|---|---|---|
| LD$=, AND$=, OR$= | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] = [Character string (s2)]: Conductive state<br>[Character string (s1)] ≠ [Character string (s2)]: Non-Conductive state | Page 397 |
| LD$<>, AND$<>, OR$<> | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] ≠ [Character string (s2)]: Conductive state<br>[Character string (s1)] = [Character string (s2)]: Non-Conductive state | |
| LD$>, AND$>, OR$> | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] > [Character string (s2)]: Conductive state<br>[Character string (s1)] ≤ [Character string (s2)]: Non-Conductive state | |
| LD$<=, AND$<=, OR$<= | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] ≤ [Character string (s2)]: Conductive state<br>[Character string (s1)] > [Character string (s2)]: Non-Conductive state | |
| LD$<, AND$<, OR$< | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] < [Character string (s2)]: Conductive state<br>[Character string (s1)] ≥ [Character string (s2)]: Non-Conductive state | |
| LD$>=, AND$>=, OR$> | Compares the character string (s1) with the character string (s2) one character at a time.*1<br>[Character string (s1)] ≥ [Character string (s2)]: Conductive state<br>[Character string (s1)] < [Character string (s2)]: Non-Conductive state | |

*1 The following shows comparison conditions for comparing character strings.
- Match: All characters in the strings must match
- Larger string: In case of different character strings, character string with the larger character code (If character string lengths are different, the longer character string)
- Smaller string: In case of different character strings, character string with the smaller character code (If character string lengths are different, the shorter character string)

### ■Concatenating character strings

| Instruction symbol | Description | Reference |
|---|---|---|
| $+<br>$+P | • In case of 2 operands<br>Connect the character string specified by (s) to the end of the character string specified by (d), and store in (d). | Page 400 |
| $+<br>$+P | • In case of 3 operands<br>Connect the character string specified by (s2) to the end of the character string specified by (s1), and store in (d). | Page 402 |

### ■Transferring character strings

| Instruction symbol | Description | Reference |
|---|---|---|
| $MOV<br>$MOVP | Transfer the character strings specified by (s) to the devices specified by (d) onwards. | Page 404 |

## ■Converting 16-bit/32-bit binary data to decimal ASCII

| Instruction symbol | Description | Reference |
|---|---|---|
| BINDA | Converts the 1 word binary value specified by (s) to 5 digits decimal ASCII value, and stores in the word device specified by (d). | Page 406 |
| BINDAP | | |
| BINDA_U | | |
| BINDAP_U | | |
| DBINDA | Converts the 2 word binary value specified by (s) to 10 digits decimal ASCII value, and stores in the word device area specified by (d) onwards. | Page 408 |
| DBINDAP | | |
| DBINDA_U | | |
| DBINDAP_U | | |

## ■Converting HEX code data to ASCII

| Instruction symbol | Description | Reference |
|---|---|---|
| ASCI | Converts the (n) characters within the HEX code data specified by (s) to ASCII, and stores in the device area specified by (d) onwards. | Page 410 |
| ASCIP | | |

## ■Converting 16-bit/32-bit binary data to character string

| Instruction symbol | Description | Reference |
|---|---|---|
| STR | Converts the 1 word binary value specified by (s2) to the decimal character string with total number of digits and the number of digits in the decimal fraction part as specified in (s1), and stores this in the device specified by (d). | Page 414 |
| STRP | | |
| STR_U | | |
| STRP_U | | |
| DSTR | Convert the 2 word binary value specified by (s2) to the decimal character string with total number of digits and the number of digits in the decimal fraction part as specified in (s1), and stores this in the device specified by (d). | Page 416 |
| DSTRP | | |
| DSTR_U | | |
| DSTRP_U | | |

## ■Converting single-precision real number to character string

| Instruction symbol | Description | Reference |
|---|---|---|
| ESTR | Converts the single-precision real number data specified by (s1) to a character string, and store this in the device specified by (d). | Page 419 |
| ESTRP | | |
| DESTR | | |
| DESTRP | | |

## ■Detecting a character string length

| Instruction symbol | Description | Reference |
|---|---|---|
| LEN | Stores the length of the character string data stored in the device specified by (s) in the device specified by (d). | Page 424 |
| LENP | | |

## ■Extracting character string data from the right/left

| Instruction symbol | Description | Reference |
|---|---|---|
| RIGHT | Stores the (n) characters from the last character of the character string specified by (s) in the device specified by (d). | Page 426 |
| RIGHTP | | |
| LEFT | Stores the (n) characters from the first character of the character string specified by (s) in the device specified by (d). | Page 428 |
| LEFTP | | |

### ■Storing/replacing the specified number of character strings

| Instruction symbol | Description | Reference |
|---|---|---|
| MIDR | Stores the specified number of characters from the position specified by (s2) of the character string (s1) into the device specified by (d). | Page 430 |
| MIDRP | | |
| MIDW | Stores the specified number of characters from the character string (s1) into the location specified by (s2) of the character string (d). | Page 432 |
| MIDWP | | |

### ■Searching character string

| Instruction symbol | Description | Reference |
|---|---|---|
| INSTR | Searches the character string in the device specified by (s2), starting from the (s3)th character, for the character string in the device specified by (s1), and stores the matching location in the device specified by (d). | Page 435 |
| INSTRP | | |

### ■Inserting character string

| Instruction symbol | Description | Reference |
|---|---|---|
| STRINS | Inserts the character string data specified in (s1) at the position (s2)(Insert position) from the beginning of the character string data specified by (d). | Page 437 |
| STRINSP | | |

### ■Deleting character string

| Instruction symbol | Description | Reference |
|---|---|---|
| STRDEL | From the head of the character string data specified in (d), delete (n2) characters from the location specified as the character number (n1) (deletion start location). | Page 439 |
| STRDELP | | |

## Real number instruction

### ■Comparing single-precision real numbers

| Instruction symbol | Description | Reference |
|---|---|---|
| LDE=, ANDE=, ORE= | [(s1)+1, (s1)] = [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≠ [(s2)+1, (s2)]: Non-Conductive | Page 441 |
| LDE<>, ANDE<>, ORE<> | [(s1)+1, (s1)] ≠ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] = [(s2)+1, (s2)]: Non-Conductive | |
| LDE>, ANDE>, ORE> | [(s1)+1, (s1)] > [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≤ [(s2)+1, (s2)]: Non-Conductive | |
| LDE<=, ANDE<=, ORE<= | [(s1)+1, (s1)] ≤ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] > [(s2)+1, (s2)]: Non-Conductive | |
| LDE<, ANDE<, ORE< | [(s1)+1, (s1)] < [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] ≥ [(s2)+1, (s2)]: Non-Conductive | |
| LDE>=, ANDE>=, ORE> | [(s1)+1, (s1)] ≥ [(s2)+1, (s2)]: Conductive<br>[(s1)+1, (s1)] < [(s2)+1, (s2)]: Non-Conductive | |
| DECMP | This instruction compares two data values (single-precision real numbers), and outputs the result (larger, smaller or equal) to three bit devices. | Page 443 |
| DECMPP | | |
| DEZCP | This instruction compares two data values (single-precision real numbers), and outputs the result (larger, smaller or data band) to three bit devices. | Page 445 |
| DEZCPP | | |

### ■Adding/subtracting single-precision real numbers

| Instruction symbol | Description | Reference |
|---|---|---|
| E+ | • In case of 2 operands<br>[(d)+1, (d)] + [(s)+1, (s)] → [(d)+1, (d)] | Page 447 |
| E+P | | |
| E+ | • In case of 3 operands<br>[(s1)+1, (s1)] + [(s2)+1, (s2)] → [(d)+1, (d)] | Page 449 |
| E+P | | |
| DEADD | | Page 455 |
| DEADDP | | |
| E- | • In case of 2 operands<br>[(d)+1, (d)] - [(s)+1, (s)] → [(d)+1, (d)] | Page 451 |
| E-P | | |

| Instruction symbol | Description | Reference |
|---|---|---|
| E- | • In case of 3 operands<br>[(s1)+1, (s1)] - [(s2)+1, (s2)] → [(d)+1, (d)] | Page 453 |
| E-P | | |
| DESUB | | Page 457 |
| DESUBP | | |

## ■Multiplying/dividing single-precision real numbers

| Instruction symbol | Description | Reference |
|---|---|---|
| E* | [(s1)+1, (s1)] × [(s2)+1, (s2)] → [(d)+1, (d)] | Page 459 |
| E*P | | |
| DEMUL | | Page 463 |
| DEMULP | | |
| E/ | [(s1)+1, (s1)] ÷ [(s2)+1, (s2)] → quotient [(d)+1, (d)] | Page 461 |
| E/P | | |
| DEDIV | | Page 465 |
| DEDIVP | | |

## ■Converting 16-bit/32-bit signed binary data to single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| INT2FLT | Converts the 16-bit signed binary data in the device specified by (s) to single-precision real number, and stores the converted data in the device specified by (d). | Page 467 |
| INT2FLTP | | |
| DINT2FLT | Converts the 32-bit signed binary data in the device specified by (s) to single-precision real number, and stores the converted data in the device specified by (d). | Page 469 |
| DINT2FLTP | | |

## ■Converting 16-bit/32-bit unsigned binary data to single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| UINT2FLT | Converts the 16-bit unsigned binary data in the device specified by (s) to single-precision real number, and stores the converted data in (d). | Page 468 |
| UINT2FLTP | | |
| UDINT2FLT | Converts the 32-bit unsigned binary data in the device specified by (s) to single-precision real number, and stores the converted data in (d). | Page 470 |
| UDINT2FLTP | | |

## ■Converting character string to single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| EVAL | Converts the character string specified by (s) to a single-precision real number, and stores the converted data in (d). | Page 471 |
| EVALP | | |
| DEVAL | | |
| DEVALP | | |

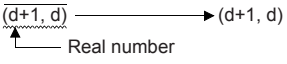
## ■Converting binary floating point to decimal floating point

| Instruction symbol | Description | Reference |
|---|---|---|
| DEBCD | Converts the binary floating point specified by (s) into decimal floating point, and stores in (d). | Page 474 |
| DEBCDP | | |

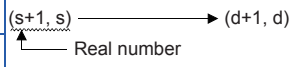
## ■Converting decimal floating point to binary floating point

| Instruction symbol | Description | Reference |
|---|---|---|
| DEBIN | Converts the decimal floating point specified by (s) into binary floating point, and stores in (d). | Page 476 |
| DEBINP | | |

## ■Inverting the sign of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| ENEG |  | Page 478 |
| ENEGP | | |
| DENEG | | |
| DENEGP | | |

## ■Transferring single-precision real number data

| Instruction symbol | Description | Reference |
|---|---|---|
| EMOV<br>EMOVP<br>DEMOV<br>DEMOVP |  | Page 479 |

## ■Calculating the sine of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| SIN<br>SINP<br>DSIN<br>DSINP | Sin [(s)+1, (s)] → [(d)+1, (d)] | Page 480 |

## ■Calculating the cosine of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| COS<br>COSP<br>DCOS<br>DCOSP | Cos [(s)+1, (s)] → [(d)+1, (d)] | Page 482 |

## ■Calculating the tangent of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| TAN<br>TANP<br>DTAN<br>DTANP | Tan [(s)+1, (s)] → [(d)+1, (d)] | Page 484 |

## ■Calculating the arc sine of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| ASIN<br>ASINP<br>DASIN<br>DASINP | $Sin^{-1}$ [(s)+1, (s)] → [(d)+1, (d)] | Page 486 |

## ■Calculating the arc cosine of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| ACOS<br>ACOSP<br>DACOS<br>DACOSP | $Cos^{-1}$ [(s)+1, (s)] → [(d)+1, (d)] | Page 488 |

## ■Calculating the arc tangent of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| ATAN<br>ATANP<br>DATAN<br>DATANP | $Tan^{-1}$ [(s)+1, (s)] → [(d)+1, (d)] | Page 490 |

### ■Converting single-precision real number angle to radian

| Instruction symbol | Description | Reference |
|---|---|---|
| RAD | (s+1, s) ⟶ (d+1, d)<br>Converts from degrees to radians | Page 492 |
| RADP | | |
| DRAD | | |
| DRADP | | |

### ■Converting single-precision real number radian to angle

| Instruction symbol | Description | Reference |
|---|---|---|
| DEG | (s+1, s) ⟶ (d+1, d)<br>Converts from radians to degrees | Page 494 |
| DEGP | | |
| DDEG | | |
| DDEGP | | |

### ■Calculating the square root of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| DESQR | $\sqrt{(s+1, s)}$ ⟶ (d+1, d) | Page 496 |
| DESQRP | | |
| ESQRT | | |
| ESQRTP | | |

### ■Calculating the exponent of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| EXP | $e^{[(s)+1, (s)]} \rightarrow [(d)+1, (d)]$ | Page 497 |
| EXPP | | |
| DEXP | | |
| DEXPP | | |

### ■Calculating the natural logarithm of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| LOG | Loge[(s)+1, (s)] → [(d)+1, (d)] | Page 499 |
| LOGP | | |
| DLOGE | | |
| DLOGEP | | |

### ■Calculating the exponentiation of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| POW | $[(s1)+1, (s1)]^{[(s2)+1, (s2)]} \rightarrow [(d)+1, (d)]$ | Page 501 |
| POWP | | |

### ■Calculating the common logarithm of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| LOG10 | log10[(s)+1, (s)] → [(d)+1, (d)] | Page 503 |
| LOG10P | | |
| DLOG10 | | |
| DLOG10P | | |

### ■Searching the maximum value of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| EMAX | These instructions search for the maximum value in the (n) points of single-precision real number block data specified by the device starting from the one specified by (s), and store the maximum value in the device area specified by (d). | Page 505 |
| EMAXP | | |

### ■Searching the minimum value of single-precision real number

| Instruction symbol | Description | Reference |
|---|---|---|
| EMIN | These instructions search for the minimum value in the (n) points of single-precision real number block data specified by the device starting from the one specified by (s), and store the minimum value in the device areas specified by (d). | Page 507 |
| EMINP | | |

## Random number instruction

### ■Generating random number

| Instruction symbol | Description | Reference |
|---|---|---|
| RND | Generates a random number from 0 to 32767, and stores this in the device specified by (d). | Page 509 |
| RNDP | | |

## Index register operation instruction

### ■Saving/returning all data of the index register

| Instruction symbol | Description | Reference |
|---|---|---|
| ZPUSH | Saves the contents of index registers to the devices specified by (d) onwards. | Page 510 |
| ZPUSHP | | |
| ZPOP | Reads the data in devices specified by (d) onwards to the index registers. | Page 512 |
| ZPOPP | | |

### ■Saving/returning the selected data of the index register and long index register

| Instruction symbol | Description | Reference |
|---|---|---|
| ZPUSH | Saves the contents of the index registers and long index registers in the range specified by (s) to devices specified by (d) onwards. | Page 513 |
| ZPUSHP | | |
| ZPOP | Reads data in the devices specified by (d) onwards to the index registers and long index registers. | Page 515 |
| ZPOPP | | |

## Data control instruction

### ■Upper and lower limit control of 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| LIMIT | (s3) < (s1): The (s1) value is stored in (d)<br>(s1) ≤ (s3) ≤ (s2): The (s3) value is stored in (d)<br>(s2) < (s3): The (s2) value is stored in (d) | Page 516 |
| LIMITP | | |
| LIMIT_U | | |
| LIMITP_U | | |
| DLIMIT | [(s3)+1, (s3)] < [(s1)+1, (s1)]: The [(s1)+1, (s1)] value is stored in [(d)+1, (d)]<br>[(s1)+1, (s1)] ≤ [(s3)+1, (s3)] ≤ [(s2)+1, (s2)]: The [(s3)+1, (s3)] value is stored in [(d)+1, (d)]<br>[(s2)+1, (s2)] < [(s3)+1, (s3)]: The [(s2)+1, (s2)] value is stored in [(d)+1, (d)] | Page 518 |
| DLIMITP | | |
| DLIMIT_U | | |
| DLIMITP_U | | |

## ■Dead band control of 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| BAND<br>BANDP<br>BAND_U<br>BANDP_U | When (s1) ≤ (s3) ≤ (s2): 0 → (d)<br>When (s3) < (s1): (s3) - (s1) → (d)<br>When (s2) < (s3): (s3) - (s2) → (d) | Page 520 |
| DBAND<br>DBANDP<br>DBAND_U<br>DBANDP_U | When [(s1)+1, (s1)] ≤ [(s3)+1, (s3)] ≤ [(s2)+1, (s2)]: 0 → (d+1, d)<br>When [(s3)+1, (s3)] < [(s1)+1, (s1)]: [(s3)+1, (s3)] - [(s1)+1, (s1)] → [(d)+1, (d)]<br>When [(s2)+1, (s2)] < [(s3)+1, (s3)]: [(s3)+1, (s3)] - [(s2)+1, (s2)] → [(d)+1, (d)] | Page 522 |

## ■Zone control of 16-bit/32-bit binary data

| Instruction symbol | Description | Reference |
|---|---|---|
| ZONE<br>ZONEP<br>ZONE_U<br>ZONEP_U | When (s3) = 0: 0 → (d)<br>When (s3) > 0: (s3) + (s2) → (d)<br>When (s3) < 0: (s3) + (s1) → (d) | Page 524 |
| DZONE<br>DZONEP<br>DZONE_U<br>DZONEP_U | When [(s3)+1, (s3)] = 0: 0 → [(d)+1, (d)]<br>When [(s3)+1, (s3)] > 0: [(s3)+1, (s3)] + [(s2)+1, (s2)] → [(d)+1, (d)]<br>When [(s3)+1, (s3)] < 0: [(s3)+1, (s3)] + [(s1)+1, (s1)] → [(d)+1, (d)] | Page 526 |

## ■Scaling 16-bit/32-bit binary data (point coordinates)

| Instruction symbol | Description | Reference |
|---|---|---|
| SCL<br>SCLP<br>SCL_U<br>SCLP_U | Executes scaling using the scaling conversion data (16-bit data units) specified by (s2) for the input value specified by (s1), and then stores the result in the device specified by (d).<br>The scaling conversion is executed based on the scaling conversion data stored in the device specified by (s2) onwards. | Page 528 |
| DSCL<br>DSCLP<br>DSCL_U<br>DSCLP_U | Executes scaling using the scaling conversion data (32-bit data units) specified by (s2) for the input value specified by (s1), and then stores the result in the device specified by (d).<br>The scaling conversion is executed based on the scaling conversion data stored in the device specified by (s2) onwards. | Page 531 |

## ■Scaling 16-bit/32-bit binary data (XY coordinates)

| Instruction symbol | Description | Reference |
|---|---|---|
| SCL2<br>SCL2P<br>SCL2_U<br>SCL2P_U | Executes scaling using the scaling conversion data (16-bit data units) specified by (s2) for the input value specified by (s1), and then stores the result in the device specified by (d).<br>The scaling conversion is executed based on the scaling conversion data stored in the device specified by (s2) onwards. | Page 534 |
| DSCL2<br>DSCL2P<br>DSCL2_U<br>DSCL2P_U | Executes scaling using the scaling conversion data (32-bit data units) specified by (s2) for the input value specified by (s1), and then stores the result in the device specified by (d).<br>The scaling conversion is executed based on the scaling conversion data stored in the device specified by (s2) onwards. | Page 537 |

## Special timer instruction

### ■Teaching timer

| Instruction symbol | Description | Reference |
|---|---|---|
| TTMR | (On time of TTMR) × (s) ⟶ (d)<br>(s)=0:1, (s)=1:10, (s)=2:100 | Page 540 |

### ■Special function timer

| Instruction symbol | Description | Reference |
|---|---|---|
| STMR | The 4 points from the bit device specified by (d) operate as shown below, depending on the ON/OFF status of the input conditions for the STMR instruction:<br>(d)+0: Off delay timer output<br>(d)+1: One shot after off timer output<br>(d)+2: One shot after on timer output<br>(d)+3: On delay and off delay timer output | Page 542 |

## Special counter instruction

### ■Signed 32-bit bi-directional counters

| Instruction symbol | Description | Reference |
|---|---|---|
| UDCNTF | This instruction increments the current value of the counter specified by (d) by 1 when the operation result up to UDCNTF instruction changes from OFF to ON, and when the counter reaches the end of its count, NO contact becomes turns ON and NC contact becomes turns OFF. When the long counter specified by (d) is a high-speed counter, up-counting and down-counting are enabled. | Page 544 |

## Shortcut control instruction

### ■Rotary table shortest direction control

| Instruction symbol | Description | Reference |
|---|---|---|
| ROTC | Rotates a rotary table with (n1) divisions from the stop position to the position specified by (s)+1 in the shortest direction. | Page 546 |

## Ramp signal instruction

### ■Ramp signal

| Instruction symbol | Description | Reference |
|---|---|---|
| RAMPF | Shifts the value from the one specified by (s1) to the one specified by (s2) in (n) scans.<br>The current value is stored in the device specified by (d1)+0. | Page 549 |

## Pulse related instruction

### ■Measuring the density of 16 bit binary/32 bit binary pulses

| Instruction symbol | Description | Reference |
|---|---|---|
| SPD | Counts the pulse input from the device specified by (s1) for the duration of time specified by (s2), and stores the count in the device specified by (d). | Page 552 |
| DSPD | | Page 556 |

### ■16 bit binary/32 bit binary pulse output

| Instruction symbol | Description | Reference |
|---|---|---|
| PLSY | • When an FX3 compatible operand is specified<br>This instruction outputs a pulse at a frequency specified by (s) for the number of times specified by (n) from the output number (Y) specified by (d).<br>• When an FX5 compatible operand is specified<br>This instruction outputs a pulse at a frequency specified by (s) for the number of times specified by (n), from the output number (axis number) specified by (d). | Page 560 |
| DPLSY | | Page 568 |

### ■16 bit binary/32 bit binary pulse width modulation

| Instruction symbol | Description | Reference |
|---|---|---|
| PWM | Outputs the pulse of the cycle specified by (s2), for the ON time on specified by (s1), to the output number specified by (d). | Page 576 |
| DPWM | | Page 580 |

## Input matrix instruction

### ■Input matrix

| Instruction symbol | Description | Reference |
|---|---|---|
| MTR | Reads matrix input as 8-point input × "n"-point output (transistor) in the time division method. | Page 585 |

## Initial State

### ■Initial State

| Instruction symbol | Description | Reference |
|---|---|---|
| IST | Automatically controls the initial state and special relays in a step ladder program. | Page 588 |

## Drum sequence

### ■16-bit binary data absolute method

| Instruction symbol | Description | Reference |
|---|---|---|
| ABSD | Creates many output patterns corresponding to the current value of a counter. | Page 599 |

### ■32-bit binary data absolute method

| Instruction symbol | Description | Reference |
|---|---|---|
| DABSD | Creates many output patterns corresponding to the current value of a counter. | Page 601 |

### ■Relative method

| Instruction symbol | Description | Reference |
|---|---|---|
| INCD | This instruction compares the current value of a counter with the data table having (n) lines starting from (s1) (which occupies (n) lines × 1 device). If the counter value is equivalent to the table data, the current output is reset, and the ON/OFF status of the specified sequential outputs is controlled. | Page 603 |

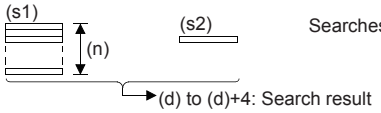
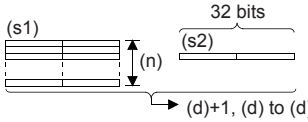
## Check code

### ■Check code

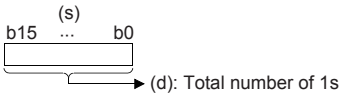
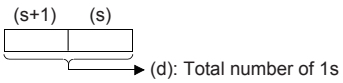
| Instruction symbol | Description | Reference |
|---|---|---|
| CCD | This instruction calculates the sum data and horizontal parity value of data stored in (s) to (s)+(n)-1. The sum data is stored in (d), and the horizontal parity value is stored in (d)+1. | Page 605 |
| CCDP | | |

## Data operation instruction

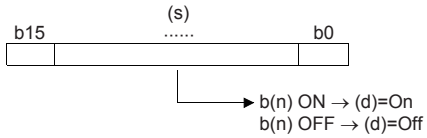
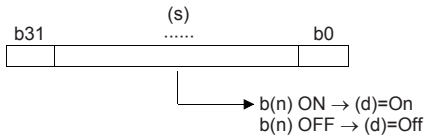
### ■Searching 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| SERMM<br>SERMMP |  Searches for data same as (s2) in (s1). | Page 608 |
| DSERMM<br>DSERMMP |  Searches for data same as (s2) in (s1). | Page 610 |

## ■Bit check of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| SUM<br>SUMP |  | Page 612 |
| DSUM<br>DSUMP |  | Page 613 |

## ■Bit judgment of 16-bit data/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| BON<br>BONP |  | Page 614 |
| DBON<br>DBONP |  | Page 615 |

## ■Searching the maximum value of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| MAX<br>MAXP<br>MAX_U<br>MAXP_U | This instruction searches the data of (n) points from the device specified by (s) in 16-bit units, and stores the maximum value in the device specified by (d). | Page 616 |
| DMAX<br>DMAXP<br>DMAX_U<br>DMAXP_U | This instruction searches the data of (n) points from the device specified by (s) in 32-bit units, and stores the maximum value in the device specified by (d). | Page 618 |

## ■Searching the minimum value of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| MIN<br>MINP<br>MIN_U<br>MINP_U | This instruction searches the data of (n) points from the device specified by (s) in 16-bit units, and stores the minimum value in the device specified by (d). | Page 620 |
| DMIN<br>DMINP<br>DMIN_U<br>DMINP_U | This instruction searches the data of (n) points from the device specified by (s) in 32-bit units, and stores the minimum value in the device specified by (d). | Page 622 |

## ■Sorting 16-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| SORTTBL<br>SORTTBL_U | In the data table (sorting source) having ((n1)×(n2)) points specified by (s), sorts the data lines in the ascending order based on the group data in the column number (n3), and stores the result in the data table (sorting result) having ((n1)×(n2)) points specified by (d). | Page 624 |

### ■16-bit/32-bit data alignment 2

| Instruction symbol | Description | Reference |
|---|---|---|
| SORTTBL2<br>SORTTBL2_U | In the data table (sorting source) of 16-bit binary data having (n1×n2) points specified by (s), sorts the data lines in the ascending order based on the group data in the column number (n3), and stores the result in the data table (sorting result) of 16-bit binary data having ((n1)×(n2)) points specified by (d). | Page 627 |
| DSORTTBL2<br>DSORTTBL2_U | In the data table (sorting source) of 32-bit binary data having (n1×n2) points specified by (s), sorts the data lines in the ascending order based on the group data in the column number (n3), and stores the result in the data table (sorting result) of 32-bit binary data having ((n1)×(n2)) points specified by (d). | Page 630 |

### ■Adding 16-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| WSUM<br>WSUM_U<br>WSUMP<br>WSUMP_U | These instructions add the (n) points of 16-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d). | Page 633 |

### ■Adding 32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| DWSUM<br>DWSUM_U<br>DWSUMP<br>DWSUMP_U | These instructions add the (n) points of 32-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d). | Page 634 |

### ■Calculating the mean value of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| MEAN<br>MEANP<br>MEAN_U<br>MEANP_U | These instructions calculate the mean value of (n) points (16-bit binary data) in the devices starting from the one specified by (s), and store the result in the device specified by (d). | Page 636 |
| DMEAN<br>DMEANP<br>DMEAN_U<br>DMEANP_U | These instructions calculate the mean value of (n) points (32-bit binary data) in the devices starting from the one specified by (s), and store the result in the device specified by (d). | Page 638 |

### ■Calculating the square root of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| SQRT<br>SQRTP | $\sqrt{(s)} \rightarrow (d)$ | Page 640 |
| DSQRT<br>DSQRTP | $\sqrt{(s)+1,(s)} \rightarrow (d)+1,(d)$ | Page 641 |

### ■CRC calculation

| Instruction symbol | Description | Reference |
|---|---|---|
| CRC<br>CRCP | This instruction generates a CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and stores the CRC value to (d). | Page 642 |

## File register operation instruction

### ■Reading the indirect address

| Instruction symbol | Description | Reference |
|---|---|---|
| ADRSET<br>ADRSETP | (s) → (d)<br>↑ Indirect address of the specified device<br>Device name | Page 645 |

## Clock instruction

### ■Reading clock data

| Instruction symbol | Description | Reference |
|---|---|---|
| TRD<br>TRDP | (Clock element) → (d)+0 Year<br>+1 Month<br>+2 Day<br>+3 Hour<br>+4 Minute<br>+5 Seconds<br>+6 Day of week | Page 647 |

### ■Writing clock data

| Instruction symbol | Description | Reference |
|---|---|---|
| TWR<br>TWRP | (d)+0 Year<br>+1 Month<br>+2 Day<br>+3 Hour<br>+4 Minute<br>+5 Seconds<br>+6 Day of week → (Clock element) | Page 649 |

### ■Adding clock data

| Instruction symbol | Description | Reference |
|---|---|---|
| TADD<br>TADDP | (s1) Hour, Minute, Seconds + (s2) Hour, Minute, Seconds → (d) Hour, Minute, Seconds | Page 651 |

### ■Subtracting clock data

| Instruction symbol | Description | Reference |
|---|---|---|
| TSUB<br>TSUBP | (s1) Hour, Minute, Seconds - (s2) Hour, Minute, Seconds → (d) Hour, Minute, Seconds | Page 653 |

### ■Converting time data from hour/minute/second to seconds in 16 bits/32 bits

| Instruction symbol | Description | Reference |
|---|---|---|
| HTOS<br>HTOSP | (s) Hour, Minute, Seconds → (d) Seconds | Page 655 |
| DHTOS<br>DHTOSP | (s) Hour, Minute, Seconds → (d)+1, (d) Seconds | Page 657 |

### ■Converting time data from seconds to hour/minute/second in 16 bits/32 bits

| Instruction symbol | Description | Reference |
|---|---|---|
| STOH<br>STOHP | (s) Seconds → (d) Hour, Minute, Seconds | Page 659 |
| DSTOH<br>DSTOHP | (s)+1, (s) Seconds → (d) Hour, Minute, Seconds | Page 660 |

## ■Comparing date data

| Instruction symbol | Description | Reference |
|---|---|---|
| LDDT=, ANDDT=, ORDT= | (s1) Year, (s1)+1 Month, (s1)+2 Day = (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | Page 661 |
| LDDT<>, ANDDT<>, ORDT<> | (s1) Year, (s1)+1 Month, (s1)+2 Day < > (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | |
| LDDT>, ANDDT>, ORDT> | (s1) Year, (s1)+1 Month, (s1)+2 Day > (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | |
| LDDT<=, ANDDT<=, ORDT<= | (s1) Year, (s1)+1 Month, (s1)+2 Day <= (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | |
| LDDT<, ANDDT<, ORDT< | (s1) Year, (s1)+1 Month, (s1)+2 Day < (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | |
| LDDT>=, ANDDT>=, ORDT>= | (s1) Year, (s1)+1 Month, (s1)+2 Day >= (s2) Year, (s2)+1 Month, (s2)+2 Day → Result | |

## ■Comparing time data

| Instruction symbol | Description | Reference |
|---|---|---|
| LDTM=, ANDTM=, ORTM= | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds = (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | Page 664 |
| LDTM<>, ANDTM<>, ORTM<> | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds < > (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | |
| LDTM>, ANDTM>, ORTM> | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds > (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | |
| LDTM<=, ANDTM<=, ORTM<= | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds <= (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | |
| LDTM<, ANDTM<, ORTM< | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds < (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | |
| LDTM>=, ANDTM>=, ORTM>= | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds >= (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds → Result | |

## ■Comparing clock data

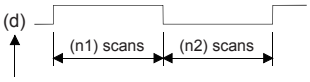
| Instruction symbol | Description | Reference |
|---|---|---|
| TCMP<br>TCMPP | (s1) Hour, (s2) Minute, (s3) Seconds > (s4) Hour, (s4)+1 Minute, (s4)+2 Seconds ⇨ (d) = ON<br>(s1) Hour, (s2) Minute, (s3) Seconds = (s4) Hour, (s4)+1 Minute, (s4)+2 Seconds ⇨ (d)+1 = ON<br>(s1) Hour, (s2) Minute, (s3) Seconds < (s4) Hour, (s4)+1 Minute, (s4)+2 Seconds ⇨ (d)+2 = ON | Page 667 |

### ■Comparing clock data zones

| Instruction symbol | Description | Reference |
|---|---|---|
| TZCP | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds > (s3) Hour, (s3)+1 Minute, (s3)+2 Seconds ⇨ (d) = ON | Page 669 |
| TZCPP | (s1) Hour, (s1)+1 Minute, (s1)+2 Seconds ≤ (s3) Hour, (s3)+1 Minute, (s3)+2 Seconds ≤ (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds ⇨ (d)+1 = ON<br>(s3) Hour, (s3)+1 Minute, (s3)+2 Seconds > (s2) Hour, (s2)+1 Minute, (s2)+2 Seconds ⇨ (d)+2 = ON | |

## Timing check instruction

### ■Generating timing pulses

| Instruction symbol | Description | Reference |
|---|---|---|
| DUTY |  SM420 to SM424, SM2330 to SM2334 | Page 671 |

### ■Hour meter

| Instruction symbol | Description | Reference |
|---|---|---|
| HOURM | This instruction adds the time during which the input contact is ON in units of 1 hour, turns ON the device specified by (d2) when the total ON time exceeds the time specified by (s) (16-bit binary data), and stores the current value in units of 1 hour (16-bit binary data) to (d1), and the current value that is less than one hour (16-bit binary data) to (d1)+1 in units of seconds. | Page 673 |
| DHOURM | This instruction adds the time during which the input contact is ON in units of 1 hour, turns ON the device specified by (d2) when the total ON time exceeds the time specified by (s) (32-bit binary data), and stores the current value in units of 1 hour (32-bit binary data) to (d1), and the current value that is less than one hour (16-bit binary data) to (d1)+2 in units of seconds. | Page 675 |

## Module access instruction

### ■Performing I/O refresh

| Instruction symbol | Description | Reference |
|---|---|---|
| REF | This instruction refreshes the relevant I/O area during a scan. | Page 676 |
| REFP | | |
| RFS | | |
| RFSP | | |

### ■Reading 1-word/2-word data from another module (16-bit specification)

| Instruction symbol | Description | Reference |
|---|---|---|
| FROM | These instructions read the (n) word data from the buffer memory of the intelligent function module. | Page 678 |
| FROMP | | |
| DFROM | These instructions read the (n)×2 word data from the buffer memory of the intelligent function module. | |
| DFROMP | | |

### ■Writing 1-word/2-word data to another module (16-bit specification)

| Instruction symbol | Description | Reference |
|---|---|---|
| TO | These instructions write the (n) word data to the buffer memory of the intelligent function module. | Page 681 |
| TOP | | |
| DTO | These instructions write the (n)×2 word data to the buffer memory of the intelligent function module. | |
| DTOP | | |

### ■Reading 1-word/2-word data from another module (32-bit specification)

| Instruction symbol | Description | Reference |
|---|---|---|
| FROMD | These instructions read the (n) word data from the buffer memory of the intelligent function module. | Page 684 |
| FROMDP | | |
| DFROMD | These instructions read the (n)×2 word data from the buffer memory of the intelligent function module. | |
| DFROMDP | | |

### ■Writing 1-word/2-word data to another module (32-bit specification)

| Instruction symbol | Description | Reference |
|---|---|---|
| TOD | These instructions write the (n) word data to the buffer memory of the intelligent function module. | Page 687 |
| TODP | | |
| DTOD | These instructions write the (n)×2 word data to the buffer memory of the intelligent function module. | |
| DTODP | | |

# 3.4 Step Ladder Instructions

## Starts/Ends step ladder

| Instruction symbol | Description | Reference |
|---|---|---|
| STL | Start of step ladder | Page 690 |
| RETSTL | End of step ladder | |

# 3.5 Built-in Ethernet Function Instruction

## Open/Close Processing Instructions

### ■Opening a connection

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.SOCOPEN | This instruction opens the connection specified by (s1). | Page 693 |

### ■Closing a connection

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.SOCCLOSE | This instruction closes the connection specified by (s1). (Closing a connection) | Page 696 |

## Socket communication function instruction

### ■Reading receive data during the END processing

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.SOCRCV | This instruction reads the received data of the connection specified by (s1) from the socket communication receive data area, during the END processing. | Page 698 |

### ■Sending data

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.SOCSND | This instruction sends the data set in (s3) to the target device of the connection specified by (s1). | Page 701 |

### ■Reading connection information

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.SOCCINF | This instruction reads the connection information of the connection specified by (s1). | Page 704 |

### ■Reading socket communication receive data

| Instruction symbol | Description | Reference |
|---|---|---|
| S.SOCRDATA | This instruction reads the data of the number of words specified in (n) from the socket communication receive data area of the connection specified by (s1), and stores it to the device specified by (d) onwards. | Page 706 |
| SP.SOCRDATA | | |

## Predefined Protocol Support Function Instruction

### ■Executing the protocols registered for the predefined protocol support function

| Instruction symbol | Description | Reference |
|---|---|---|
| SP.ECPRTCL | Executes the protocol specified by the communication protocol support tool of the engineering tool. | Page 708 |

# 3.6 PID Control Instruction



## PID control loop

| Instruction symbol | Description | Reference |
|---|---|---|
| PID | This instruction executes PID control which changes the output value according to the input variation. | Page 712 |

# 4 MODULE SPECIFIC INSTRUCTION

## 4.1 High-speed Counter Instruction

### High-speed processing instruction

#### ■Setting 32-bit data comparison

| Instruction symbol | Description | Reference |
|---|---|---|
| DHSCS | Turns ON the bit device of (d) when the current value of the high-speed counter of CH specified by (s2) is changed to the value specified by (s1). | Page 716 |

#### ■Reset 32-bit data comparison

| Instruction symbol | Description | Reference |
|---|---|---|
| DHSCR | Turns OFF the bit device of (d) when the current value of the high-speed counter of CH specified by (s2) is changed to the value specified by (s1). | Page 718 |

#### ■Comparison of 32-bit data band

| Instruction symbol | Description | Reference |
|---|---|---|
| DHSZ | Compares whether the current value of the high-speed counter is within or outside the value range specified by (s1) or (s2). | Page 720 |

#### ■Start/stop of the 16-bit/32-bit data high-speed I/O function

| Instruction symbol | Description | Reference |
|---|---|---|
| HIOEN | Start or stop high-speed I/O for the specified CH. | Page 722 |
| HIOENP | | |
| DHIOEN | | Page 725 |
| DHIOENP | | |

### High-speed current value transfer instruction

#### ■High-speed current value transfer of 16-bit/32-bit data

| Instruction symbol | Description | Reference |
|---|---|---|
| HCMOV | Transfers the current value of the high-speed I/O. | Page 728 |
| HCMOVP | | |
| DHCMOV | | Page 730 |
| DHCMOVP | | |

# 4.2 External Device Communication Instruction

## Serial communication 2

| Instruction symbol | Description | Reference |
|---|---|---|
| RS2 | Sends/receives data by non-protocol communication. | Page 732 |

## Inverter Communication Instruction

### ■Inverter operation monitoring (Status check)

| Instruction symbol | Description | Reference |
|---|---|---|
| IVCK | Reads the contents of the corresponding instruction code from the specified inverter station number. | Page 734 |

### ■Inverter operations control (Drive)

| Instruction symbol | Description | Reference |
|---|---|---|
| IVDR | Writes the contents of the corresponding instruction code to the specified inverter station number. | Page 736 |

### ■Inverter parameter read

| Instruction symbol | Description | Reference |
|---|---|---|
| IVRD | Reads a parameter from the specified inverter station number. | Page 738 |

### ■Inverter parameter write

| Instruction symbol | Description | Reference |
|---|---|---|
| IVWR | Writes a parameter to the specified inverter station number. | Page 740 |

### ■Inverter parameter block write

| Instruction symbol | Description | Reference |
|---|---|---|
| IVBWR | Writes the range of the specified data tables to the specified inverter station number in batch. | Page 742 |

### ■Inverter multi command

| Instruction symbol | Description | Reference |
|---|---|---|
| IVMC | Sends/receives data corresponding to the send/receive data type to/from the specified inverter station number. | Page 744 |

## MODBUS Communication Instruction

| Instruction symbol | Description | Reference |
|---|---|---|
| ADPRW | Sends the function code from the master to the slave of the MODBUS serial communication and reads or writes the data. | Page 746 |

## Predefined Protocol Support Function Instruction

| Instruction symbol | Description | Reference |
|---|---|---|
| S.CPRTCL | Executes the protocol specified by the communication protocol support tool of the engineering tool. | Page 748 |
| SP.CPRTCL | | |

# 4.3 Positioning Instruction

## Positioning instruction

### ■Zero return(OPR) with 16-bit/32- bit data DOG search

| Instruction symbol | Description | Reference |
|---|---|---|
| DSZR | • When FX3 compatible operand is specified<br>Specifies the proximity dog signal, zero signal and device (Y). Outputs a pulse with the specified device (Y) to perform the zero return operation.<br>• When FX5 operand is specified<br>Specifies the original position return speed, creep speed and axis number. Outputs a pulse with the specified axis to perform the zero return operation. | Page 752 |
| DDSZR | | Page 755 |

### ■16-bit/32-bit data interrupt positioning

| Instruction symbol | Description | Reference |
|---|---|---|
| DVIT | • When FX3 compatible operand is specified<br>Performs interrupt positioning with the specified travel distance, speed, and device (Y).<br>• When FX5 operand is specified<br>Performs interrupt positioning with the specified travel distance, speed, and axis number. | Page 756 |
| DDVIT | | Page 758 |

### ■Positioning by one table operation

| Instruction symbol | Description | Reference |
|---|---|---|
| TBL | • When FX3 compatible operand is specified<br>Outputs 1 table operation from the table set by the parameter as pulse with specified device (Y).<br>• When FX5 operand is specified<br>Outputs 1 table operation from the table set by the parameter as pulse with specified axis number. | Page 760 |

### ■Positioning by multiple table operation

| Instruction symbol | Description | Reference |
|---|---|---|
| DRVTBL | Outputs continuous multiple table operations from the table set by the parameter as pulse with specified axis number. | Page 762 |

### ■Multiple axes concurrent drive positioning

| Instruction symbol | Description | Reference |
|---|---|---|
| DRVMUL | Outputs the table set by the parameter as pulse with specified multiple axes. | Page 764 |

### ■32-bit data ABS current value read

| Instruction symbol | Description | Reference |
|---|---|---|
| DABS | Reads the absolute position data of the servo amplifier. | Page 766 |

### ■16-bit/32-bit data variable speed pulse

| Instruction symbol | Description | Reference |
|---|---|---|
| PLSV | • When FX3 compatible operand is specified<br>Specifies the command speed and output device (Y) and uses the specified device (Y) to perform pulse output.<br>• When FX5 operand is specified<br>Specifies the command speed and performs pulse output with the specified axis number. | Page 767 |
| DPLSV | | Page 769 |

### ■16-bit/32-bit data relative positioning

| Instruction symbol | Description | Reference |
|---|---|---|
| DRVI | • When FX3 compatible operand is specified<br>Specifies the travel distance from the current position, speed and performs pulse output with the specified device (Y).<br>• When FX5 operand is specified<br>Specifies the travel distance from the current position, speed and performs pulse output with the specified axis number. | Page 771 |
| DDRVI | | Page 773 |

### ■16-bit/32-bit data absolute positioning

| Instruction symbol | Description | Reference |
|---|---|---|
| DRVA | • When FX3 compatible operand is specified<br>Specifies the travel distance from the reference position, speed and performs pulse output with the specified device (Y).<br>• When FX5 operand is specified<br>Specifies the travel distance from the reference position, speed and performs pulse output with the specified axis number. | Page 775 |
| DDRVA | | Page 777 |

# 4.4 BFM Device Read/ Write Instruction



## Divided BFM Read

| Instruction symbol | Description | Reference |
|---|---|---|
| RBFM | Divides and reads data from the continuous buffer memory in the intelligent module.<br>(This instruction cannot be used with the FX5 intelligent module.) | Page 779 |

## Divided BFM Write

| Instruction symbol | Description | Reference |
|---|---|---|
| WBFM | Divides and writes data to the continuous buffer memory in the intelligent module.<br>(This instruction cannot be used with the FX5 intelligent module.) | Page 782 |

# 5 STANDARD FUNCTIONS/FUNCTION BLOCKS

## 5.1 Standard Functions

### Type conversion functions

#### Converting BOOL to WORD/DWORD

| Function symbol | Description | Reference |
|---|---|---|
| BOOL_TO_WORD | Converts BOOL type data to WORD type data. | Page 786 |
| BOOL_TO_WORD_E | | |
| BOOL_TO_DWORD | Converts BOOL type data to DWORD type data. | Page 787 |
| BOOL_TO_DWORD_E | | |

#### Converting BOOL to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| BOOL_TO_INT | Converts BOOL type data to INT type data. | Page 788 |
| BOOL_TO_INT_E | | |
| BOOL_TO_DINT | Converts BOOL type data to DINT type data. | Page 789 |
| BOOL_TO_DINT_E | | |

#### Converting BOOL to TIME

| Function symbol | Description | Reference |
|---|---|---|
| BOOL_TO_TIME | Converts BOOL type data to TIME type data. | Page 790 |
| BOOL_TO_TIME_E | | |

#### Converting BOOL to STRING

| Function symbol | Description | Reference |
|---|---|---|
| BOOL_TO_STRING | Converts BOOL type data to STRING type data. | Page 791 |
| BOOL_TO_STRING_E | | |

#### Converting WORD to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| WORD_TO_BOOL | Converts WORD type data to BOOL type data. | Page 792 |
| WORD_TO_BOOL_E | | |

#### Converting WORD to DWORD

| Function symbol | Description | Reference |
|---|---|---|
| WORD_TO_DWORD | Converts WORD type data to DWORD type data. | Page 793 |
| WORD_TO_DWORD_E | | |

## Converting WORD to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| WORD_TO_INT | Converts WORD type data to INT type data. | Page 794 |
| WORD_TO_INT_E | | |
| WORD_TO_DINT | Converts WORD type data to DINT type data. | Page 795 |
| WORD_TO_DINT_E | | |

## Converting WORD to TIME

| Function symbol | Description | Reference |
|---|---|---|
| WORD_TO_TIME | Converts WORD type data to TIME type data. | Page 796 |
| WORD_TO_TIME_E | | |

## Converting DWORD to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| DWORD_TO_BOOL | Converts DWORD type data to BOOL type data. | Page 797 |
| DWORD_TO_BOOL_E | | |

## Converting DWORD to WORD

| Function symbol | Description | Reference |
|---|---|---|
| DWORD_TO_WORD | Converts DWORD type data to WORD type data. | Page 798 |
| DWORD_TO_WORD_E | | |

## Converting DWORD to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| DWORD_TO_INT | Converts DWORD type data to INT type data. | Page 800 |
| DWORD_TO_INT_E | | |
| DWORD_TO_DINT | Converts DWORD type data to DINT type data. | Page 802 |
| DWORD_TO_DINT_E | | |

## Converting DWORD to TIME

| Function symbol | Description | Reference |
|---|---|---|
| DWORD_TO_TIME | Converts DWORD type data to TIME type data. | Page 803 |
| DWORD_TO_TIME_E | | |

## Converting INT to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_BOOL | Converts INT type data to BOOL type data. | Page 804 |
| INT_TO_BOOL_E | | |

## Converting INT to WORD/DWORD

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_WORD | Converts INT type data to WORD type data. | Page 805 |
| INT_TO_WORD_E | | |
| INT_TO_DWORD | Converts INT type data to DWORD type data. | Page 806 |
| INT_TO_DWORD_E | | |

## Converting INT to DINT

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_DINT | Converts INT type data to DINT type data. | Page 807 |
| INT_TO_DINT_E | | |

## Converting INT to BCD

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_BCD | Converts INT type data to BCD type data. | Page 808 |
| INT_TO_BCD_E | | |

## Converting INT to REAL

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_REAL | Converts INT type data to REAL type data. | Page 810 |
| INT_TO_REAL_E | | |

## Converting INT to TIME

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_TIME | Converts INT type data to TIME type data. | Page 811 |
| INT_TO_TIME_E | | |

## Converting INT to STRING

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_STRING | Converts INT type data to STRING type data. | Page 812 |
| INT_TO_STRING_E | | |

## Converting DINT to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_BOOL | Converts DINT type data to BOOL type data. | Page 814 |
| DINT_TO_BOOL_E | | |

## Converting DINT to WORD/DWORD

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_WORD | Converts DINT type data to WORD type data. | Page 815 |
| DINT_TO_WORD_E | | |
| DINT_TO_DWORD | Converts DINT type data to DWORD type data. | Page 817 |
| DINT_TO_DWORD_E | | |

## Converting DINT to INT

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_INT | Converts DINT type data to INT type data. | Page 818 |
| DINT_TO_INT_E | | |

## Converting DINT to BCD

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_BCD | Converts DINT type data to BCD type data. | Page 819 |
| DINT_TO_BCD_E | | |

## Converting DINT to REAL

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_REAL | Converts DINT type data to REAL type data. | Page 821 |
| DINT_TO_REAL_E | | |

## Converting DINT to TIME

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_TIME | Converts DINT type data to TIME type data. | Page 822 |
| DINT_TO_TIME_E | | |

## Converting DINT to STRING

| Function symbol | Description | Reference |
|---|---|---|
| DINT_TO_STRING | Converts DINT type data to STRING type data. | Page 823 |
| DINT_TO_STRING_E | | |

## Converting BCD to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| BCD_TO_INT | Converts BCD type data to INT type data. | Page 825 |
| BCD_TO_INT_E | | |
| BCD_TO_DINT | Converts BCD type data to DINT type data. | Page 827 |
| BCD_TO_DINT_E | | |

## Converting REAL to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| REAL_TO_INT | Converts REAL type data to INT type data. | Page 829 |
| REAL_TO_INT_E | | |
| REAL_TO_DINT | Converts REAL type data to DINT type data. | Page 831 |
| REAL_TO_DINT_E | | |

## Converting REAL to STRING

| Function symbol | Description | Reference |
|---|---|---|
| REAL_TO_STRING | Converts REAL type data to STRING type data (exponent format). | Page 833 |
| REAL_TO_STRING_E | | |

## Converting TIME to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| TIME_TO_BOOL | Converts TIME type data to BOOL type data. | Page 835 |
| TIME_TO_BOOL_E | | |

## Converting TIME to WORD/DWORD

| Function symbol | Description | Reference |
|---|---|---|
| TIME_TO_WORD | Converts TIME type data to WORD type data. | Page 836 |
| TIME_TO_WORD_E | | |
| TIME_TO_DWORD | Converts TIME type data to DWORD type data. | Page 837 |
| TIME_TO_DWORD_E | | |

## Converting TIME to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| TIME_TO_INT | Converts TIME type data to INT type data. | Page 838 |
| TIME_TO_INT_E | | |
| TIME_TO_DINT | Converts TIME type data to DINT type data. | Page 839 |
| TIME_TO_DINT_E | | |

## Converting TIME to STRING

| Function symbol | Description | Reference |
|---|---|---|
| TIME_TO_STRING | Converts TIME type data to STRING type data. | Page 840 |
| TIME_TO_STRING_E | | |

## Converting STRING to BOOL

| Function symbol | Description | Reference |
|---|---|---|
| STRING_TO_ BOOL | Converts STRING type data to BOOL type data. | Page 841 |
| STRING_TO_ BOOL_E | | |

## Converting STRING to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| STRING_TO_INT | Converts STRING type data to INT type data. | Page 842 |
| STRING_TO_INT_E | | |
| STRING_TO_DINT | Converts STRING type data to DINT type data. | Page 844 |
| STRING_TO_DINT_E | | |

## Converting STRING to REAL

| Function symbol | Description | Reference |
|---|---|---|
| STRING_TO_REAL | Converts STRING type data to REAL type data. | Page 846 |
| STRING_TO_REAL_E | | |

## Converting STRING to TIME

| Function symbol | Description | Reference |
|---|---|---|
| STRING_TO_TIME | Converts STRING type data to TIME type data. | Page 849 |
| STRING_TO_TIME_E | | |

## Converting bit array to INT/DINT

| Function symbol | Description | Reference |
|---|---|---|
| BITARR_TO_INT | Converts a bit array to INT type data for a specified number of bits. | Page 850 |
| BITARR_TO_INT_E | | |
| BITARR_TO_DINT | Converts a bit array to DINT type data for a specified number of bits. | Page 851 |
| BITARR_TO_DINT_E | | |

## Converting INT/DINT to bit array

| Function symbol | Description | Reference |
|---|---|---|
| INT_TO_BITARR | Outputs low-order (n) bits of INT type data to a bit array. | Page 852 |
| INT_TO_BITARR_E | | |
| DINT_TO_BITARR | Outputs low-order (n) bits of DINT type data to a bit array. | Page 853 |
| DINT_TO_BITARR_E | | |

## Bit array copy

| Function symbol | Description | Reference |
|---|---|---|
| CPY_BITARR | Copies specified number of bits of a bit array. | Page 854 |
| CPY_BITARR_E | | |

## Reading the specified bit of word label

| Function symbol | Description | Reference |
|---|---|---|
| GET_BIT_OF_INT | Reads a value of a specified bit of INT type data. | Page 855 |
| GET_BIT_OF_INT_E | | |

## Writing the specified bit of word label

| Function symbol | Description | Reference |
|---|---|---|
| SET_BIT_OF_INT | Writes a value to a specified bit of INT type data. | Page 856 |
| SET_BIT_OF_INT_E | | |

## Copying the specified bit of word label

| Function symbol | Description | Reference |
|---|---|---|
| CPY_BIT_OF_INT | Copies a specified bit of INT type data to a specified bit of another INT type data. | Page 857 |
| CPY_BIT_OF_INT_E | | |

## Unnecessary of type conversion

| Function symbol | Description | Reference |
|---|---|---|
| GET_BOOL_ADDR | Converts a data type to the BOOL type. | Page 858 |
| GET_INT_ADDR | Converts a data type to the INT type. | |
| GET_WORD_ADDR | Converts a data type to the WORD type. | |

# Standard functions of one numeric variable

## Absolute value

| Function symbol | Description | Reference |
|---|---|---|
| ABS | Outputs the absolute value of an input value. | Page 859 |
| ABS_E | | |

## Square root

| Function symbol | Description | Reference |
|---|---|---|
| SQRT | Outputs the square root of an input value. | Page 861 |
| SQRT_E | | |

## Natural logarithm operation

| Function symbol | Description | Reference |
|---|---|---|
| LN | Outputs the natural logarithm operation result of an input value. | Page 862 |
| LN_E | | |

## Calculating the common logarithm

| Function symbol | Description | Reference |
|---|---|---|
| LOG | Outputs the operation result of the common logarithm (the logarithm whose base is 10) of an input value. | Page 863 |
| LOG_E | | |

## Exponential operation

| Function symbol | Description | Reference |
|---|---|---|
| EXP | Outputs the exponential operation result of an input value. | Page 865 |
| EXP_E | | |

## Sine operation

| Function symbol | Description | Reference |
|---|---|---|
| SIN | Outputs the sine of the angle of an input value. | Page 866 |
| SIN_E | | |

## Cosine operation

| Function symbol | Description | Reference |
|---|---|---|
| COS | Outputs the cosine of the angle of an input value. | Page 867 |
| COS_E | | |

## Tangent operation

| Function symbol | Description | Reference |
|---|---|---|
| TAN | Outputs the tangent of the angle value of an input value. | Page 868 |
| TAN_E | | |

## Arc sine operation

| Function symbol | Description | Reference |
|---|---|---|
| ASIN | Outputs the arc sine value of an input value. | Page 869 |
| ASIN_E | | |

### Arc cosine operation

| Function symbol | Description | Reference |
|---|---|---|
| ACOS | Outputs the arc cosine value of an input value. | Page 870 |
| ACOS_E | | |

### Arc tangent operation

| Function symbol | Description | Reference |
|---|---|---|
| ATAN | Outputs the arc tangent value of an input value. | Page 871 |
| ATAN_E | | |

## Standard arithmetic functions

### Addition

| Function symbol | Description | Reference |
|---|---|---|
| ADD | Outputs the sum of input values ((s1) + (s2) + ... + (s28)). | Page 872 |
| ADD_E | | |

### Multiplication

| Function symbol | Description | Reference |
|---|---|---|
| MUL | Outputs the product of input values ((s1) × (s2) × ... × (s28)). | Page 874 |
| MUL_E | | |

### Subtraction

| Function symbol | Description | Reference |
|---|---|---|
| SUB | Outputs the difference of input values ((s1) - (s2)). | Page 876 |
| SUB_E | | |

### Division

| Function symbol | Description | Reference |
|---|---|---|
| DIV | Outputs the quotient of input values ((s1) ÷ (s2)). | Page 878 |
| DIV_E | | |

### Remainder

| Function symbol | Description | Reference |
|---|---|---|
| MOD | Outputs the remainder of input values ((s1) ÷ (s2)). | Page 880 |
| MOD_E | | |

### Exponentiation

| Function symbol | Description | Reference |
|---|---|---|
| EXPT | Outputs the exponentiation of an input value. | Page 882 |
| EXPT_E | | |

### Move operation

| Function symbol | Description | Reference |
|---|---|---|
| MOVE | Assigns an input value to (d). | Page 884 |
| MOVE_E | | |

# Standard bit shift functions

## Shifting n-bit data to left/right

| Function symbol | Description | Reference |
|---|---|---|
| SHL | Shifts an input value leftward by (n) bits and outputs the result. | Page 886 |
| SHL_E | | |
| SHR | Shifts an input value rightward by (n) bits and outputs the result. | Page 888 |
| SHR_E | | |

## Rotating n-bit data to left/right

| Function symbol | Description | Reference |
|---|---|---|
| ROL | Rotates an input value leftward by (n) bits and outputs the result. | Page 890 |
| ROL_E | | |
| ROR | Rotates an input value rightward by (n) bits and outputs the result. | Page 892 |
| ROR_E | | |

# Standard bitwise boolean functions

## AND operation, OR operation, XOR operation, NOT operation

| Function symbol | Description | Reference |
|---|---|---|
| AND | Outputs the logical product of input values. | Page 894 |
| AND_E | | |
| OR | Outputs the logical sum of input values. | |
| OR_E | | |
| XOR | Outputs the exclusive logical sum of input values. | |
| XOR_E | | |
| NOT | Outputs the logical negation of input values. | Page 896 |
| NOT_E | | |

# Standard selection functions

## Selection

| Function symbol | Description | Reference |
|---|---|---|
| SEL | Outputs a selected input value. | Page 897 |
| SEL_E | | |

## Selecting Maximum/Minimum Value

| Function symbol | Description | Reference |
|---|---|---|
| MAX | Outputs the maximum value of an input value. | Page 899 |
| MAX_E | | |
| MIN | Outputs the minimum value of an input value. | |
| MIN_E | | |

## Limit Control

| Function symbol | Description | Reference |
|---|---|---|
| LIMIT | Outputs an input value controlled with the upper and lower limits. | Page 901 |
| LIMIT_E | | |

### Multiplexer

| Function symbol | Description | Reference |
|---|---|---|
| MUX | Outputs one of multiple input values. | Page 903 |
| MUX_E | | |

## Standard comparison functions

### Compare

| Function symbol | Description | Reference |
|---|---|---|
| GT | Outputs the data comparison result of input values. | Page 905 |
| GT_E | | |
| GE | | |
| GE_E | | |
| EQ | | |
| EQ_E | | |
| LE | | |
| LE_E | | |
| LT | | |
| LT_E | | |
| NE | | Page 907 |
| NE_E | | |

## Standard character string functions

### Character string length detection

| Function symbol | Description | Reference |
|---|---|---|
| LEN | Detects the length of an input character string and outputs the result. | Page 909 |
| LEN_E | | |

### Extracting character string data from the left/right

| Function symbol | Description | Reference |
|---|---|---|
| LEFT | Outputs specified number of characters from the left of input character string data. | Page 910 |
| LEFT_E | | |
| RIGHT | Outputs specified number of characters from the right of input character string data. | |
| RIGHT_E | | |

### Extract mid string

| Function symbol | Description | Reference |
|---|---|---|
| MID | Outputs specified number of characters from an arbitrary position of an input character string. | Page 912 |
| MID_E | | |

### String concatenation

| Function symbol | Description | Reference |
|---|---|---|
| CONCAT | Concatenates character strings and output the result. | Page 914 |
| CONCAT_E | | |

### Inserting character string

| Function symbol | Description | Reference |
|---|---|---|
| INSERT | Inserts a character string into another character string and output the result. | Page 916 |
| INSERT_E | | |

### Deleting character string

| Function symbol | Description | Reference |
|---|---|---|
| DELETE | Deletes an arbitrary range of a character string and outputs the result. | Page 918 |
| DELETE_E | | |

### Replacing character string

| Function symbol | Description | Reference |
|---|---|---|
| REPLACE | Replaces an arbitrary range of a character string and outputs the result. | Page 920 |
| REPLACE_E | | |

### Searching character string

| Function symbol | Description | Reference |
|---|---|---|
| FIND | Searches for a character string and outputs the result. | Page 923 |
| FIND_E | | |

## Time data functions

### Addition

| Function symbol | Description | Reference |
|---|---|---|
| ADD_TIME | Outputs the sum of input values (time data) ((s1) + (s2)). | Page 925 |
| ADD_TIME_E | | |

### Subtraction

| Function symbol | Description | Reference |
|---|---|---|
| SUB_TIME | Outputs the difference of input values (time data) ((s1) - (s2)). | Page 927 |
| SUB_TIME_E | | |

### Multiplication

| Function symbol | Description | Reference |
|---|---|---|
| MUL_TIME | Outputs the product of input values (time data) ((s1) × (s2)). | Page 929 |
| MUL_TIME_E | | |

### Division

| Function symbol | Description | Reference |
|---|---|---|
| DIV_TIME | Outputs the quotient of input values (time data) ((s1) ÷ (s2)). | Page 931 |
| DIV_TIME_E | | |

# 5.2 Standard Function Blocks

## Bistable function blocks

### Bistable function blocks (set priority)

| Function block symbol | Description | Reference |
|---|---|---|
| SR | Judges two input values and outputs 1 (TRUE) or 0 (FALSE). (Set priority) | Page 934 |
| SR_E | | |

### Bistable function blocks (reset priority)

| Function block symbol | Description | Reference |
|---|---|---|
| RS | Judges two input values and outputs 1 (TRUE) or 0 (FALSE). (Reset priority) | Page 936 |
| RS_E | | |

## Edge detection function blocks

### Rising edge detector

| Function block symbol | Description | Reference |
|---|---|---|
| R_TRIG | Detects the rising edge of a signal, and outputs a pulse signal. | Page 938 |
| R_TRIG_E | | |

### Falling edge detector

| Function block symbol | Description | Reference |
|---|---|---|
| F_TRIG | Detects the falling edge of a signal, and outputs a pulse signal. | Page 940 |
| F_TRIG_E | | |

## Counter function blocks

### Up counter

| Function block symbol | Description | Reference |
|---|---|---|
| CTU | Counts up the number of times of rising of a signal. | Page 942 |
| CTU_E | | |

### Down counter

| Function block symbol | Description | Reference |
|---|---|---|
| CTD | Counts down the number of times of rising of a signal. | Page 944 |
| CTD_E | | |

### Up/Down counter

| Function block symbol | Description | Reference |
|---|---|---|
| CTUD | Counts up/down the number of times of rising of a signal. | Page 946 |
| CTUD_E | | |

### Counter function block

| Function block symbol | Description | Reference |
|---|---|---|
| COUNTER_FB_M | Counts up the number of times of rising of a signal from (s3) to (s2). | Page 949 |

# Timer function blocks

## Pulse timer

| Function block symbol | Description | Reference |
|---|---|---|
| TP | Keeps ON a signal for specified duration. | Page 951 |
| TP_E | | |
| TP_10 | | |
| TP_10_E | | |

## On-delay timer

| Function block symbol | Description | Reference |
|---|---|---|
| TON | Turns ON a signal after a specified time. | Page 953 |
| TON_E | | |
| TON_10 | | |
| TON_10_E | | |

## Off-delay timer

| Function block symbol | Description | Reference |
|---|---|---|
| TOF | Turns OFF a signal after a specified time. | Page 955 |
| TOF_E | | |
| TOF_10 | | |
| TOF_10_E | | |

## Timer function blocks

| Function block symbol | Description | Reference |
|---|---|---|
| TIMER_1_FB_M | When the execution condition is established, these function blocks start the timer count to the set time. | Page 957 |
| TIMER_10_FB_M | | |
| TIMER_100_FB_M | | |
| TIMER_CONT_FB_M | | |
| TIMER_CONTHS_FB_M | | |

# PART 3 CPU MODULE INSTRUCTIONS

This part consists of the following chapters.

# 6 SEQUENCE INSTRUCTIONS

## 6.1 Contact Instructions

### Operation start, series connection, parallel connection

#### LD, LDI, AND, ANI, OR, ORI

• LD: NO contact operation start instruction/LDI: NC contact operation start instruction

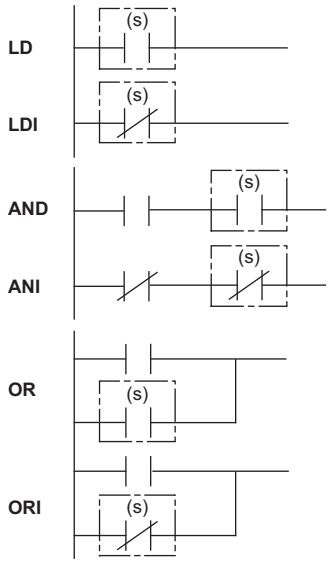
These instructions capture the ON/OFF information of the device specified by (s), and use that as the operation result.

• AND: NO contact series connection instruction/ANI: NC contact series connection instruction

These instructions capture the ON/OFF information of the device specified by (s), AND with the operation result so far, and use the result as the operation result.

• OR: NO contact parallel connection instruction/ORI: NC contact parallel instruction

These instructions capture the ON/OFF information of the device specified by (s), OR with the operation result so far, and use the result as the operation result.

| Ladder diagram | Structured text |
|---|---|
|  | This becomes a assignment statement, operator, control syntax, etc.<br>In the ST language, there are sometimes no instructions (symbols) that directly correspond to contacts such as LD, AND, and OR.<br>When programming using assignment statements, express as shown in the following example.<br>Example<br>Y1:=(X0 OR X1) AND X2 AND NOT X3;<br>Y2:=NOT X4 OR NOT X5; |

| FBD/LD |
|---|
| In the FBD/LD language, contact is used as well as the Ladder diagram. |

Setting data

■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device used as contact | — | Bit | ANY_BOOL |

■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DX) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | ○ |

*1 T, ST, C cannot be used.

## Processing details

### ■LD, LDI

- The LD instruction is the NO contact operation start instruction, and the LDI instruction is the NC contact operation start instruction. These instructions capture the ON/OFF information*1 of the specified device, and use the result as the operation result.

*1 When bits of word devices are specified, devices turn ON/OFF by the 1/0 status of the specified bit.

### ■AND, ANI

- The AND instruction is NO contact series connection instruction and the ANI instruction is NC contact series connection instruction. These instructions capture the ON/OFF information*1 of the specified bit device, AND with the operation result so far, and use the result as the operation result.

*1 When bits of word devices are specified, devices turn ON/OFF by the 1/0 status of the specified bit.

- There is no limitation to the number of series contacts. Any number of contacts can use this instructions consecutively.
- Output to other coils through contacts after the OUT instruction is called cascade output, and these outputs can be repeated any number of times as long as their order is correct.

### ■OR, ORI

- The OR instruction is NO contact parallel connection and the ORI instruction is NC contact parallel connection. These instructions capture the ON/OFF information*1 of the specified device, OR with the operation result so far, and use the result as the operation result.

*1 When bits of word devices are specified, devices turn ON/OFF by the 1/0 status of the specified bit.

- These instructions are connected in parallel from the step with this instruction to the previous step with the LD and LDI instruction.
- There is no limitation in the number of parallel connections.

**Point**

- When bits of word devices are specified, bits are specified in hexadecimal. (For example, b11 of D0 is specified by "D0.B".)

## Operation error

There is no operation error.

# Pulse operation start, pulse series connection, pulse parallel connection

## LDP, LDF, ANDP, ANDF, ORP, ORF

• LDP: Rising edge pulse operation start instruction

This becomes conductive (ON) only at the rising edge (OFF to ON) of the bit device specified by (s).

• LDF: Falling edge pulse operation start instruction

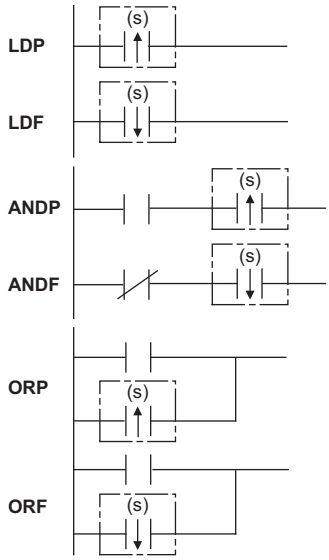
This becomes conductive (ON) only at the falling edge (ON to OFF) of the bit device specified by (s).

• ANDP: Rising edge pulse series connection instruction/ANDF: Falling edge pulse series connection instruction

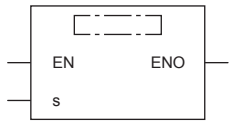
This instruction ANDs the bit device specified by (s) with the operation result so far, and uses the result as the operation result.

• ORP: Rising edge pulse parallel connection/ORF: Falling edge pulse parallel connection

This instruction ORs the bit device specified by (s) with the operation result so far, and uses the result as the operation result.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=LDP(EN,s);<br>ENO:=LDF(EN,s);<br>ENO:=ANDP(EN,s);<br>ENO:=ANDF(EN,s);<br>ENO:=ORP(EN,s);<br>ENO:=ORF(EN,s); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device used as contact | — | Bit | ANY_BOOL |

#### ■Applicable devices

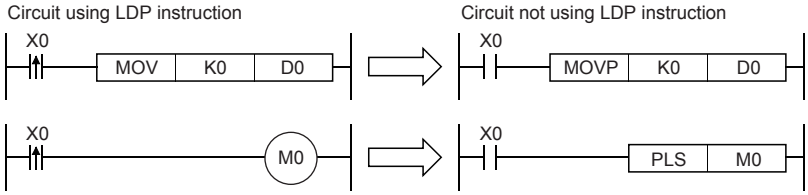
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DX) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | ○ |

*1 T, ST, C cannot be used.

## Processing details

### ■LDP, LDF

- The LDP instruction is the rising edge pulse operation start instruction, and becomes conductive (ON) only at the rising edge (OFF to ON) of the specified bit device. When word devices are specified by bits, this instruction becomes conductive (ON) only when the status of the specified bit changes to 0→1. When only the LDP instruction is programmed, operation is the same as the conversion of the instruction under execution to pulse instruction (□P).




- The LDF instruction is the falling edge pulse operation start instruction, and becomes conductive (ON) at the falling edge (ON to OFF) of the specified bit device. When word devices are specified by bits, this instruction becomes conductive only when the status of the specified bit changes to 1→0.

### ■ANDP, ANDF

- The ANDP instruction is the rising edge pulse series connection instruction, and the ANDF instruction is the falling edge pulse series connection. These instructions AND with the operation result so far, and uses the result as the operation result. The table below shows the ON/OFF information used by these instructions.

| Device specified by ANDP, ANDF | | ANDP status | ANDF status |
|---|---|---|---|
| Bit device | Bit specification of word device | | |
| OFF to ON | 0→1 | ON | OFF |
| OFF | 0 | OFF | OFF |
| ON | 1 | OFF | OFF |
| ON to OFF | 1→0 | OFF | ON |

### ■ORP, ORF

- The ORP instruction is the rising edge pulse parallel connection instruction, and the ORF instruction is the falling edge pulse parallel connection instruction. These instructions OR with the operation result so far, and use the result as the operation result. The table below shows the ON/OFF information used by these instructions.

| Device specified by ORP, ORF | | ORP status | ORF status |
|---|---|---|---|
| Bit device | Bit specification of word device | | |
| OFF to ON | 0→1 | ON | OFF |
| OFF | 0 | OFF | OFF |
| ON | 1 | OFF | OFF |
| ON to OFF | 1→0 | OFF | ON |

## Operation error

There is no operation error.

# Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection

## LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI

• LDPI: Rising edge pulse NOT operation start instruction

This instruction becomes conductive (ON) at OFF, ON and the falling edge (ON to OFF) of the bit device specified by (s).

• LDFI: Falling edge pulse NOT operation start instruction

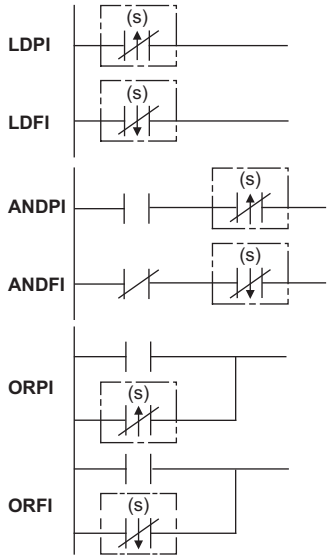
This instruction becomes conductive (ON) at the rising edge (OFF to ON), OFF and ON of the bit device specified by (s).

• ANDPI: Rising edge pulse NOT series connection instruction/ANDFI: Falling edge pulse NOT series connection instruction

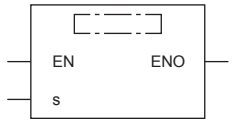
This instruction ANDs the bit devices specified by (s) with the operation result so far, and uses the result as the operation result.

• ORPI: Rising edge pulse NOT parallel connection instruction/ORFI: Falling edge pulse NOT parallel connection instruction

This instruction ORs the bit devices specified by (s) with the operation result so far, and uses the result as the operation result.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=LDPI(EN,s);<br>ENO:=LDFI(EN,s);<br>ENO:=ANDPI(EN,s);<br>ENO:=ANDFI(EN,s);<br>ENO:=ORPI(EN,s);<br>ENO:=ORFI(EN,s); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device used as contact | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DX) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | ○ |

*1 T, ST, C cannot be used.

## Processing details

### ■LDPI, LDFI

- The LDPI instruction is the rising edge pulse NOT operation start instruction, and becomes conductive (ON) at OFF, ON and the falling edge (ON to OFF) of the specified bit device. When word devices are specified by bits, this instruction becomes conductive when the status of the specified bit is 0, 1, and when it changes 1→0.
- The LDFI instruction is the falling edge pulse NOT operation start instruction, and becomes conductive (ON) at the rising edge (OFF to ON), OFF and ON of the specified bit device. When word devices are specified by bits, this instruction becomes conductive (ON) when the status of the specified bit is 0, 1, and when it changes 0→1. The table below shows the ON/OFF information used by these instructions.

| Device specified by LDPI, LDFI | | LDPI status | LDFI status |
|---|---|---|---|
| Bit device | Bit specification of word device | | |
| OFF to ON | 0→1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1→0 | ON | OFF |

### ■ANDPI, ANDFI

- The ANDPI instruction is the rising edge pulse NOT series connection instruction, and the ANDFI instruction is the falling edge pulse NOT series connection instruction. These instructions AND with the operation result so far, and use the result as the operation result. The table below shows the ON/OFF information used by these instructions.

| Device specified by ANDPI, ANDFI | | ANDPI status | ANDFI status |
|---|---|---|---|
| Bit device | Bit specification of word device | | |
| OFF to ON | 0→1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1→0 | ON | OFF |

### ■ORPI, ORFI

- The ORPI instruction is the rising edge pulse NOT parallel connection instruction, and the ORFI instruction is the falling edge pulse NOT parallel connection instruction. These instructions OR with the operation result so far, and use the result as the operation result. The table below shows the ON/OFF information used by these instructions.

| Device specified by ORPI, ORFI | | ORPI status | ORFI status |
|---|---|---|---|
| Bit device | Bit specification of word device | | |
| OFF to ON | 0→1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1→0 | ON | OFF |

## Operation error

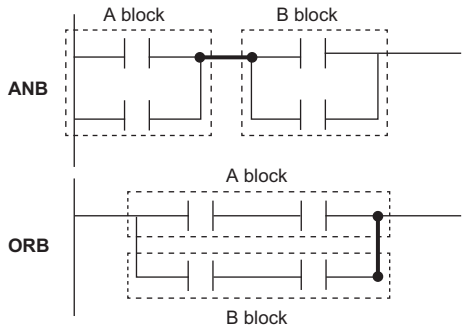
There is no operation error.

# 6.2 Association Instruction

## Ladder block series/parallel connection

### ANB, ORB

These instructions AND or OR the A and B blocks, and use the result as the operation result.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported. |

| FBD/LD |
|---|
| Not supported. |

#### Processing details

**■ANB**

- This instruction ANDs the A and B blocks, and uses the result as the operation result.
- The symbol of this instruction is not NO contact symbol but a connection symbol.

**■ORB**

- This instruction ORs the A and B blocks, and uses the result as the operation result.
- This instruction connects the ladder blocks of two contacts or more in parallel. For parallel connection of only one contact, use the OR and ORI instructions; there is no need to use this instruction.
- The symbol of this instruction is not NO contact symbol but a connection symbol.

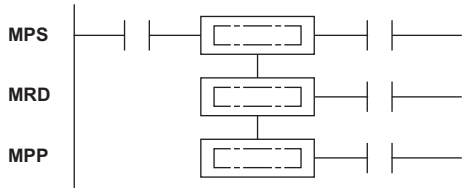
#### Operation error

There is no operation error.

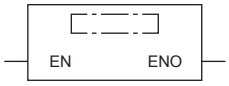
# Storing/reading/clearing the operation result

## MPS, MRD, MPP

- MPS: This instruction stores the preceding operation result (ON/OFF) to memory.
- MRD, MPP: These instructions read the operation result stored by the MPS instruction, and executes operations from the next step using that operation result.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MPS(EN);<br>ENO:=MRD(EN);<br>ENO:=MPP(EN); |

**FBD/LD**






### Processing details

**■MPS**

- This instruction stores the preceding operation result (ON/OFF) to memory.
- This instruction can be used up to 16 times consecutively. When MPP instruction is used in between, the number of uses of MPS instruction is decremented by 1.

**■MRD**

- This instruction reads the operation result stored by the MPS instruction to memory, and executes operations from the next step using that operation result.

**■MPP**

- This instruction reads the operation result stored by the MPS instruction to memory, and executes operations from the next step using that operation result.
- This instruction clears the operation result stored by the MPS instruction from memory.
- The number of uses of MPS instruction is decremented by 1.

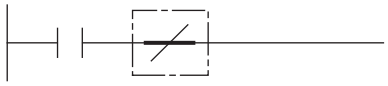
### Operation error

There is no operation error.

# Inverting the operation result

## INV

This instruction inverts the operation result up to this instruction.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=INV(EN); |

**FBD/LD**


### Processing details

- This instruction inverts the operation result up to this instruction.

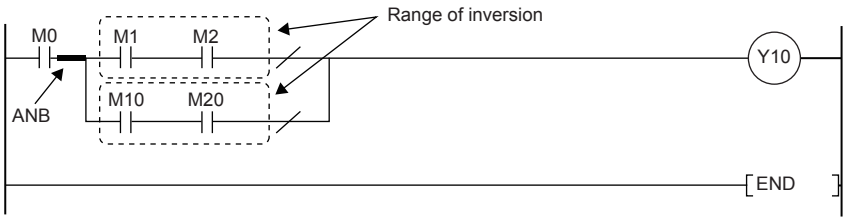
| Operation result up to the INV instruction | Operation result after execution of INV instruction |
|---|---|
| OFF | ON |
| ON | OFF |

### Operation error

There is no operation error.

**Point**

- This instruction operates using the operation result so far. Hence, use it at the same position as the AND instruction. This instruction cannot be used at positions where the LD and OR instructions are programmed.
- If a ladder block is used, the operation result is inverted within the range of the ladder block. When operating a ladder with this instruction and the ANB instruction, pay attention to the inversion range.



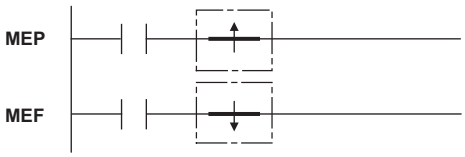

For details ANB instruction, refer to the following.

☞ Page 106 ANB, ORB

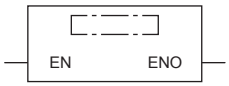
# Converting the operation result into a pulse

## MEP, MEF

- MEP: This instruction turns ON at the rising edge of the operation result up to the MEP instruction and turns OFF in other instances.
- MEF: This instruction turns ON at the falling edge of the operation result up to the MEF instruction and turns OFF in other instances.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MEP(EN);<br>ENO:=MEF(EN); |

**FBD/LD**






### Processing details

**■MEP**

- This instruction turns ON (conductive state) at the rising edge (OFF to ON) of the operation result up to this instruction. This instruction turns OFF (non-conductive state) in instances other than the rising edge of the operation result up to this instruction.
- Use of this instruction makes conversion to pulse easier when multiple contacts are connected in series.

**■MEF**

- This instruction turns ON (conductive state) at the falling edge (ON to OFF) of the operation result up to this instruction. This instruction turns OFF (non-conductive state) in instances other than the falling edge of the operation result up to this instruction.
- Use of this instruction makes conversion to pulse easier when multiple contacts are connected in series.

### Operation error

There is no operation error.

**Point**

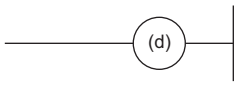
- If an indexed contact is converted to pulse by the subroutine program and the FOR to NEXT instructions, etc., these instructions may not function properly.
- These instructions operate using the operation result so far. Hence, use them at the same position as the AND instruction. These instructions cannot be used at positions where the LD and OR instructions are programmed.

# 6.3 Output Instructions

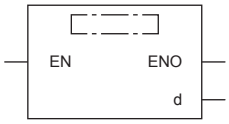
## Out (excluding the timer, counter and annunciator)

### OUT

This instruction outputs the operation result up to this instruction to the specified device.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=OUT(EN,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Number of the device that turns ON/OFF | — | Bit | ANY_BOOL |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DY) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | ○*2 | —*3 | ○*4 | — | — | — | — | — | — | — | — | ○ |

*1 When using F, refer to ☞ Page 115 OUT F.
*2 Only the FX5 intelligent function module can be specified.
*3 When using T, ST, refer to ☞ Page 111 OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST, OUTHS ST.
When using C, refer to ☞ Page 113 OUT C.
When using LC, refer to ☞ Page 114 OUT LC.
*4 T, ST, C cannot be used.

#### Processing details

- This instruction outputs the operation result up to this instruction to the specified device.

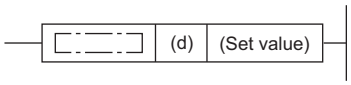
| Condition | Operation result | Coil/specified bit |
|---|---|---|
| When bit device is used | OFF | OFF |
| | ON | ON |
| When bit of word device is specified | OFF | 0 |
| | ON | 1 |

#### Operation error

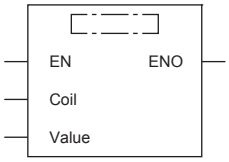
There is no operation error.

## OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST, OUTHS ST

The timer counts up to the set value when the operation result up to the OUT instruction is ON and the coil of the timer/retentive timer specified by (d) turns ON. When the timer times up, NO contact becomes conductive and NC contact becomes non conductive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=OUT_T(EN,Coil,Value);<br>ENO:=OUTH(EN,Coil,Value);<br>ENO:=OUTHS(EN,Coil,Value); |

**FBD/LD**




("OUT_T", "OUTH", "OUTHS" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d)*1 | Timer Number | — | Bit | ANY*3 |
| (Set value)*2 | Timer set value | 0 to 65535 | 16-bit unsigned binary*4 | ANY16*4 |

*1 In the case of the ST language and the FBD/LD language, d displays as Coil.
*2 In the case of the ST language and the FBD/LD language, Set value displays as Value.
*3 Only timer type/retentive timer type label can be used.
*4 In the case of the OUT_T instruction of the ST language and the FBD/LD language , data type is ANY_INT.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | ○*1 | — | — | — | — | — | — | — | — | — | — |
| (Set value) | — | — | — | ○*2 | ○ | — | — | — | — | ○*3 | — | — | — |

*1 Only T and ST can be used.
*2 T, ST, C cannot be used.
*3 Only decimal constant (K) can be used.

### Processing details

- These instructions count up to the set value when the operation result up to the OUT instruction is ON and the coil of the timer/retentive timer specified by (d) turns ON. When the timer reaches the end of its count (current value ≥ set value), NO contact becomes conductive and NC contact becomes non-conductive.
- Operation is as follows when the operation result up to the OUT instruction changes from ON to OFF.

| Timer type | Timer coil | Current timer value | Before time-out | | After time-out | |
|---|---|---|---|---|---|---|
| | | | NO contact | NC contact | NO contact | NC contact |
| Timer | off | 0 | Non-Conductive state | Conductive state | Non-Conductive state | Conductive state |
| Retentive timer | off | Holds current value | Non-Conductive state | Conductive state | Conductive state | Non-Conductive state |

- After the timer times up, clear the current value of the retentive timer and turn the contact off by the RST instruction.
- When the set value is 0, the timer times up when the OUT instruction is executed.

- The following processing is executed when the OUT instruction is executed:
  - The coil in the OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST and OUTHS ST instructions turns ON/OFF
  - The contact in the OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST and OUTHS ST instructions turns ON/OFF
  - The current value in the OUT T, OUTH T, OUTHS T, OUT ST, OUTH ST and OUTHS ST instructions changes
- When the OUT T instruction is skipped using the CJ instruction, etc. while the OUT T and OUT ST instructions are ON, these instructions do not update the current value or turn ON/OFF the contacts.
- When the same OUT T and OUT ST instructions are executed in the same scan twice or more, these instructions update the current value for the same number of times of execution.

**Point**

Values used for timers can be set in the range 1 to 32767. Actual timer constants are as follows since the OUT, OUTH, and OUTHS instructions operate as 100 ms, 10 ms, and 1 ms timers, respectively.

- OUT instruction: 0.1 to 3276.7 seconds
- OUTH instruction: 0.01 to 327.67 seconds
- OUTHS instruction: 0.001 to 32.767 seconds

For the counting method, refer to the following.

MELSEC iQ-F FX5 User's Manual (Application)

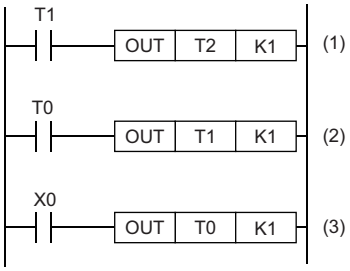
## Precautions

When creating a program for measuring another timer at a timer contact, program in order starting with the timer to be measured later on. In the following instance, all timers turn on in the same scan when the program is created in the measurement order.

- When the set value is shorter than the scan time
- When the set value is 1

**Ex.**

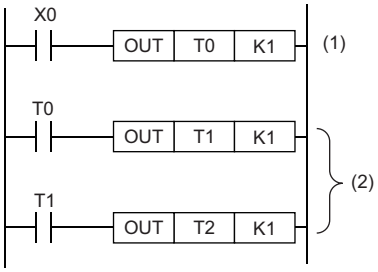
When the T0 to T2 timers are programmed in order from the timer that is measured later




(1) The T2 timer starts measurement from the scan following the scan where the T1 contact turns ON.
(2) The T1 timer starts measurement from the scan following the scan where the T0 contact turns ON.
(3) The T0 timer starts measurement when X0 turns ON.

**Ex.**

When the T0 to T2 timers are programmed in measurement order




(1) The T0 timer starts measurement when X0 turns ON.
(2) The contacts of the T1 and T2 timers also turn on when the contact of T0 turns ON.

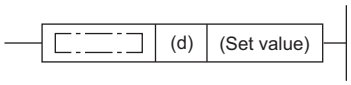
## Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 3405H | A negative value is specified for the timer value. |

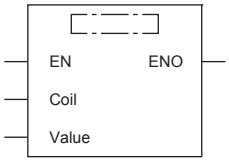
# Counter

## OUT C

This instruction increments the current value of the counter specified by (d) by 1 when the operation result up to OUT instruction changes from OFF to ON, and when the counter reaches the end of its count, NO contact becomes conductive and NC contact becomes non-conductive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=OUT_C(EN,Coil,Value); |

**FBD/LD**




("OUT_C" enters □.)



### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d)*1 | Counter Number | — | Bit | ANY*3 |
| (Set value)*2 | Counter set value | 0 to 65535 | 16-bit unsigned binary | ANY_INT |

*1 In the case of the ST language and the FBD/LD language, d displays as Coil.
*2 In the case of the ST language and the FBD/LD language, Set value displays as Value.
*3 Only timer type/retentive timer type label can be used.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | ○*1 | — | — | — | — | — | — | — | — | — | — |
| (Set value) | — | — | — | ○*2 | ○ | — | — | — | — | ○*3 | — | — | — |

*1 Only C can be used.
*2 T, ST, C cannot be used.
*3 Only decimal constant (K) can be used.

### Processing details

- This instruction increments the current value of the counter specified by (d) by 1 when the operation result up to OUT instruction changes from OFF to ON, and when the counter reaches the end of its count (current value ≥ set value), NO contact becomes conductive and NC contact becomes non-conductive.
- The counter does not count while the operation result remains on. (Count input does not need to be converted to pulses.)
- After a count up, the count value and contact status do not change until the RST instruction is executed.
- When the set value is 0, the same processing as for set value 1 is performed.

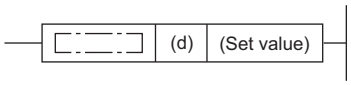
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 3405H | A negative value is specified for the set value. |

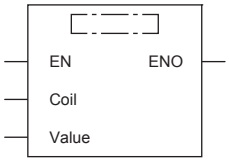
# Long counter

## OUT LC

This instruction increments the current value of the long counter specified by (d) by 1 when the operation result up to the OUT instruction changes from OFF to ON, and when the counter reaches the end of its count, NO contact becomes conductive and NC contact becomes non-conductive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=OUT_C(EN,Coil,Value); |

**FBD/LD**




("OUT_C" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d)*1 | Long counter number | — | Bit | ANY*3 |
| (Set value)*2 | Long counter set value | 0 to 4294967295 | 32-bit unsigned binary | ANY_INT |

*1 In the case of the ST language and the FBD/LD language, d displays as Coil.
*2 In the case of the ST language and the FBD/LD language, Set value displays as Value.
*3 Only timer type/retentive timer type label can be used.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | ○*1 | — | — | — | — | — | — | — | — | — | — |
| (Set value) | — | — | — | ○*2 | ○ | — | — | — | — | ○*3 | — | — | — |

*1 Only LC can be used.
*2 T, ST, C cannot be used.
*3 Only decimal constant (K) can be used.

### Processing details

- This instruction increments the current value of the long counter specified by (d) by 1 when the operation result up to the OUT instruction changes from OFF to ON, and when the counter reaches the end of its count (current value ≥ set value), NO contact becomes conductive and NC contact becomes non-conductive.
- The counter does not count while the operation result remains on. (Count input does not need to be converted to pulses.)
- After a count up, the count value and contact status do not change until the RST or ZRST instruction is executed.
- When the set value is 0, the same processing as for set value 1 is performed.
- When signed (-2147483648 to + 2147483647) or high speed counter is assigned to the LC, the UDCNTF instruction is used. For the UDCNTF instruction, refer to ☞ Page 544 UDCNTF.

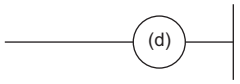
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2821H | When the high speed counter is assigned to the specification long counter. |

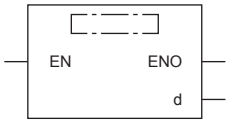
# Annunciator

## OUT F

This instruction outputs the operation result up to the OUT F instruction to the specified annunciator.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=OUT(EN,d); |

**FBD/LD**




("OUT" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Annunciator number that turns ON | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only F can be used.

### Processing details

- This instruction outputs the operation result up to the OUT F instruction to the specified annunciator.
- Operation is as follows when annunciator (F) is turned ON by the OUT F instruction.
  - The annunciator number (F number) that turns ON is stored in special registers (SD64 to SD79).
  - The content of SD63 is incremented by 1.
- When the content of SD63 is 16 (16 annunciators are already on), the annunciator number that turns ON is not stored in SD64 to SD79 even if a new annunciator turns ON.
- Operation is as follows when annunciator (F) is turned OFF by the OUT F instruction:
  - The coil turns OFF, but the contents of SD64 to SD79 do not change.
  - To delete an annunciator that has turned OFF by the OUT F instruction from SD64 to SD79, use the RST F instruction.

#### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SD62 | Annunciator (F) Detection No. | This register stores the earliest detected annunciator (F) No.. |
| SD63 | Annunciator (F) Detection Number | This register stores the number of annunciator (F) detections. |
| SD64 to SD79 | Annunciator (F) Detection No. table | This register stores the annunciator (F) detection No. |

### Operation error

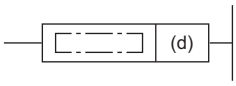
There is no operation error.

# Setting devices (excluding annunciator)

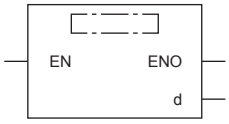
## SET

The status of the device specified by (d) changes as follows when the execution command turns ON.

- Bit device: Turns the coils and contacts ON.
- Bit specification of word device: Set the specified bit to 1.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SET(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Bit device number/ Bit specification of word device to be set (turns ON) | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DY) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | ○*2 | — | ○*3 | — | — | — | — | — | — | — | — | ○ |

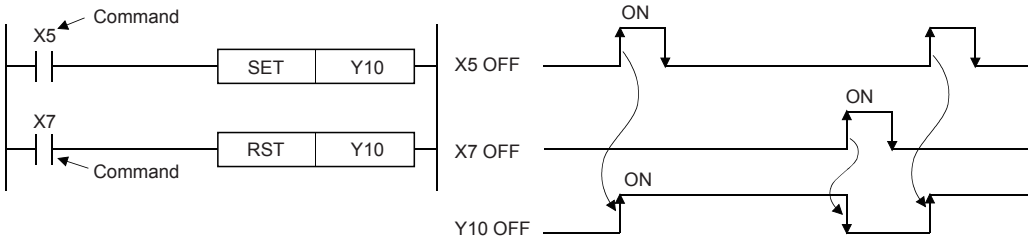
*1 When using F, refer to ☞ Page 120.
*2 Only the FX5 intelligent function module can be used.
*3 T, ST, C cannot be used.

### Processing details

- The status of the specified device changes as follows when the execution command turns ON.

| Device | Device status |
|---|---|
| Bit devices | Turns coils and contacts ON. |
| Bit specification of word device | Sets the specified bit to 1. |

- A device that is turned ON is held on even if the execution command turns OFF. Devices that are turned ON by the SET instruction can be turned OFF by the RST instruction.




- When the execution command is OFF, the device status does not change.

### Precautions

When the SET and RST instructions are executed on the same output relay (Y), the result of the instruction nearer the END instruction (end of program) is output.

## Operation error

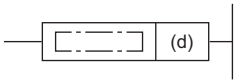
There is no operation error.

# Resetting devices (excluding annunciator)

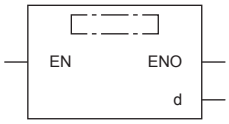
## RST

The status of the device specified by (d) changes as follows when the RST input turns ON.

- Bit devices: Turns the coils and contacts OFF.
- Timers, counters: Sets the current value to 0, and turns coils and contacts OFF.
- Bit specification of word device: Sets the specified bit to 0.
- Word devices, module access devices, index registers: Sets content to 0.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RST(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Bit device number/ bit specification of word device to be reset, or word device number to be reset | — | Bit/word/double word | ANY_ELEMENTARY |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | (DY) |
| (d) | ○*1 | ○*2 | ○ | ○*3 | ○ | ○ | ○ | ○ | ○ | — | — | — | ○ |

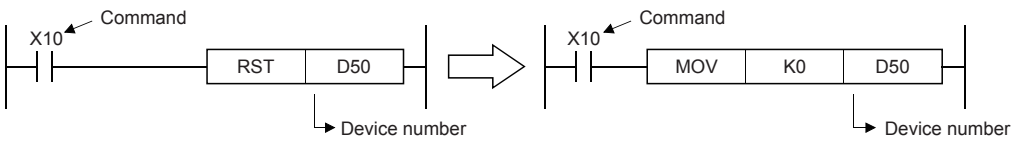
*1 When using F, refer to ☞ Page 122.
*2 Only the FX5 intelligent function module can be used.
*3 T, ST, C cannot be used.

### Processing details

- The status of the specified device changes as follows when the execution command turns ON.

| Device | Device status |
|---|---|
| Bit devices | Turns coils and contacts OFF. |
| Timers, counters | Sets the current value to 0, and turns coils and contacts OFF. |
| Bit specification of word device | Set the specified bit to 0. |
| Word devices, module access device, index registers | Sets content to 0. |

- When the execution command is OFF, the device status does not change.
- Function when a word device is specified by the RST instruction is the same as the following circuit.



## Precautions

When the RST instruction for a timer or counter is executed by a program containing a jump or by a subroutine program or interrupt program, the timer or counter is held in a reset state, and the timer or counter may not work normally.

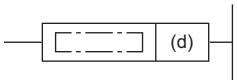
## Operation error

There is no operation error.

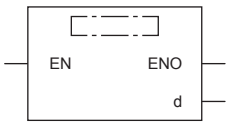
# Setting annunciator

## SET F

This instruction turns ON the specified annunciator.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SET(EN,d); |

**FBD/LD**




("SET" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Annunciator number (F number) that is set | — | Bit | ANY_BOOL |

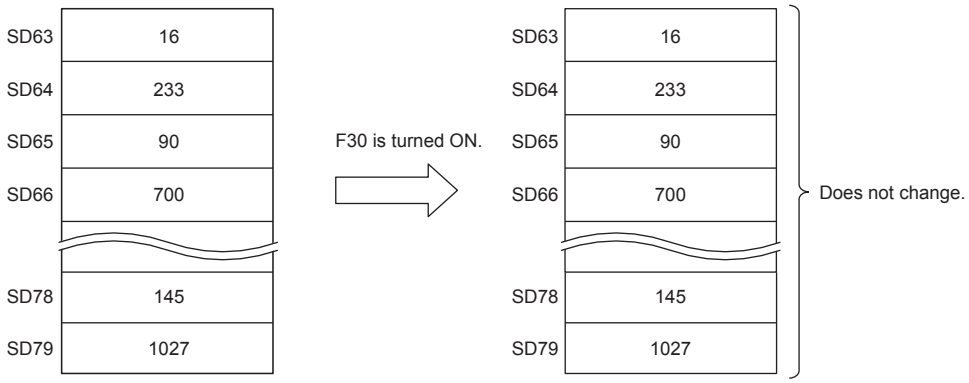
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only F can be used.

### Processing details

- This instruction turns ON the annunciator specified by (d) when the execution command turns ON.
- Operation is as follows when annunciator (F) is turned ON.
  - The annunciator number (F number) that turns ON is stored in special registers (SD64 to SD79).
  - The content of SD63 is incremented by 1.
- When the content of SD63 is 16 (16 annunciators are already ON), the annunciator number that turns ON is not stored in SD64 to SD79 even if a new annunciator turns ON.




#### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SD62 | Annunciator (F) Detection No. | This register stores the earliest detected annunciator (F) No.. |
| SD63 | Annunciator (F) Detection Number | This register stores the number of annunciator (F) detections. |
| SD64 to SD79 | Annunciator (F) Detection No. table | This register stores the annunciator (F) detection No. |

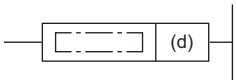
## Operation error

There is no operation error.

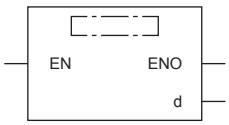
# Resetting annunciator

## RST F

This instruction turns OFF the specified annunciator.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RST(EN,d); |

**FBD/LD**




("RST" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Annunciator number (F number) that is reset | — | Bit | ANY_ELEMENTARY |

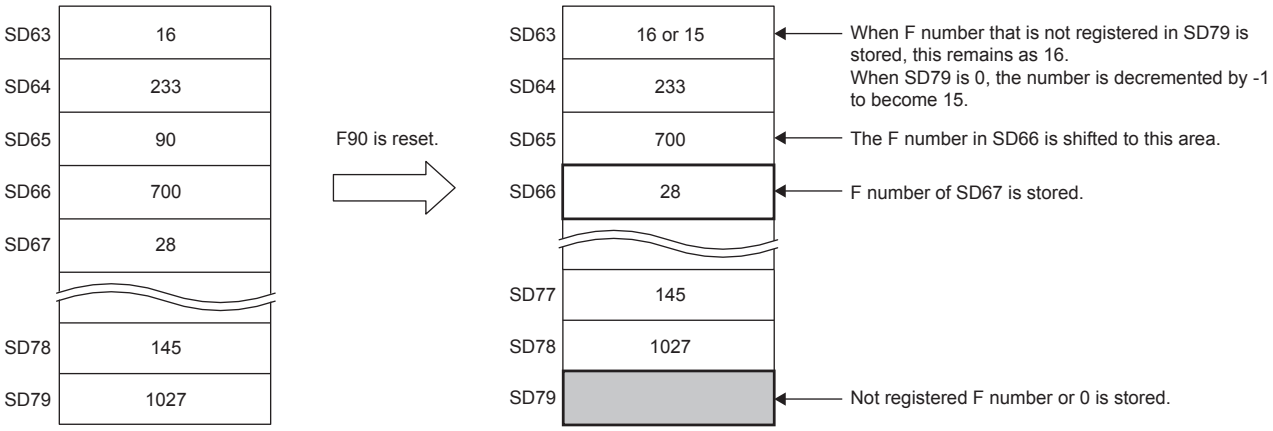
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only F can be used.

### Processing details

- This instruction turns OFF the annunciator specified by (d) when the execution command turns ON.
- An annunciator number (F number) that turns OFF is deleted from special registers (SD64 to SD79) and the content of SD63 is decremented by 1.
- When the content of SD63 is 16, annunciator numbers are deleted from SD64 to SD79 by the RST instruction. Also, if an annunciator not registered in SD64 to SD79 turns ON, its number is registered. When there are two or more unregistered numbers, this instruction adds the numbers starting from the smallest annunciator number. SD63 is not decremented by 1 when the numbers not registered in SD64 to SD79 are turned OFF.



## ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SD62 | Annunciator (F) Detection No. | This register stores the earliest detected annunciator (F) No.. |
| SD63 | Annunciator (F) Detection Number | This register stores the number of annunciator (F) detections. |
| SD64 to SD79 | Annunciator (F) Detection No. table | This register stores the annunciator (F) detection No.. |

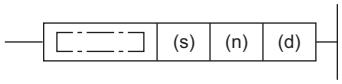
### Operation error

There is no operation error.

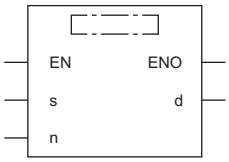
# Setting annunciator (with check time)

## ANS

This instruction sets the annunciator (F device).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ANS(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Timer number for evaluation time | — | 16-bit signed binary | ANY16 |
| (n) | Evaluation time data | 1 to 32767 | 16-bit unsigned binary | ANY16_U |
| (d) | Annunciator device to be set | — | Bit | ANY_BOOL |

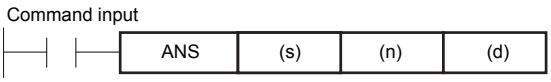
#### ■Applicable devices

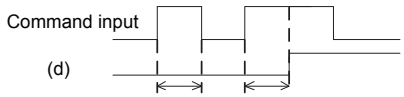
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*2 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only T can be used.
*2 Only F can be used.

### Processing details

- This instruction sets (d) when the command input remains ON continuously for the evaluation time [(n)×100 ms, (s)] or more. This instruction resets the current value of (s) evaluation timer and does not set (d) when the command time is less than the evaluation time [(n)×100 ms]. Also, this instruction resets the evaluation timer when the command input turns OFF.




Less than the evaluation time ((n) × less than 100 ms)

Equivalent to or longer than the evaluation time ((n) × 100 ms or more)

#### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM8049 | ON status annunciator smallest number enabled | When SM8049 is turned ON, SM8048 and SD8049 are enabled. |
| SM8048 | Annunciator operation | When one of the F devices is operating, SM8048 turns ON. |
| SD8049 | ON status annunciator smallest number | The smallest number of the F devices that are operating is stored. |

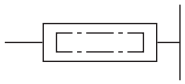
### Operation error

There is no operation error.

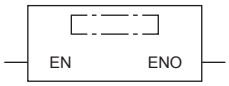
# Resetting annunciator (smallest number reset)

## ANR(P)

This instruction resets the lowest number annunciator (F device) in the ON status.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ANR(EN);<br>ENO:=ANRP(EN); |

**FBD/LD**




### Processing details

• Annunciator (F device) that is operating (in ON status) is reset when the command input turns ON.

This instruction resets the annunciator with the smallest number when multiple annunciators are ON. If the command input is turned ON again, this instruction resets the annunciator with the next smallest number among annunciators (F devices) that are operating.


### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM8049 | On status annunciator smallest number enabled | When SM8049 is turned ON, SM8048 and SD8049 are enabled. |
| SM8048 | Annunciator operation | When one of the F devices is operating, SM8048 turns ON. |
| SD8049 | On status annunciator smallest number | The smallest number of the F devices that are operating is stored. |

### Precautions

• When ANR instruction is used, annunciators in the ON status are reset in turn in each operation cycle.
• This is executed for only 1 operation cycle (only once) when the ANRP instruction is used.

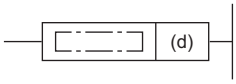
### Operation error

There is no error.

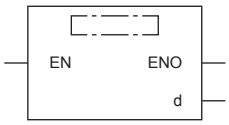
# Rising edge output

## PLS

This instruction turns ON the device specified by (d) for one scan when the PLS command turns from OFF to ON, and turns OFF in other cases.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLS(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device to be converted to pulse | — | Bit | ANY_BOOL |

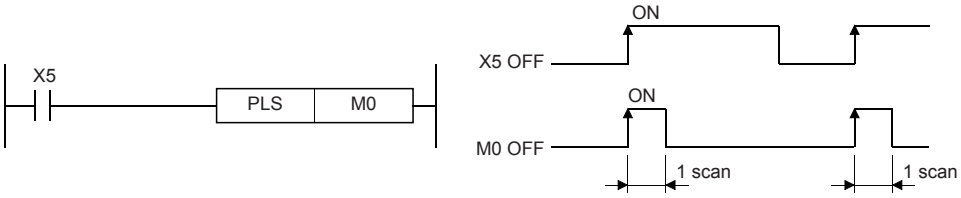
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DY) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | ○*1 | — | ○*2 | — | — | — | — | — | — | — | — | ○ |

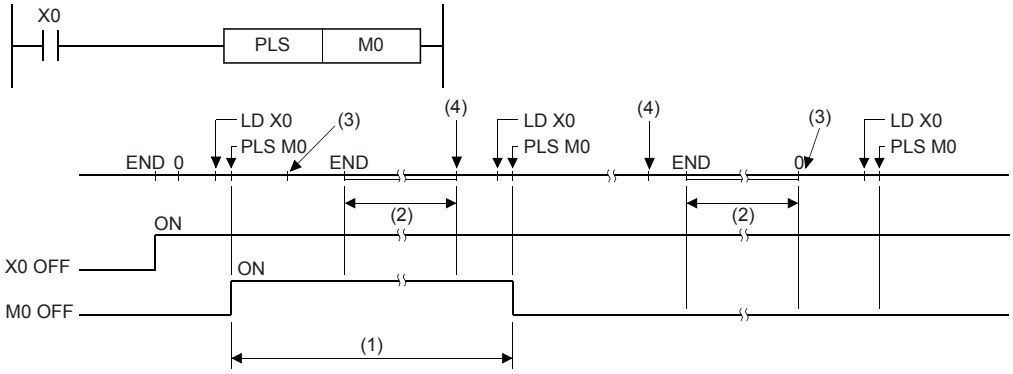
*1 Only the FX5 intelligent function module can be used.
*2 T, ST, C cannot be used.

### Processing details

- This instruction turns ON the specified device for one scan when the PLS command turns from OFF to ON, and turns OFF in other cases. When there is one PLS instruction programmed for the device specified by (d) during a scan, the specified device turns ON for one scan.



- If the RUN/STOP/RESET switch is changed from RUN to STOP after execution of the PLS instruction, the PLS instruction will not be executed even if the switch is set to RUN again.




(1) 1 scan of PLS M0
(2) CPU module operation stop time
(3) Set the RUN/STOP/RESET switch on the CPU module to RUN→STOP.
(4) Set the RUN/STOP/RESET switch on the CPU module to STOP→RUN.

## Precautions

- When write during RUN is completed for a circuit including a rising edge instruction (LDP/ANDP/ORP instruction), the instruction is not executed regardless of the ON/OFF status of the target device of the rising edge instruction. Also, in the case of a rising edge instruction (PLS instruction), the instruction is not executed regardless of the ON/OFF status of the device that is set as the operation condition. The instruction is executed when the target device and the device in the operation conditions is set from OFF to ON again.
- Note that the device specified by (d) sometimes turns ON for one scan or more when the PLS instruction is made to jump by the CJ instruction or the executed subroutine program was not called by the CALL(P) instruction.

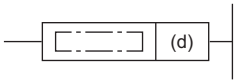
## Operation error

There is no operation error.

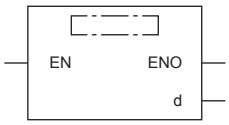
# Falling edge output

## PLF

This instruction turns ON the device specified by (d) for one scan when the PLF command turns from ON to OFF, and turns OFF in other cases.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLF(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device to be converted to pulse | — | Bit | ANY_BOOL |

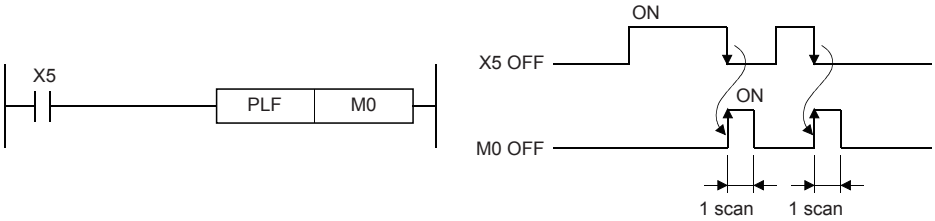
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | (DY) |
| (d) | ○ | ○*1 | — | ○*2 | — | — | — | — | — | — | — | — | ○ |

*1 Only the FX5 intelligent function module can be used.
*2 T, ST, C cannot be used.

### Processing details

- This instruction turns ON the specified device for one scan when the PLF command turns OFF from ON, and turns OFF in other cases. When there is one PLF instruction programmed for the device specified by (d) during a scan, the specified device turns ON for one scan.




- If the RUN/STOP/RESET switch is changed from RUN to STOP after execution of the PLF instruction, the PLF instruction will not be executed even if the switch is set to RUN again.

### Precautions

- When write during RUN is completed for a circuit including a falling edge instruction (LDF/ANDF/ORF instruction), the instruction is not executed regardless of the ON/OFF status of the target device of the falling edge instruction. Also, in the case of a falling edge instruction (PLF instruction), the instruction is not executed regardless of the ON/OFF status of the device that is set as the operation condition. The instruction is executed when the target device and the device in the operation conditions is set from ON to OFF again.
- Note that the device specified by (d) sometimes turns ON for one scan or more when the PLF instruction is made to jump by the CJ instruction or the executed subroutine program was not called by the CALL(P) instruction.

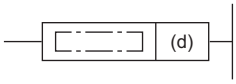
## Operation error

There is no operation error.

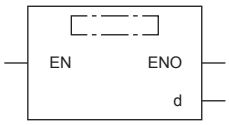
# Inverting the bit device output

## FF

This instruction reverses the output status of the device specified by (d) when the execution command changes from OFF to ON.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FF(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device number to be reversed | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | (DY) |
| (d) | ○ | ○*1 | — | ○*2 | — | — | — | — | — | — | — | — | ○ |

*1 Only the FX5 intelligent function module can be used.
*2 T, ST, C cannot be used.

### Processing details

- This instruction reverses the state of the device specified by (d) when the execution command changes from OFF to ON.

| Device | Device status | |
|---|---|---|
| | Before execution of FF instruction | After execution of FF instruction |
| Bit devices | OFF | ON |
| | ON | OFF |
| Bit specification of word device | 0 | 1 |
| | 1 | 0 |

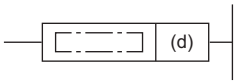
### Operation error

There is no operation error.

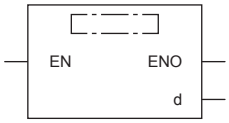
# Inverting the bit device output

## ALT(P)

These instructions reverse (ON ↔ OFF) bit devices when input turns ON.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ALT(EN,d);<br>ENO:=ALTP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Bit device number whose output is alternated | — | Bit | ANY_BOOL |

#### ■Applicable devices

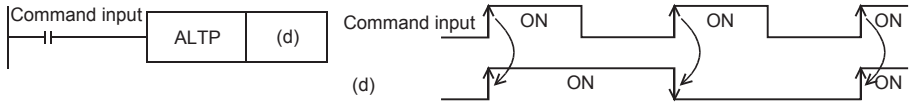
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | ○*1 | — | ○*2 | — | — | — | — | — | — | — | — | — |

*1 Only the FX5 intelligent function module can be used.
*2 T, ST, C cannot be used.

### Processing details

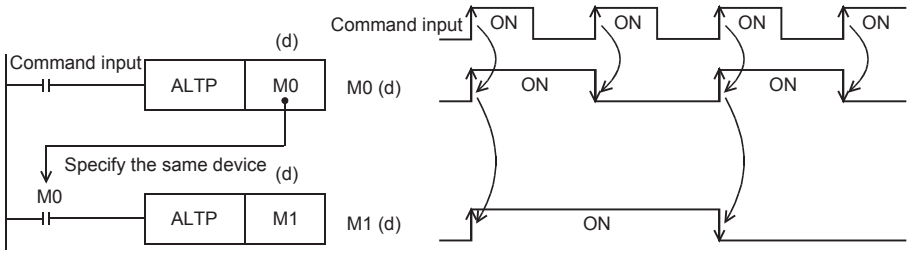
#### ■Alternating output (1-step)

The bit device specified by (d) is reversed ON ↔ OFF each time the command input changes from OFF to ON.




#### ■Division output (according to alternating output (2-step))

The ALTP instruction can be used in multiple combinations to perform division output.




### Precautions

When the CPU module is programmed with the ALT instruction, reversal operation is performed at every operation cycle. To perform reversal operation by command ON/OFF, either use the ALTP instruction (pulse execution type) or set a command contact as LDP etc. (pulse execution type).

### Operation error

There is no operation error.

# 6.4 Shift Instructions

## Shifting bit devices

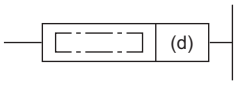
### SFT(P)

• In case of bit device:

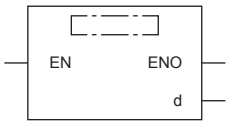
These instructions shift the ON/OFF status of the device before the device specified by (d) to the device specified by (d).

• When bit of word device is specified:

These instructions shift the 1/0 status of the bit before the bit specified by (d) to the bit specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFT(EN,d);<br>ENO:=SFTP(EN,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device number to receive shift | — | Bit | ANY_BOOL |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | (DY) |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | ○ |

*1 T, ST, C cannot be used.

#### Processing details

##### ■In case of bit device

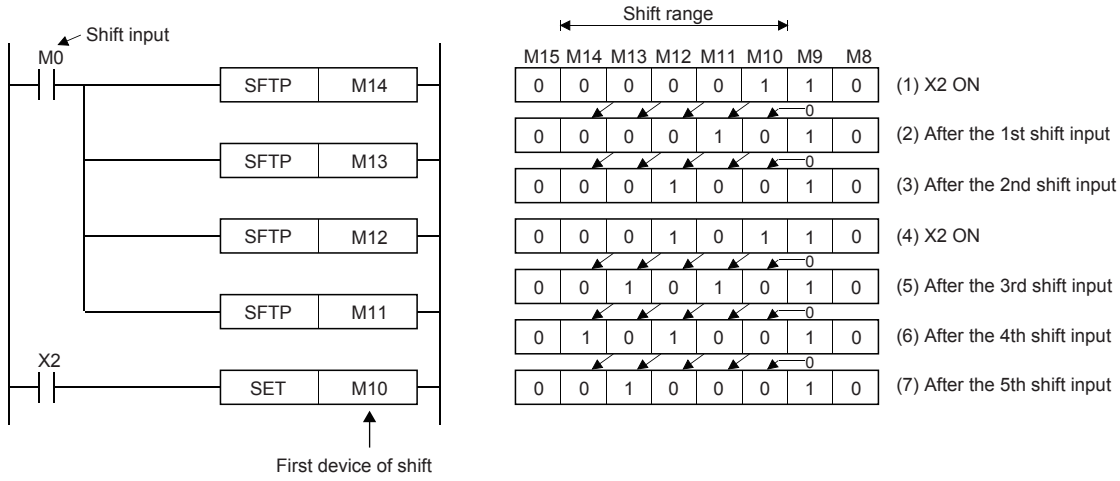
• This instruction shifts the ON/OFF status of the device before the device specified by (d) to the device specified by (d). The device before the device specified by (d) turns OFF.

Ex.

When M11 is specified by the SFTP instruction and the SFTP instruction is executed, the ON/OFF status of M10 is shifted to M11 and M10 is turned OFF.

• Turn ON the first device to be shifted by the SET instruction.

• When the SFT(P) instruction is used consecutively, create the program to start from the device with the largest number.

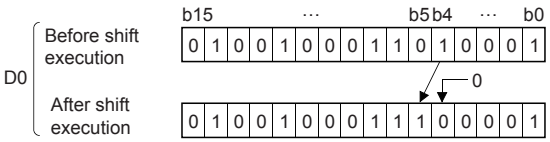



## ■When bit of word device is specified:

• This instruction shifts the 1/0 status of the bit before the bit specified by (d) to the bit specified by (d). The bit before the bit specified by (d) becomes 0.

**Ex.**

When D0.5 (bit 5 (b5) of D0) is specified by the SFT(P) instruction and the SFT(P) instruction is executed, the 1/0 status of b4 of the D0 is shifted to b5 and b4 is set to 0.



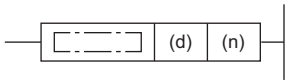

## Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The device specified by (d) exceeds the corresponding device range. |

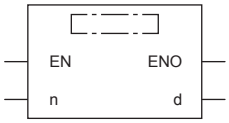
# Shifting 16-bit data to the right by n bit(s)

## SFR(P)

These instructions shift the 16-bit data in the device specified by (d) to the right by (n) bit(s).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFR(EN,n,d);<br>ENO:=SFRP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

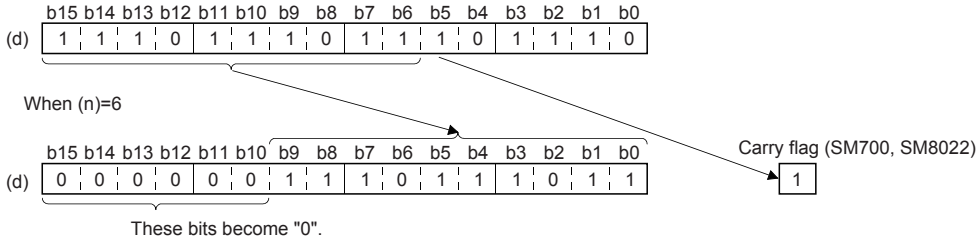
| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the shift-target data is stored | — | 16-bit signed binary | ANY16 |
| (n) | Number of shifts | 0 to 15 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

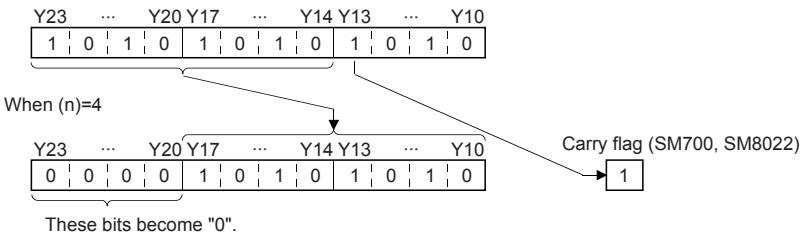
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction shifts the 16-bit data in the device specified by (d) to the right by (n) bit(s) from the most significant bit. The (n) bit(s) from the most significant bit is/are filled with 0(s).




- When (d) is a bit device, bits are shifted to the right within the device range specified by nibble specification.




- Specify any value between 0 and 15 for (n). If a value 16 or larger is specified for (n), bits are shifted to the right by the remainder value of (n)÷16. For example, when (n) is 18, data is shifted by 2 bits to the right because 18 divided by 16 equals 1 with a remainder of 2.

## ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM700 | Carry | ON/OFF according to the status (1/0) of the (n-1)th bit. |
| SM8022 | | |

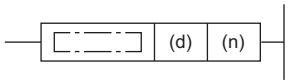
## Operation error

There is no operation error.

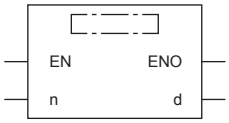
# Shifting 16-bit data to the left by n bit(s)

## SFL(P)

These instructions shift the 16-bit data in the device specified by (d) to the left by (n) bit(s).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFL(EN,n,d);<br>ENO:=SFLP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

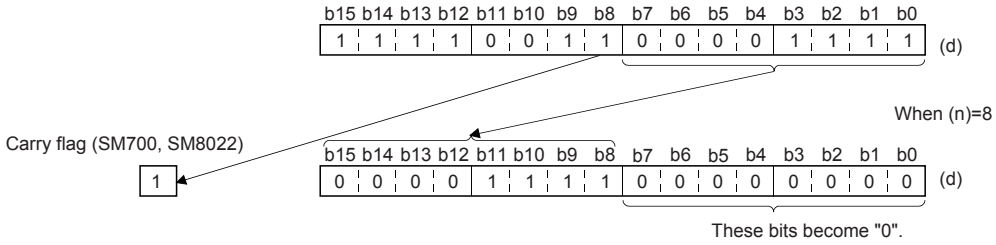
| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the shift-target data is stored | — | 16-bit signed binary | ANY16 |
| (n) | Number of shifts | 0 to 15 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

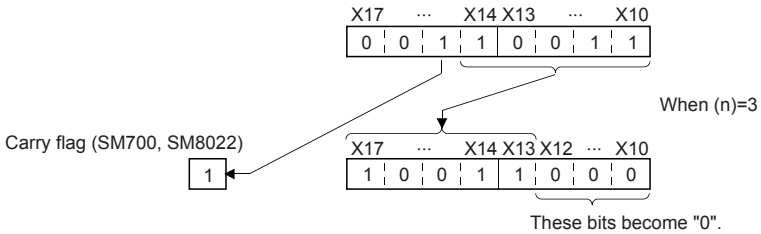
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions shift the 16-bit data in the device specified by (d) to the left by (n) bit(s) from the least significant bit. (n) bits from the least significant bit are filled with "0".




- When (d) is a bit device, bit(s) are shifted to the left within the device range specified by nibble specification.




- Specify any value between 0 and 15 for (n). If a value 16 or larger is specified for (n), bit(s) are shifted to the left by the remainder value of (n)÷16. For example, when (n) is 18, data is shifted by 2 bits to the left because 18 divided by 16 equals 1 with a remainder of 2.

## ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM700 | Carry | ON/OFF according to the status (1/0) of the (n-1)th bit. |
| SM8022 | | |

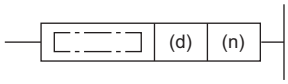
## Operation error

There is no operation error.

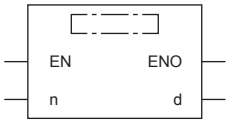
# Shifting n-bit data to the right by 1 bit

## BSFR(P)

These instructions shift (n) point(s) of data to the right by 1 bit from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BSFR(EN,n,d);<br>ENO:=BSFRP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number to be shifted | — | Bit | ANY_BOOL |
| (n) | Number of devices to be shifted | 0 to 65535 | 16-bit unsigned binary | ANY16 |

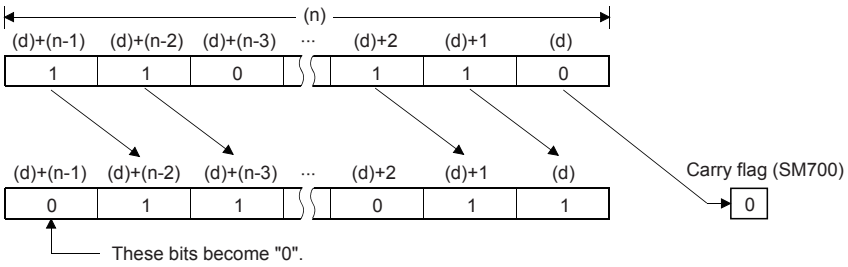
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions shift (n) point(s) of data to the right by 1 bit from the device specified by (d).




- The value of the device specified by (d) + (n-1) becomes 0.

#### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM700 | Carry | ON/OFF according to the status (1/0) of the (d) bit. |

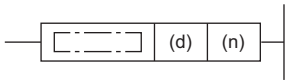
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n) points of data starting from the device specified by (d) exceed in the corresponding device. |

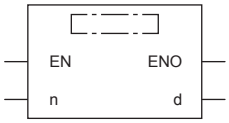
# Shifting n-bit data to the left by 1 bit

## BSFL(P)

These instructions shift (n) point(s) of data to the left by 1 bit from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BSFL(EN,n,d);<br>ENO:=BSFLP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number to be shifted | — | Bit | ANY_BOOL |
| (n) | Number of devices to be shifted | 0 to 65535 | 16-bit unsigned binary | ANY16 |

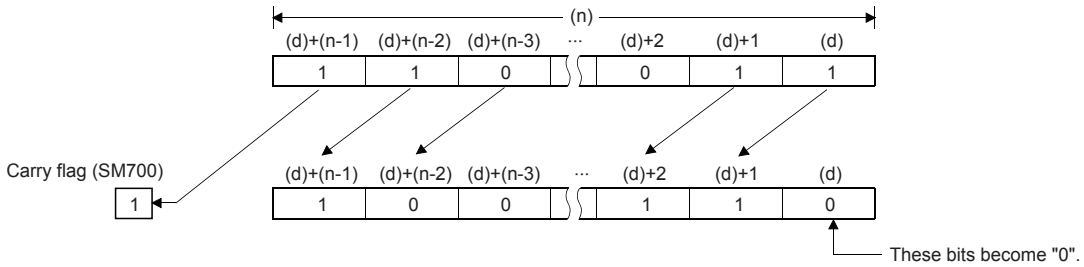
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions shift (n) point(s) of data to the left by 1 bit from the device specified by (d).




- The value of the device specified by (d) becomes 0.

#### ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM700 | Carry | ON/OFF according to the status (1/0) of the (d) bit. |

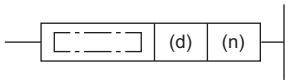
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n) points of data starting from the device specified by (d) exceed in the corresponding device. |

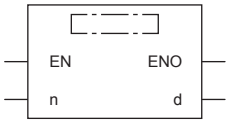
# Shifting n-word data to the right by 1 word

## DSFR(P)

These instructions shift (n) point(s) of data to the right by 1 word from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSFR(EN,n,d);<br>ENO:=DSFRP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

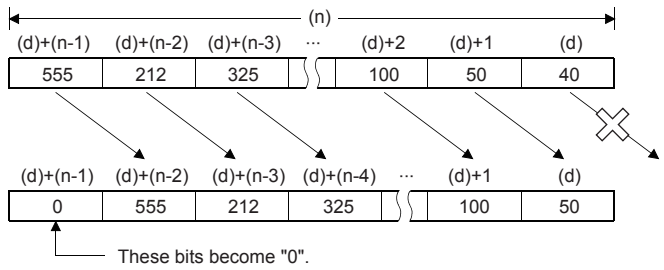
| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number to be shifted | — | Word | ANY16 |
| (n) | Number of devices to be shifted | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions shift (n) point(s) of data to the right by 1 word from the device specified by (d).




- The value of the device specified by (d) + (n-1) becomes 0.

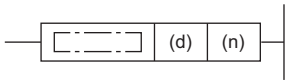
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n) points of data starting from the device specified by (d) exceed in the corresponding device. |

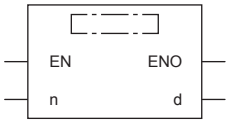
# Shifting n-word data to the left by 1 word

## DSFL(P)

These instructions shift (n) point(s) of data to the left by 1 word from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSFL(EN,n,d);<br>ENO:=DSFLP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

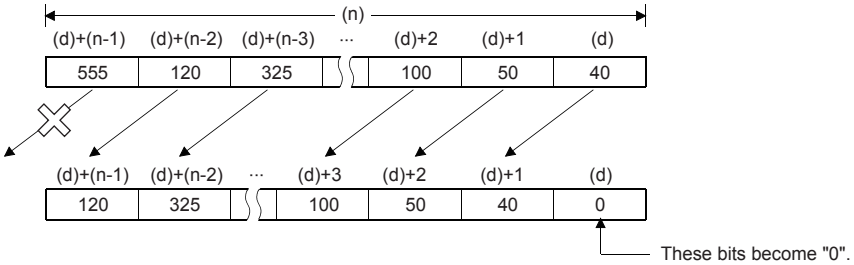
| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number to be shifted | — | Word | ANY16 |
| (n) | Number of devices to be shifted | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions shift (n) point(s) of data to the left by 1 word from the device specified by (d).




- The value of the device specified by (d) becomes 0.

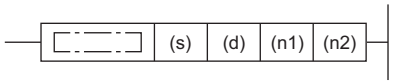
### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n) points of data starting from the device specified by (d) exceed in the corresponding device. |

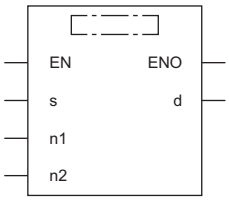
# Shifting n-bit(s) data to the right by (n) bit(s)

## SFTR(P)

These instructions shift (n1) bits of data to the right by (n2) bit(s) from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFTR(EN,s,n1,n2,d);<br>ENO:=SFTRP(EN,s,n1,n2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number stored to the shift data after the shift | — | Bit | ANY_BOOL |
| (d) | Head device number to be shifted | — | Bit | ANY_BOOL |
| (n1)*1 | Data length of shift data | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (n2)*1 | Number of shifts | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

*1 Set so that n2≤n1.

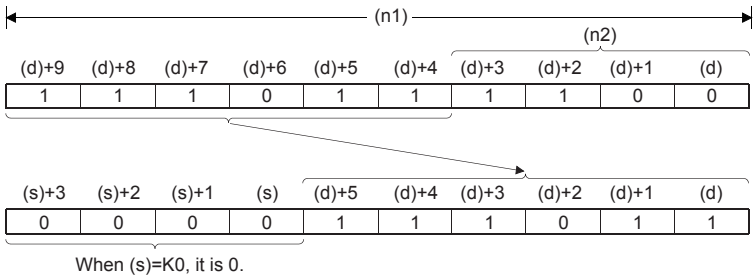
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | — | — | — | — | — | ○*2 | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.
*2 Only 0 or 1 can be used.

### Processing details

- These instructions shift (n1) bits of data to the right by (n2) bit(s) from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d)+(n1-n2).
- When K0 is specified for (s), set 0s for (n2) points of bits from (d)+(n1-n2) after the shift.
- When K1 is specified for (s), set 1s for (n2) points of bits from (d)+(n1-n2) after the shift.



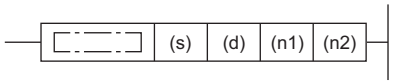
## Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n2) points of data starting from the device specified by (s) exceed in the corresponding device. |
| | The (n1) points of data starting from the device specified by (d) exceed in the corresponding device. |
| 2821H | The transfer source data (s) overlaps with shift device (d). |
| 3405H | A constant other than K0 or K1 is specified when the constant (s) is specified. |
| | The values specified in (n1) and (n2) are such that (n1)<(n2). |

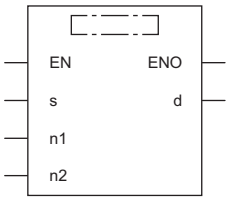
# Shifting n-bit data to the left by n bit(s)

## SFTL(P)

These instructions shift (n1) bits of data to the left by (n2) bit(s) from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFTL(EN,s,n1,n2,d);<br>ENO:=SFTLP(EN,s,n1,n2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number stored to the shift data after the shift | — | Bit | ANY_BOOL |
| (d) | Head device number to be shifted | — | Bit | ANY_BOOL |
| (n1)*1 | Data length of shift data | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (n2)*1 | Number of shifts | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

*1 Set so that n2≤n1.

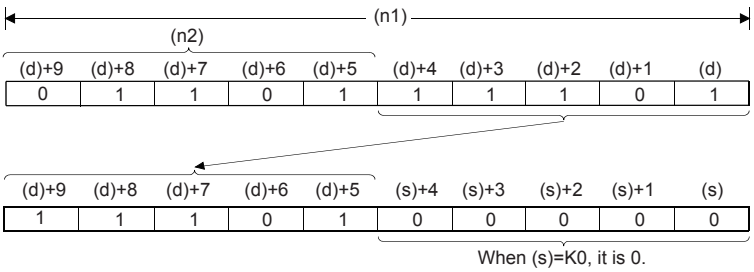
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | — | — | — | — | — | ○*2 | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.
*2 Only 0 or 1 can be used.

### Processing details

- These instructions shift (n1) bits of data to the left by (n2) bit(s) from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d).
- When K0 is specified for (s), set 0s for (n2) points of bits from (d) after the shift.
- When K1 is specified for (s), set 1s for (n2) points of bits from (d) after the shift.



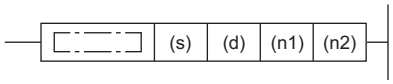
## Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n2) points of data starting from the device specified by (s) exceed in the corresponding device. |
| | The (n1) points of data starting from the device specified by (d) exceed in the corresponding device. |
| 2821H | The transfer source data (s) overlaps with shift device (d). |
| 3405H | A constant other than K0 or K1 is specified when the constant (s) is specified. |
| | The values specified in (n1) and (n2) are such that (n1)<(n2). |

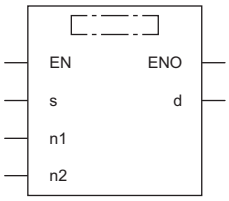
# Shifting n-word data to the right by n word(s)

## WSFR(P)

This instruction shifts (n1) words of data to the right by (n2) word(s) from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WSFR(EN,s,n1,n2,d);<br>ENO:=WSFRP(EN,s,n1,n2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number stored to the shift data after the shift | — | Word | ANY16 |
| (d) | Head device number to be shifted | — | Word | ANY16 |
| (n1)*1 | Data length of shift data | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (n2)*1 | Number of shifts | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

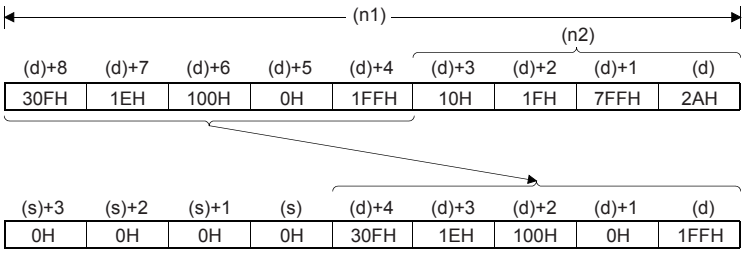
*1 Set so that n2≤n1.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction shifts (n1) words of data to the right by (n2) word(s) from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d)+(n1-n2).
- This instruction sets the specified value for (n2) points of devices from (d) + (n1-n2) after the shift when K is specified for (s).




- When the value specified for (n1) or (n2) is 0, the processing is not performed.

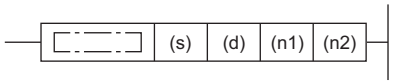
## Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n2) points of data starting from the device specified by (s) exceed in the corresponding device. |
| | The (n1) points of data starting from the device specified by (d) exceed in the corresponding device. |
| 2821H | The transfer source data (s) overlaps with shift device (d). |
| 3405H | A constant other than K0 or K1 is specified when the constant (s) is specified. |
| | The values specified in (n1) and (n2) are such that (n1)<(n2). |

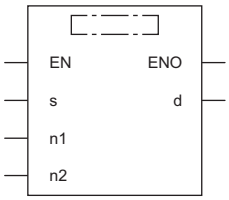
# Shifting n-word data to the left by n word(s)

## WSFL(P)

This instruction shifts (n1) words of data to the left by (n2) word(s) from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WSFL(EN,s,n1,n2,d);<br>ENO:=WSFLP(EN,s,n1,n2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number stored to the shift data after the shift | — | Word | ANY16 |
| (d) | Head device number to be shifted | — | Word | ANY16 |
| (n1)*1 | Data length of shift data | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (n2)*1 | Number of shifts | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

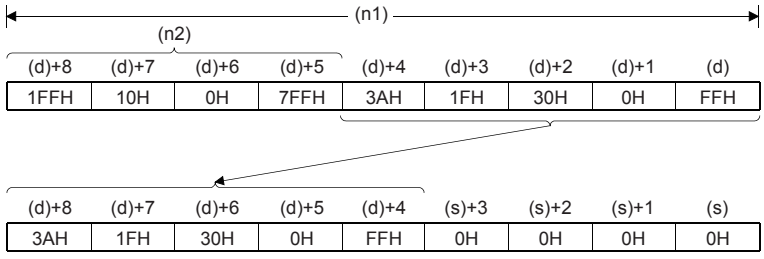
*1 Set so that n2≤n1.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction shifts (n1) words of data to the left by (n2) word(s) from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d).
- This instruction sets the specified value for (n2) points of devices from (d) + (n1-n2) after the shift when K is specified for (s).




- When the value specified for (n1) or (n2) is 0, the processing is not performed.

## Operation error

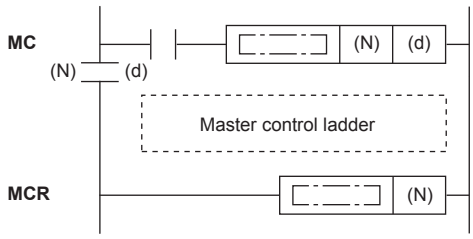
| Error code (SD0/SD8067) | Remarks |
|---|---|
| 2820H | The (n2) points of data starting from the device specified by (s) exceed in the corresponding device. |
| | The (n1) points of data starting from the device specified by (d) exceed in the corresponding device. |
| 2821H | The transfer source data (s) overlaps with shift device (d). |
| 3405H | A constant other than K0 or K1 is specified when the constant (s) is specified. |
| | The values specified in (n1) and (n2) are such that (n1)<(n2). |

# 6.5 Master Control Instruction

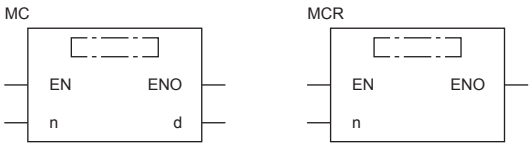
## Setting/resetting the master control

### MC, MCR

- MC: This instruction starts master control.
- MCR: This instruction ends master control.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MC(EN,n,d);<br>ENO:=MCR(EN,n); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Remarks | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (N)*1 | Nesting | 0 to 14 | Device name | ANY16_S |
| (d) | Number of device to be turned ON | — | Bit | ANY_BOOL |

*1 In the case of the ST language and the FBD/LD language, N displays as n.

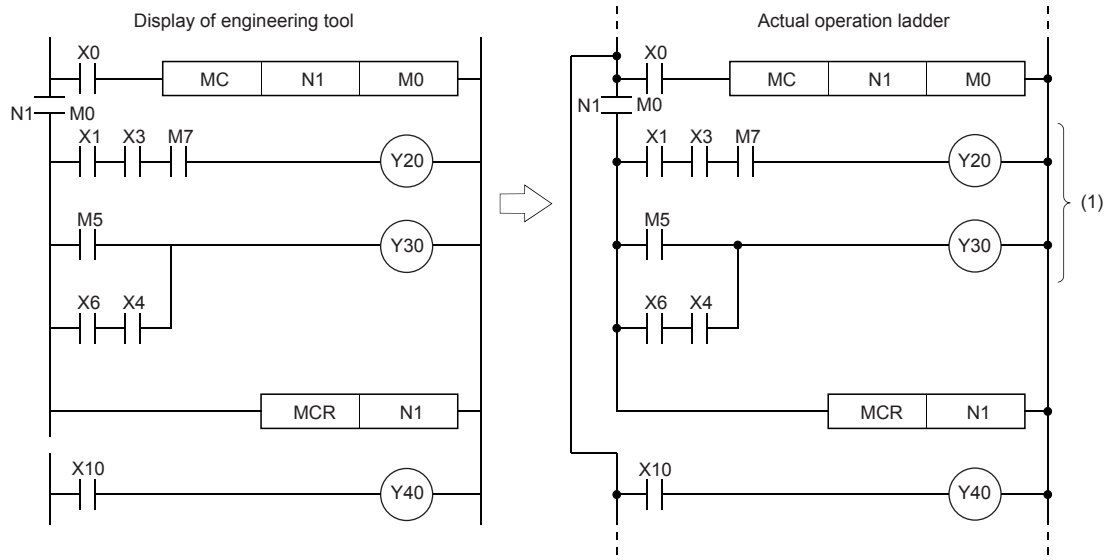
##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | N |
| (N) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

These instructions create program with efficient ladder switching by opening/closing common buses in ladders.
Ladder using master control is illustrated below.




(1) Executed only when X0 is on

### ■MC

- When the execution command of the MC instruction turns ON at the start of master control, the operation result between the MC and MCR instructions is as per the instructions (according to ladder). When the execution command of MC instruction turns OFF, the operation result between the MC and MCR instructions becomes as follows.

| Device | Device status |
|---|---|
| Timer | The count value becomes 0, and both coils and contacts turn OFF. |
| Counters, retentive timers | Coils turn OFF but the current status of both count values and contacts is maintained. |
| Devices in OUT instruction | Forcibly turned OFF. |
| Devices in SET and RST instructions<br>Devices in SFT(P) instruction<br>Devices in basic instructions and applied instructions | Current status is maintained. |

**Point**

When an instruction (e.g. FOR to NEXT instructions etc.) not requiring NO contact instruction is programmed in a ladder using master control, the CPU module executes that instruction regardless of the execution command of this instruction.

- With this instruction, the same nesting (N) number can be used as many times as necessary by changing the device specified by (d).
- When this instruction is ON, the coil of the device specified by (d) turns ON. Also, the coil becomes a double coil when the same device is used by the OUT instruction, for example. So, do not use the device specified by (d) in other instructions.

### ■MCR

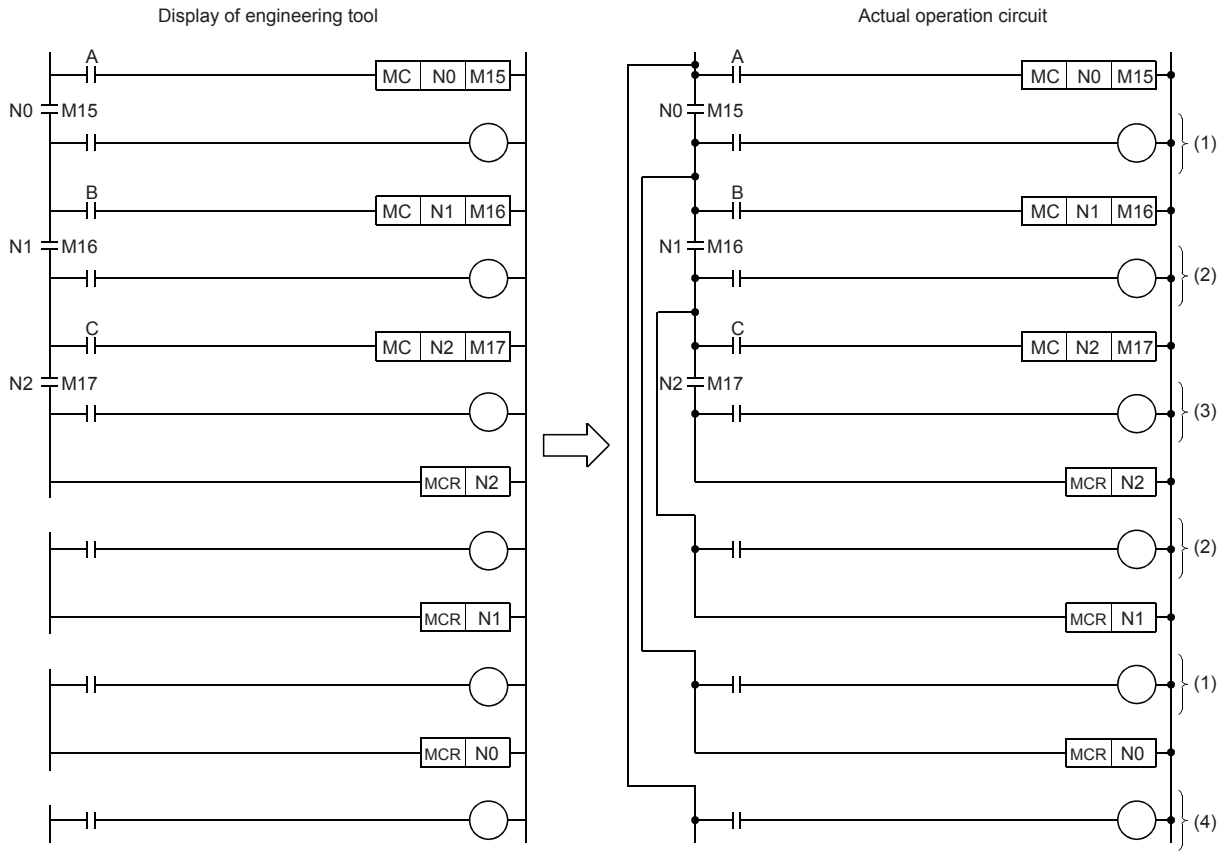
- This instruction indicates the end of the master control range by the master control release instruction.
- Do not prefix this instruction with NO contact instruction.
- Use these (MC and MCR) instructions with same nesting number as a pair. Note, however, that when this instruction is nested at a single location, all master controls can be ended by just one (N) number, the smallest number. (Refer to Caution.)

**Point**

Master control instructions can be used in a nested fashion. Each master control section is distinguished by nesting (N). Nesting is available within the range N0 to N14.

A nested structure allows you to create a ladder for successively restricting program execution conditions.

A nested structure ladder is illustrated as follows:




(1) Executed when A is ON

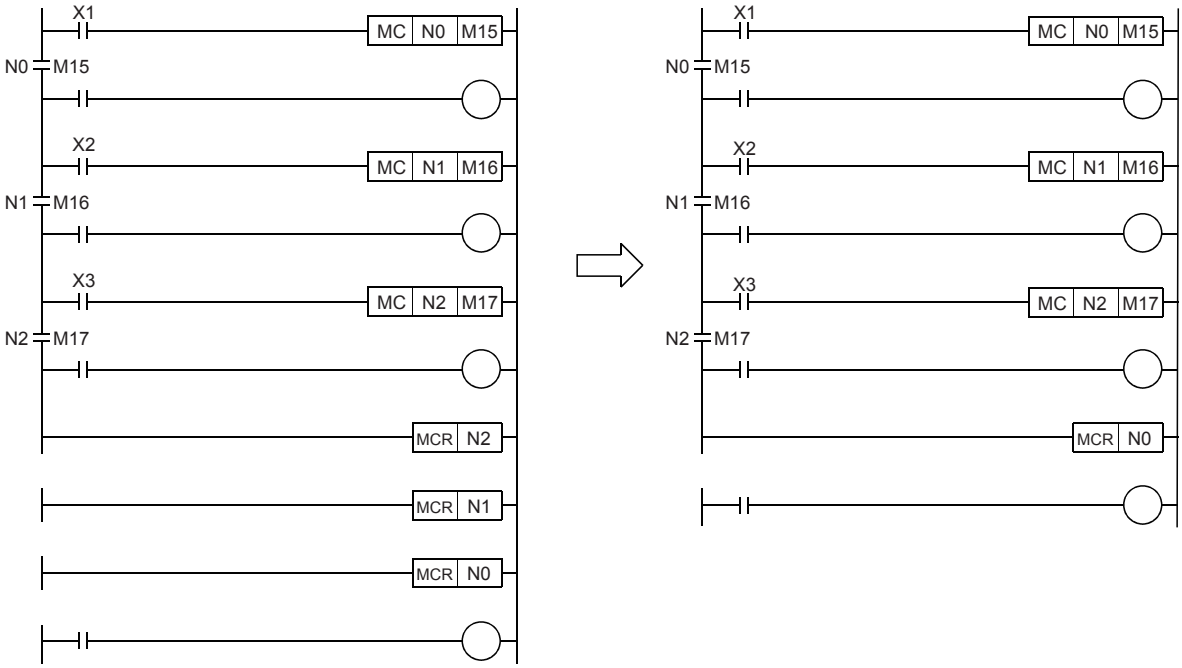
(2) Executed when A and B are ON

(3) Executed when A, B, and C are ON

(4) Regardless of A, B, and C

## Precautions

- If an instruction (e.g. LD, LDI) to be connected to the bus is not programmed following the MC instruction, a ladder error (error code: 33E0) occurs.
- These instructions cannot be used in FOR to NEXT, STL to RETSTL, P to RET (SRET), and I to IRET. Also, do not block by I, IRET, FEND, END, RET (SRET), RETSTL, etc. Addition by write during RUN mode results in an error.
- Nesting up to 15 levels (N0 to N14) is possible. When nesting instructions, the MC instruction is used starting from the smallest (N) number and the MCR instruction is started starting from the biggest number. Programming in reverse order does not produce a nested structure and hence the CPU module cannot execute operations properly.
- When the MCR instruction is nested at a single location, all master controls can be ended by just one nesting (N) number, the smallest number.




## Operation error

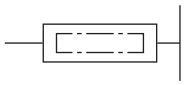
There is no operation error.

# 6.6 Termination Instructions

## Ending the main routine program

### FEND

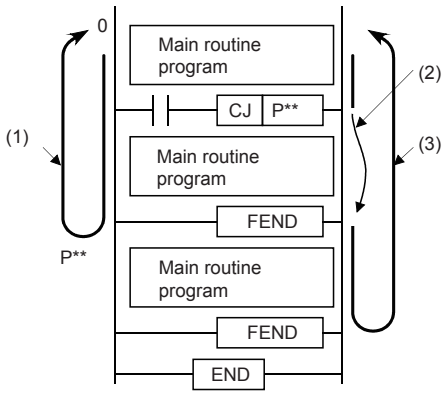
This instruction is used to branch operation of the sequence program by the CJ instruction or to divide the main routine program into a subroutine program or an interrupt program.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported. |

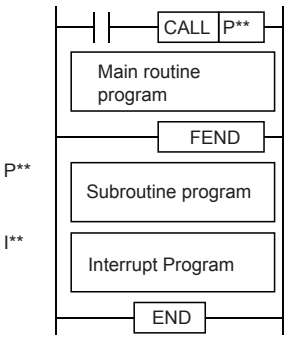
| FBD/LD |
|---|
| Not supported. |

#### Processing details

- This instruction branches operation of the sequence program by the CJ instruction or dividing the main routine program into subroutine programs and interrupt programs.
- When this instruction is executed, program execution returns to the program at step 0 after output processing, input processing and refreshing of the watchdog timer.
- The sequence program from this instruction onwards can also be displayed as ladder by the engineering tool.




(a) When the CJ instruction is used




(b) When there are subroutine programs and interrupt programs

(1) Operation when the CJ instruction is not executed
(2) Jump by the CJ instruction
(3) Operation when the CJ instruction has been executed

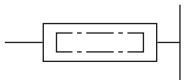
#### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 3340H | The FEND instruction is executed before the NEXT instruction after the FOR instruction is executed. |
| 3381H | The FEND instruction is executed before the RET instruction after the CALL(P) instruction is executed. |
| 33E3H | The FEND instruction is programmed between FOR-NEXT. |
| 33E4H | The FEND instruction is programmed between MC-MCR. |
| 33E5H | The FEND instruction is programmed between STL-RETSTL. |
| 33E7H | The FEND instruction is programmed between I-IRET. |
| 3100H | The FEND instruction is programmed in standby type program. |
| | The FEND instruction is programmed in FB file. |

# Ending the sequence program

## END

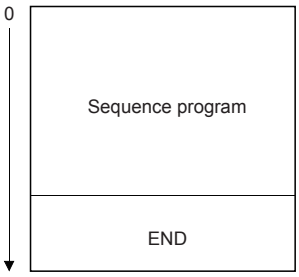
This instruction indicates the end of a program.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported. |

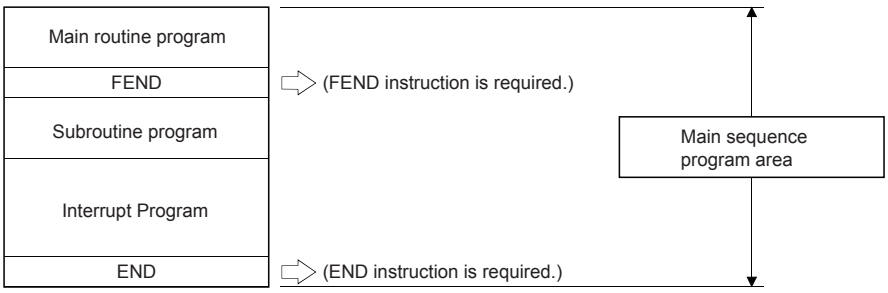
| FBD/LD |
|---|
| Not supported. |

### Processing details

- This instruction indicates the end of all programs including the main routine program, subroutine program, and interrupt program. When this instruction is executed, the CPU module ends execution of the currently executing program.




- The first time the RUN is started, execution begins from this instruction.
- This instruction cannot be programmed midway during the main sequence program. When this processing is required midway during the program, use the FEND instruction.
- When programming is performed using the engineering tool in ladder edit mode, the END instruction is automatically input and cannot be edited.
- The following illustrates how the END and FEND instructions are used properly when a program contains a main routine program, subroutine program, and interrupt program.




**Point**

The END instruction executed while a program is divided into multiple program blocks indicates the end of a program block.
The END instruction executed for END processing is executed at the end of the last executed program registered in the program settings.

## Operation error

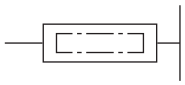
| Error code (SD0/SD8067) | Remarks |
|---|---|
| 3340H | The END instruction is executed before the NEXT instruction after the FOR instruction is executed. |
| 3381H | The END instruction is executed before the RET instruction after the CALL(P) instruction is executed. |
| 33E3H | The END instruction is programmed between FOR-NEXT. |
| 33E4H | The END instruction is programmed between MC-MCR. |
| 33E5H | The END instruction is programmed between STL-RETSTL. |
| 33E7H | The END instruction is programmed between I-IRET. |

# 6.7 Stop Instruction

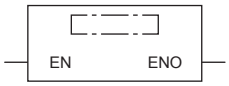
## Stopping the sequence program

### STOP

This instruction resets outputs (Y) and stops operation of the CPU module when the execution command turns ON. (This operation is the same as setting the switch to STOP.)

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=STOP(EN); |

**FBD/LD**




#### Processing details

- This instruction resets outputs (Y) and stops operation of the CPU module when the execution command turns ON. (This operation is the same as setting the switch to STOP.)
- To restart operation of the CPU module after this instruction is executed, return the switch from RUN→STOP and set it to RUN again.

#### Operation error

| Error code (SD0/SD8067) | Remarks |
|---|---|
| 3340H | The STOP instruction is executed before the NEXT instruction is executed after the FOR instruction is executed. |
| 3381H | The STOP instruction is executed before the RET instruction is executed after the CALL(P) or XCALL(P) instruction is executed. |
| 3582H | The STOP instruction is executed before the IRET instruction is executed in the interruption program. |

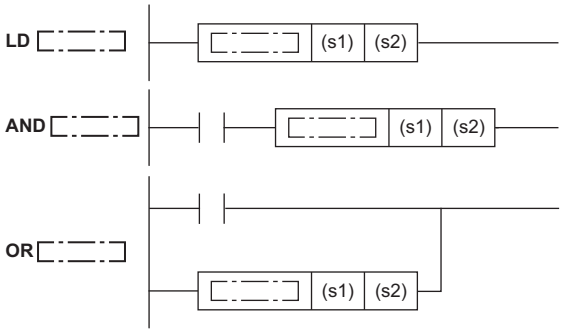
# 7 BASIC INSTRUCTIONS

## 7.1 Comparison Operation Instructions

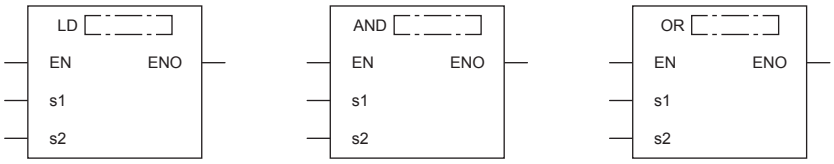
### Comparing 16-bit binary data

#### LD□(_U), AND□(_U), OR□(_U)

These instructions perform a comparison operation between the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2). (Devices are used as NO contacts.)

| Ladder diagram | Structured text |
|---|---|
|  ("=(_U)", "<>(_U)", ">(_U)", "<=(_U)", "<(_U)", ">=(_U)" enters □.) | Not supported |

**FBD/LD**




("_EQ(_U)", "_NE(_U)", "_GT(_U)", "_LE(_U)", "_LT(_U)", "_GE(_U)" enters □.)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | LD□, AND□, OR□ | Comparison data or device where the comparison data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | LD□_U, AND□_U, OR□_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | LD□, AND□, OR□ | Comparison data or device where the comparison data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | LD□_U, AND□_U, OR□_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- These instructions perform a comparison operation between the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2). (Devices are used as NO contacts.)
- The following table lists the comparison operation result of each instruction.

| Instruction symbol | Condition | Result |
|---|---|---|
| =(_U) | (s1) = (s2) | Conductive state |
| <>(_U) | (s1) ≠ (s2) | |
| >(_U) | (s1) > (s2) | |
| <=(_U) | (s1) ≤ (s2) | |
| <(_U) | (s1) < (s2) | |
| >=(_U) | (s1) ≥ (s2) | |
| =(_U) | (s1) ≠ (s2) | Non-conductive state |
| <>(_U) | (s1) = (s2) | |
| >(_U) | (s1) ≤ (s2) | |
| <=(_U) | (s1) > (s2) | |
| <(_U) | (s1) ≥ (s2) | |
| >=(_U) | (s1) < (s2) | |

## Precautions

- When the most significant bit is "1" in the data stored in (s1) or (s2), it is regarded as a negative binary value for comparison. (Excluding unsigned operation)

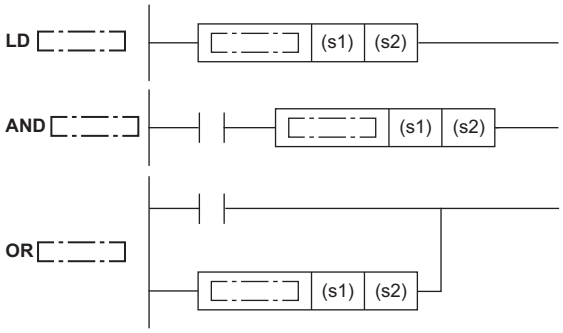
## Operation error

There is no operation error.

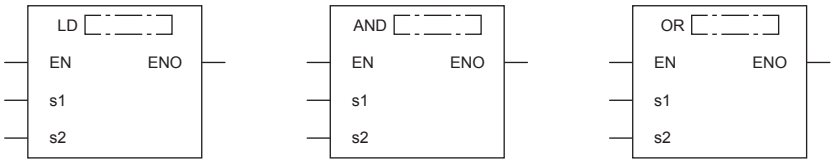
# Comparing 32-bit binary data

## LDD☐(_U), ANDD☐(_U), ORD☐(_U)

These instructions perform a comparison operation between the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2). (Devices are used as NO contacts.)

| Ladder diagram | Structured text |
|---|---|
|  ("D=(_U)", "D<>(_U)", "D>(_U)", "D<=(_U)", "D<(_U)", "D>=(_U)" enters ☐.) | Not supported |

**FBD/LD**




("D_EQ(_U)", "D_NE(_U)", "D_GT(_U)", "D_LE(_U)", "D_LT(_U)", "D_GE(_U)" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | LDD☐, ANDD☐, ORD☐ | Comparison data or head device where the comparison data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | LDD☐_U, ANDD☐_U, ORD☐_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | LDD☐, ANDD☐, ORD☐ | Comparison data or head device where the comparison data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | LDD☐_U, ANDD☐_U, ORD☐_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |

## Processing details

- These instructions perform a comparison operation between the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2). (Devices are used as NO contacts)
- The following table lists the comparison operation results of each instruction.

| Instruction symbol | Condition | Result |
|---|---|---|
| D=(_U) | (s1) = (s2) | Conductive state |
| D<>(_U) | (s1) ≠ (s2) | |
| D>(_U) | (s1) > (s2) | |
| D<=(_U) | (s1) ≤ (s2) | |
| D<(_U) | (s1) < (s2) | |
| D>=(_U) | (s1) ≥ (s2) | |
| D=(_U) | (s1) ≠ (s2) | Non-conductive state |
| D<>(_U) | (s1) = (s2) | |
| D>(_U) | (s1) ≤ (s2) | |
| D<=(_U) | (s1) > (s2) | |
| D<(_U) | (s1) ≥ (s2) | |
| D>=(_U) | (s1) < (s2) | |

## Precautions

- When the most significant bit is "1" in the data stored in (s1) or (s2), it is regarded as a negative binary value for comparison. (Excluding unsigned operation)
- For comparison of 32-bit counter (LC), specify an instruction (LDD=, etc.) that handles 32-bit data. If an instruction (LD=, etc.) that handles 16-bit data is specified, a program error or operation error occurs. (Same applies for index device (LZ) as well.)

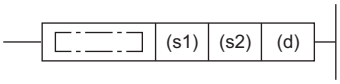
## Operation error

There is no operation error.

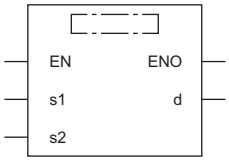
# Comparison output 16-bit binary data

## CMP(P)(_U)

These instructions perform a comparison operation between the 16-bit binary data in the devices specified by (s1) and (s2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=CMP(EN,s1,s2,d);<br>ENO:=CMPP(EN,s1,s2,d); | ENO:=CMP_U(EN,s1,s2,d);<br>ENO:=CMPP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | CMP(P) | Comparison value data or the device where the comparison value data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | CMP(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | CMP(P) | Comparison source data or the device where the comparison source data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | CMP(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | | The starting bit device to which the comparison result is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

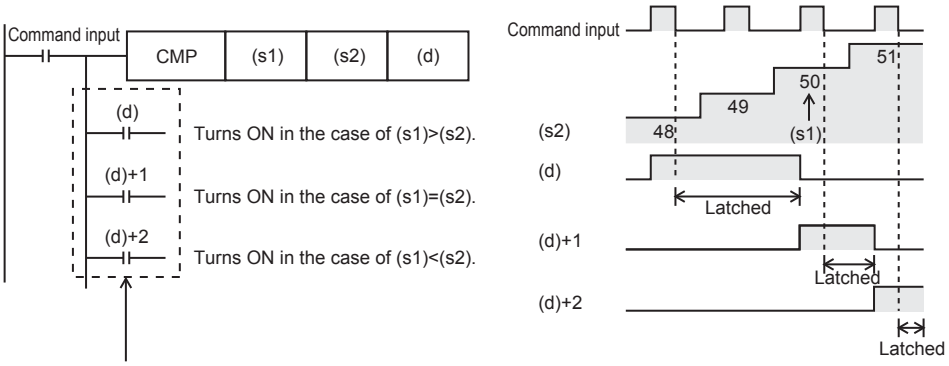
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions perform a comparison operation between the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2) and according to the result (small, equal, large), (d), (d) + 1, or (d) + 2 is turned ON.
- (s1) and (s2) are handled as binary values within the range of above data setting.

• Large and small comparison is executed algebraically.
  • With sign… -10 (FFF6H) < 2 (0002H)
  • Without sign… 32767 (7FFFH) < 65280 (FF00H)




Even if the command input turns OFF and the CMP instruction is not executed, (d) to (d)+2 latches the status just before the command input turns from ON to OFF.

## Precautions

Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

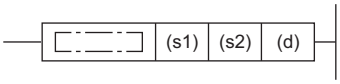
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of 3 points of data starting from the device specified by (d) exceeds said device. |

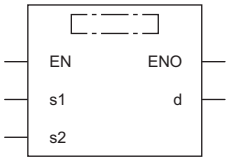
# Comparison output 32-bit binary data

## DCMP(P)(_U)

These instructions perform a comparison operation between the 32-bit binary data in the devices specified by (s1) and (s2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DCMP(EN,s1,s2,d);<br>ENO:=DCMPP(EN,s1,s2,d); | ENO:=DCMP_U(EN,s1,s2,d);<br>ENO:=DCMPP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DCMP(P) | Comparison value data or the head device where the comparison value data is stored | -2147483647 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DCMP(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DCMP(P) | Comparison source data or the head device where the comparison source data is stored | -2147483647 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DCMP(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | | The starting bit device to which the comparison result is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

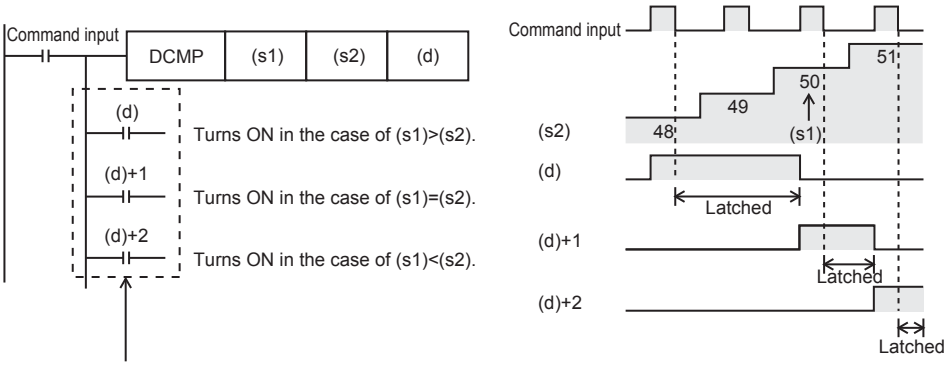
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions perform a comparison operation between the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2) and according to the result (small, equal, large), (d), (d) + 1, or (d) + 2 is turned ON.
- (s1) and (s2) are handled as binary values within the range of above data setting.

- Large and small comparison is executed algebraically.
  - With sign… -125400 (FFFE1628H) < 224566 (00036D36H)
  - Without sign… 16776690 (00FFFDF2H) < 4294967176 (FFFFFF88H)




Even if the command input turns OFF and the DCMP instruction is not executed, (d) to (d)+2 latches the status just before the command input turns from ON to OFF.

## Precautions

Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

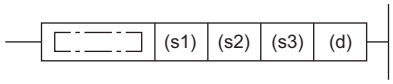
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of 3 points of data starting from the device specified by (d) exceeds said device. |

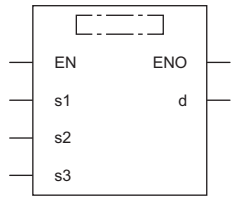
# Comparing 16-bit binary data band

## ZCP(P)(_U)

These instructions perform a comparison operation on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2) with the 16-bit binary data in the device specified by comparison source (s3), and output the comparison result (below, within zone, above) to the device specified by (d) onwards.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=ZCP(EN,s1,s2,s3,d);<br>ENO:=ZCPP(EN,s1,s2,s3,d); | ENO:=ZCP_U(EN,s1,s2,s3,d);<br>ENO:=ZCPP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | ZCP(P) | Lower limit comparison data or the device where the comparison data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZCP(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | ZCP(P) | Upper limit comparison data or the device where the comparison data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZCP(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s3) | ZCP(P) | Comparison source data or the device where the comparison source data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZCP(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | | The starting bit device to which the comparison result is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

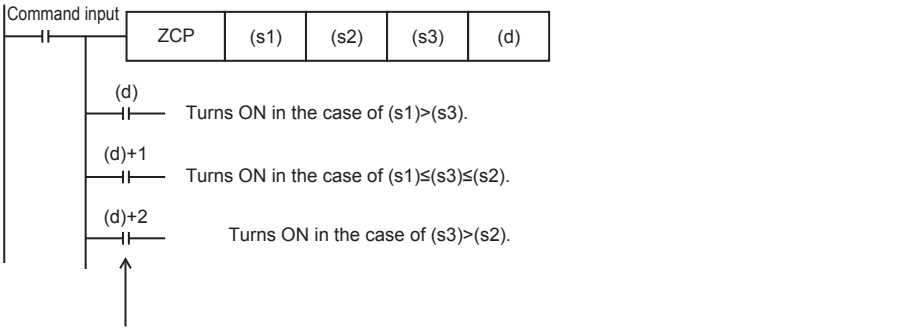
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

- These instructions perform a comparison operation on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2) with the 16-bit binary data in the device specified by comparison source (s3), and according to the comparison result (below, within zone, above), (d), (d) + 1, or (d) + 2 is turned ON. (s1), (s2), and (s3) are handled as binary values within the range of above data setting. Large and small comparison is executed algebraically.
- Large and small comparison is executed algebraically.
  - With sign… -10 (FFF6H) < 2 (0002H) < 10 (000AH)
  - Without sign… 0 (0000H) < 32767 (7FFFH) < 40000 (9C40H)




Even if the command input turns OFF and the ZCP instruction is not executed, (d) to (d)+2 latches the status just before the command input turns from ON to OFF.

## Precautions

- Set (s1) to a value less than (s2).
- Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

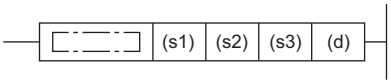
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the 3 points of data starting from the device specified by (d) exceeds said device. |

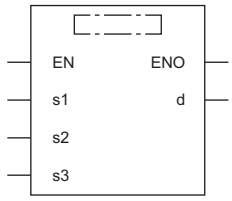
# Comparing 32-bit binary data band

## DZCP(P)(_U)

These instructions perform a comparison operation on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2) with the 32-bit binary data in the device specified by comparison source (s3), and output the comparison result (below, within zone, above) to the device specified by (d) onwards.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DZCP(EN,s1,s2,s3,d);<br>ENO:=DZCPP(EN,s1,s2,s3,d); | ENO:=DZCP_U(EN,s1,s2,s3,d);<br>ENO:=DZCPP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DZCP(P) | Lower limit comparison data or the head device where the comparison data is stored | -2147483647 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZCP(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DZCP(P) | Upper limit comparison data or the head device where the comparison data is stored | -2147483647 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZCP(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s3) | DZCP(P) | Comparison source data or the head device where the comparison source data is stored | -2147483647 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZCP(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | | The starting bit device to which the comparison result is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

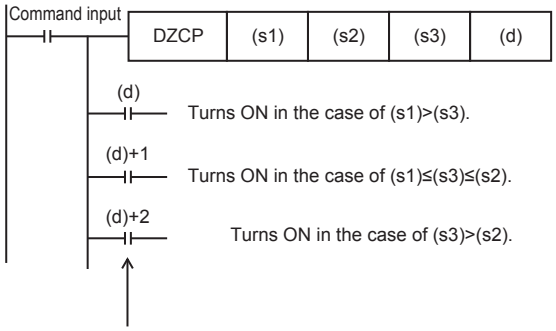
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

- These instructions perform a comparison operation on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2) with the 32-bit binary data in the device specified by comparison source (s3), and according to the comparison result (below, within zone, above), (d), (d) + 1, or (d) + 2 is turned ON. (s1), (s2), and (s3) are handled as binary values within the range of above data setting.
- Large and small comparison is executed algebraically.
  - With sign… -125400 (FFFE1628H) < 22466 (000057C2H) < 1015444 (000F7E94H)
  - Without sign… 0 (00000000H) < 2147483647 (7FFFFFFFH) < 4026531840 (F0000000H)




Even if the command input turns OFF and the DZCP instruction is not executed, (d) to (d)+2 latches the status just before the command input turns from ON to OFF.

## Precautions

- Set (s1) to a value less than (s2).
- Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

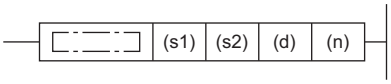
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the 3 points of data starting from the device specified by (d) exceeds said device. |

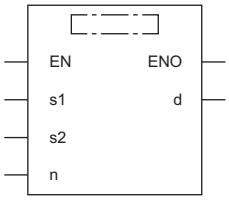
# Comparing 16-bit binary block data

## BKCMP□(P)(_U)

These instructions perform a comparison operation between (n) point(s) of 16-bit binary data in the device starting from the one specified by (s1) and (n) point(s) of 16-bit binary data in the device starting from the one specified by (s2), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  <br>("BKCMP=(P)(_U)", "BKCMP<>(P)(_U)", "BKCMP>(P)(_U)", "BKCMP<=(P)(_U)", "BKCMP<(P)(_U)", "BKCMP>=(P)(_U)" enters □.) | Not supported |

**FBD/LD**




("BKCMP_EQ(P)(_U)", "BKCMP_NE(P)(_U)", "BKCMP_GT(P)(_U)", "BKCMP_LE(P)(_U)", "BKCMP_LT(P)(_U)", "BKCMP_GE(P)(_U)" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | BKCMP□(P) | Comparison data or the device where the comparison data is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BKCMP□(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | BKCMP□(P) | Device where the comparison source data is stored | — | 16-bit signed binary | ANY16_S |
| | BKCMP□(P)_U | | — | 16-bit unsigned binary | ANY16_U |
| (d) | | Head device storing comparison result | — | Bit | ANY_BOOL |
| (n) | | Number of data to be compared | 0 to 65535 | 16-bit unsigned binary | ANY16 |

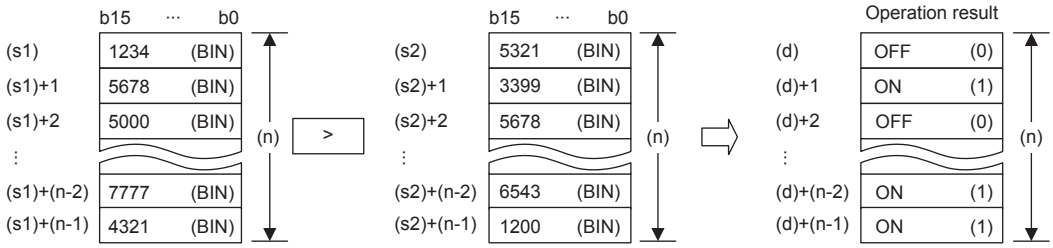
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

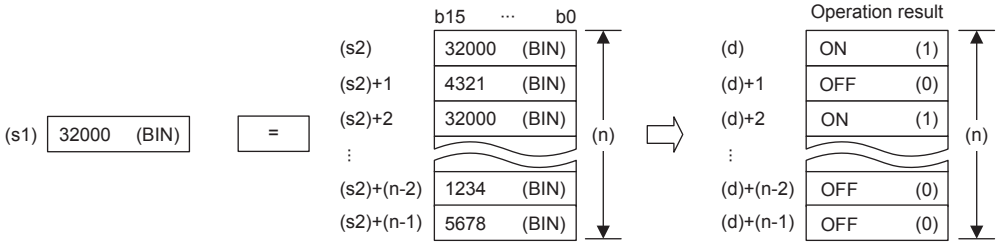
*1 T, ST, C cannot be used.

## Processing details

- These instructions perform a comparison operation between (n) point(s) of 16-bit binary data in the device starting from the one specified by (s1) and (n) point(s) of 16-bit binary data in the device starting from the one specified by (s2), and store the comparison result in (n) point(s) of data starting from the device specified by (d).
- The relevant devices of (n) point(s) of data starting from the device specified by (d) are turned ON when the comparison conditions are met and turned OFF when the comparison conditions are not met.




- Comparison operation is performed in units of 16 bits.
- A constant can be directly specified in (s1).




- The following table lists the comparison operation result of each instruction.

| Instruction symbol | Condition | Result |
|---|---|---|
| BKCMP=(P)(_U) | (s1) = (s2) | On(1) |
| BKCMP<>(P)(_U) | (s1) ≠ (s2) | |
| BKCMP>(P)(_U) | (s1) > (s2) | |
| BKCMP<=(P)(_U) | (s1) ≤ (s2) | |
| BKCMP<(P)(_U) | (s1) < (s2) | |
| BKCMP>=(P)(_U) | (s1) ≥ (s2) | |
| BKCMP=(P)(_U) | (s1) ≠ (s2) | Off(0) |
| BKCMP<>(P)(_U) | (s1) = (s2) | |
| BKCMP>(P)(_U) | (s1) ≤ (s2) | |
| BKCMP<=(P)(_U) | (s1) > (s2) | |
| BKCMP<(P)(_U) | (s1) ≥ (s2) | |
| BKCMP>=(P)(_U) | (s1) < (s2) | |

- When the comparison operation result is all ON (1) in all (n) point(s) starting from (d), SM704 and SM8090 (block comparison signal) turns ON.

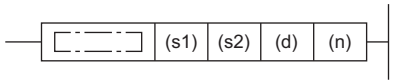
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) point(s) starting from the device specified by (s1), (s2), and (d) exceeds said device. |
| 2821H | When (d) specifies "D□.b", the data register of (d) and the (n) point(s) of data starting from the device specified by (s1) overlap. |
| | When (d) specifies "D□.b", the data register of (d) and the (n) point(s) of data starting from the device specified by (s2) overlap. |

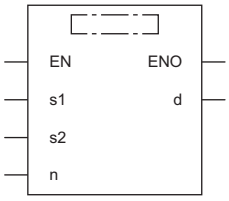
# Comparing 32-bit binary block data

## DBKCMP□(P)(_U)

These instructions perform a comparison operation between the (n) point(s) of 32-bit binary data starting from the device specified by (s1) and the (n) point(s) of 32-bit binary data starting from the device specified by (s2), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  <br>("DBKCMP=(P)(_U)", "DBKCMP<>(P)(_U)", "DBKCMP>(P)(_U)", "DBKCMP<=(P)(_U)", "DBKCMP<(P)(_U)", "DBKCMP>=(P)(_U)" enters □.) | Not supported |

**FBD/LD**




("DBKCMP_EQ(P)(_U)", "DBKCMP_NE(P)(_U)", "DBKCMP_GT(P)(_U)", "DBKCMP_LE(P)(_U)", "DBKCMP_LT(P)(_U)", "DBKCMP_GE(P)(_U)" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DBKCMP□(P) | Comparison data or the head device where the comparison data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBKCMP□(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DBKCMP□(P) | Head device where the comparison source data is stored | — | 32-bit signed binary | ANY32_S |
| | DBKCMP□(P)_U | | — | 32-bit unsigned binary | ANY32_U |
| (d) | | Head device storing comparison result | — | Bit | ANY_BOOL |
| (n) | | Number of data to be compared | 0 to 65535 | 16-bit unsigned binary | ANY16 |

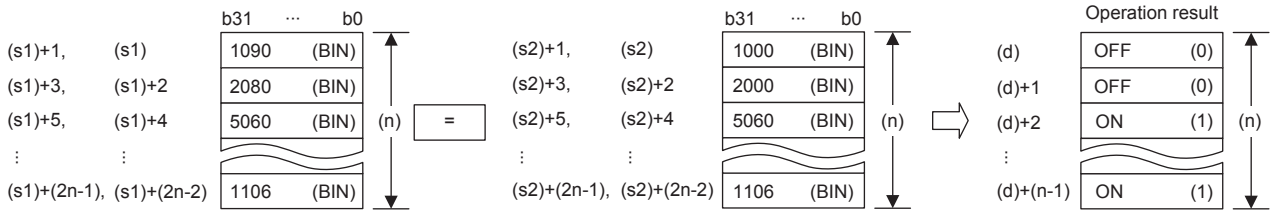
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | ○ | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

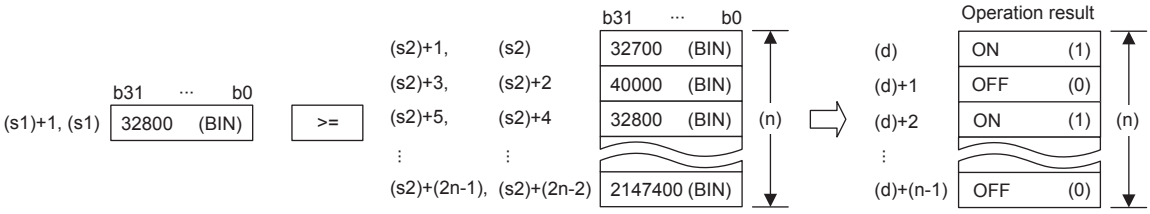
*1 T, ST, C cannot be used.

## Processing details

- These instructions perform a comparison operation between (n) point(s) of 32-bit binary data starting from the device specified by (s1) and (n) point(s) of 32-bit binary data starting from the device specified by (s2), and store the comparison result in (n) point(s) of data starting from the device specified by (d).
- The relevant (n) point(s) of data starting from the device specified by (d) are turned ON when the comparison conditions are met and turned OFF when the comparison conditions are not met.




- Comparison operation is performed in units of 32 bits.
- A constant can be directly specified in (s1).




- (d) is specified outside the device range of (n) point(s) of data starting from the one specified by (s1) and outside the device range of (n) point(s) of data starting from the one specified by (s2).
- The following table lists the comparison operation result of each instruction.

| Instruction symbol | Condition | Result |
|---|---|---|
| DBKCMP=(P)(_U) | (s1) = (s2) | On(1) |
| DBKCMP<>(P)(_U) | (s1) ≠ (s2) | |
| DBKCMP>(P)(_U) | (s1) > (s2) | |
| DBKCMP<=(P)(_U) | (s1) ≤ (s2) | |
| DBKCMP<(P)(_U) | (s1) < (s2) | |
| DBKCMP>=(P)(_U) | (s1) ≥ (s2) | |
| DBKCMP=(P)(_U) | (s1) ≠ (s2) | Off(0) |
| DBKCMP<>(P)(_U) | (s1) = (s2) | |
| DBKCMP>(P)(_U) | (s1) ≤ (s2) | |
| DBKCMP<=(P)(_U) | (s1) > (s2) | |
| DBKCMP<(P)(_U) | (s1) ≥ (s2) | |
| DBKCMP>=(P)(_U) | (s1) < (s2) | |

- When the comparison operation result is all ON (1) in all (n) point(s) starting from (d), SM704 and SM8090 (block comparison signal) turns ON.

## Precautions

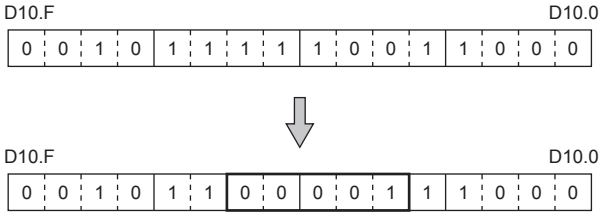
If a 32-bit counter (high-speed counter included) is used, make sure to compare using the 32-bit operation (DBKCMP=, DBKCMP>, DBKCMP<, etc.).

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) × 2 points of data starting from the device specified by (s1) and (s2) or the (n) point(s) of data starting from the device specified by (d) exceeds said device. |
| 2821H | When (d) specifies "D□.b", the (n) point(s) of data starting from the device specified by (d) and the device range of the (n) × 2 points of data starting from the device specified by (s1) overlap. |
| | When (d) specifies "D□.b", the (n) point(s) of data starting from the device specified by (d) and the device range of the (n) × 2 points of data starting from the device specified by (s2) overlap. |

**Point**

When bit is specified for word device, devices other than the bit-specified word devices where operation result is stored will not change.



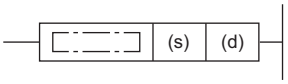
# 7.2 Arithmetic Operation Instructions

## Adding 16-bit binary data

+(P)(_U) instruction and ADD(P)(_U) instruction can be used for addition of 16-bit binary data.

### +(P)(_U) [using two operands]

These instructions add the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

#### Setting data

##### ■Descriptions, ranges, and data types

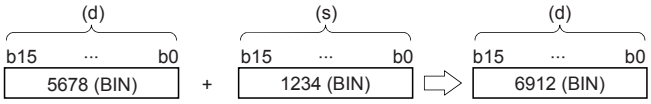
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | +(P) | Addend data or the device where the data that is added to another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | +(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | +(P) | Device where the data to which another is added is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | +(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions add the 16-bit binary data in the device specified by (s) to the 16-bit binary data in the device specified by (d), and store the addition result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of +(P)

K32767 (7FFFH) + K2 (0002H) → K-32767 (8001H) ····· Because the highest bit is 1, the value is negative.

K-32768 (8000H) + K-2 (FFFEH) → K32766 (7FFEH) ····· Because the highest bit is 0, the value is positive.

In case of +(P)(_U)

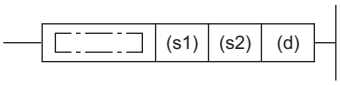
K65535 (FFFFH) + K2 (0002H) → K1 (0001H)

#### Operation error

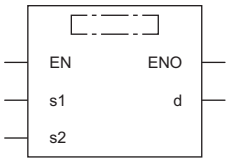
There is no operation error.

## +(P)(_U) [using three operands]

These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=PLUS(EN,s1,s2,d);<br>ENO:=PLUSP(EN,s1,s2,d); | ENO:=PLUS_U(EN,s1,s2,d);<br>ENO:=PLUSP_U(EN,s1,s2,d); |

**FBD/LD**




("PLUS", "PLUSP", "PLUS_U", "PLUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

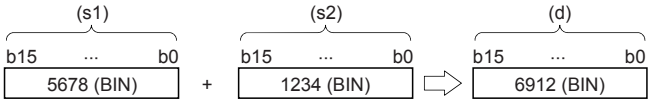
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | +(P) | Augend data or the device where the data to which another is added is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | +(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | +(P) | Addend data or the device where the data that is added to another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | +(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | +(P) | Device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | +(P)_U | | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the addition result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of +(P)

K32767 (7FFFH) + K2 (0002H) → K-32767 (8001H) ・・・・・・・・ Because the highest bit is 1, the value is negative.

K-32768 (8000H) + K-2 (FFFEH) → K32766 (7FFEH) ・・・・・・・・ Because the highest bit is 0, the value is positive.

In case of +(P)(_U)

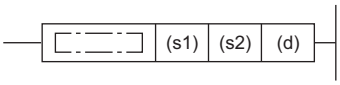
K65535 (FFFFH) + K2 (0002H) → K1 (0001H)

### Operation error

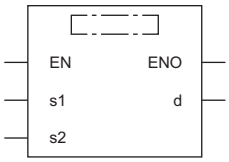
There is no operation error.

## ADD(P)(_U)

These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text*1 | |
|---|---|---|
|  | ENO:=ADDP(EN,s1,s2,d); | ENO:=ADD_U(EN,s1,s2,d);<br>ENO:=ADDP_U(EN,s1,s2,d); |

**FBD/LD*1**




("ADDP", "ADD_U", "ADDP_U" enters □.)

*1 The ADD instruction is not supported by the ST language and the FBD/LD language. Use ADD of the standard function.
☞ Page 872 ADD(_E)

### Setting data

#### ■Descriptions, ranges, and data types

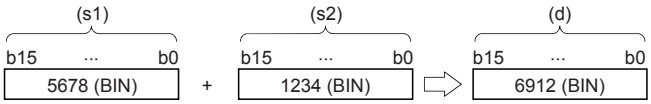
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | ADD(P) | Addend data or the device where the data that is added to another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ADD(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | ADD(P) | Addend data or the device where the data that is added to another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ADD(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | ADD(P) | Device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | ADD(P)_U | | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

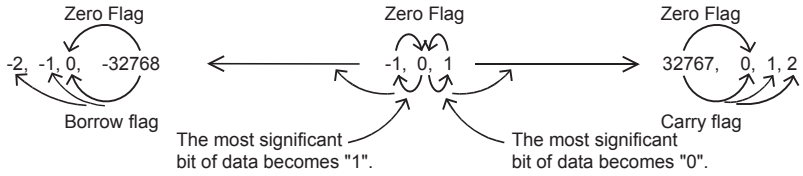
### Processing details

- These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the addition result in the device specified by (d).



### ■Relationship between the flag operation and the sign (positive or negative) of a numeric value

| Device | Name | Description |
|---|---|---|
| SM700, SM8022 | Carry | When the operation result exceeds the upper limit of the data setting range, the carry flag is turned ON. |
| SM8020 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8021 | Borrow | When the operation result is less than the lower limit of the data setting range, the borrow flag is turned ON. |




## Precautions

### ■When specifying the same device in the source and destination

The same device number can be specified for both the source and the destination. In this case, note that the addition result changes in every operation cycle if a continuous operation type ADD instruction is used.


### ■Difference between ADD(P) instruction, +(P) instruction, and INC(P) instruction in a program for adding "+1"

When ADD(P) instruction is used to add 1 to the contents of D0 every time X1 turns from OFF to ON, ADD(P) instruction is similar to +(P) instruction and INC(P) instruction described later except for the contents shown in the table below

| | | ADD(P) instruction | +(P) instruction, INC(P) instruction |
|---|---|---|---|
| Flag (zero, borrow or carry) | | Operates | Does not operate |
| Operation result | (s)+1=(d) | +32767 → 0 → +1 → +2 →… | +32767 → -32768 → -32767 →… |

## Operation error

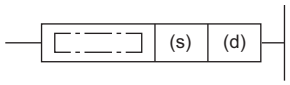
There is no operation error.

# Subtracting 16-bit binary data

-(P)(_U) instruction and SUB(P)(_U) instruction can be used for subtraction of 16-bit binary data.

## -(P)(_U) [using two operands]

These instructions subtract the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

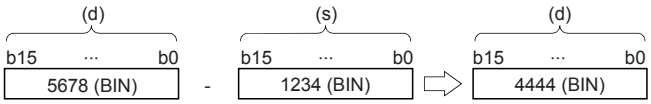
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | -(P) | Subtrahend data or the device where the data to be subtracted from another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | -(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | -(P) | Device where the data from which another is to be subtracted is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | -(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the subtraction result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of -(P)

K32768 (8000H) - K2 (0002H) → K32766 (7FFEH) ・・・・・・・・ Because the highest bit is 0, the value is positive.

K32767 (7FFFH) - K-2 (FFFEH) → K-32767 (8001H) ・・・・・・・・ Because the highest bit is 1, the value is negative.

In case of -(P)(_U)

K0 (0000H) - K1 (0001H) → K65535 (FFFFH)

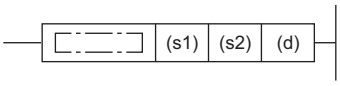
K0 (0000H) - K65535 (FFFFH) → K1 (0001H)

### Operation error

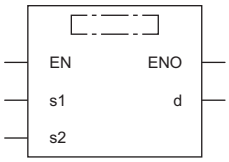
There is no operation error.

## -(P)(_U) [using three operands]

These instructions subtract the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=MINUS(EN,s1,s2,d);<br>ENO:=MINUSP(EN,s1,s2,d); | ENO:=MINUS_U(EN,s1,s2,d);<br>ENO:=MINUSP_U(EN,s1,s2,d); |

**FBD/LD**




("MINUS", "MINUSP", "MINUS_U", "MINUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

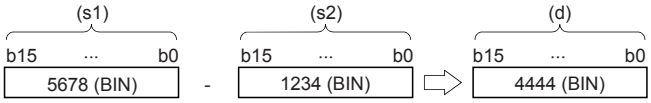
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | -(P) | Minuend data or the device where the data from which another is to be subtracted is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | -(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | -(P) | Subtrahend data or the device where the data to be subtracted from another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | -(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | -(P) | Device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | -(P)_U | | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the subtraction result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of -(P)

K-32768 (8000H) - K2 (0002H) → K32766 (7FFEH) ····· Because the highest bit is 0, the value is positive.

K32767 (7FFFH) - K-2 (FFFEH) → K-32767 (8001H) ····· Because the highest bit is 1, the value is negative.

In case of -(P)(_U)

K0 (0000H) - K1 (0001H) → K65535 (FFFFH)

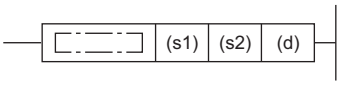
K0 (0000H) - K65535 (FFFFH) → K1 (0001H)

## Operation error

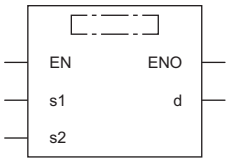
There is no operation error.

## SUB(P)(_U)

These instructions subtract the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text*1 | |
|---|---|---|
|  | ENO:=SUBP(EN,s1,s2,d); | ENO:=SUB_U(EN,s1,s2,d);<br>ENO:=SUBP_U(EN,s1,s2,d); |

**FBD/LD*1**




("SUBP", "SUB_U", "SUBP_U" enters □.)

*1 The SUB instruction is not supported by the ST language and the FBD/LD language. Use SUB of the standard function.
☞ Page 876 SUB(_E)

### Setting data

#### ■Descriptions, ranges, and data types

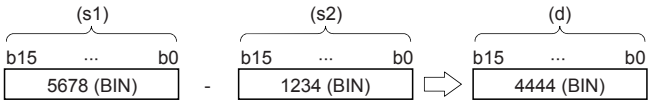
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | SUB(P) | Subtrahend data or the device where the data to be subtracted from another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | SUB(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | SUB(P) | Subtrahend data or the device where the data to be subtracted from another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | SUB(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | SUB(P) | Device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | SUB(P)_U | | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

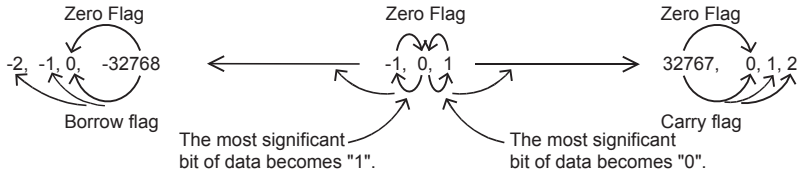
### Processing details

- These instructions subtract the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the subtraction result in the device specified by (d).



### ■Relationship between the flag operation and the sign (positive or negative) of a numeric value

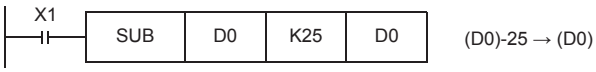
| Device | Name | Description |
|---|---|---|
| SM700, SM8022 | Carry | When the operation result exceeds the upper limit of the data setting range, the carry flag is turned ON. |
| SM8020 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8021 | Borrow | When the operation result is less than the lower limit of the data setting range, the borrow flag is turned ON. |




## Precautions

### ■When specifying the same device in the source and destination

The same device number can be specified for both the source and the destination. In this case, note that the subtraction result changes in every operation cycle if a continuous operation type SUB instruction is used.




### ■Difference between SUB(P) instruction, -(P) instruction, and DEC(P) instruction in a program for subtracting "-1"

When SUB(P) instruction is used to subtract 1 from the contents of D0 every time X1 turns from OFF to ON, SUB(P) instruction is similar to -(P) instruction and DEC(P) instruction described later except for the contents shown in the table below

| | | SUB(P) instruction | -(P) instruction, DEC(P) instruction |
|---|---|---|---|
| Flag (zero, borrow or carry) | | Operates | Does not operate |
| Operation result | (s)-1=(d) | -32768 → 0 → -1 → -2 →… | -32768 → +32767 → +32766 →… |

## Operation error

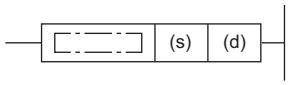
There is no operation error.

# Adding 32-bit binary data

D+(P)(_U) instruction and DADD(P)(_U) instruction can be used for addition of 32-bit binary data.

## D+(P)(_U) [using two operands]

These instructions add the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

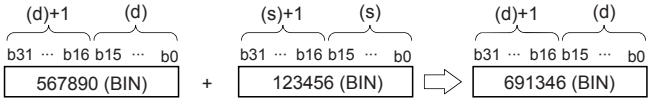
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | D+(P) | Addend data or the head device where the data that is added to another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D+(P) | Head device where the data to which another is added is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions add the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the addition result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of D+(P)

K2147483647 (7FFFFFFFH) + K2 (00000002H) → K-2147483647 (80000001H) ····· Because the highest bit is 1, the value is negative.

K-2147483648 (80000000H) + K-2 (FFFFFFFEH) → K2147483646 (7FFFFFFEH) ····· Because the highest bit is 0, the value is positive.

In case of D+(P)(_U)

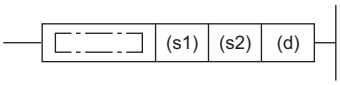
K4294967295 (FFFFFFFFH) + K2 (00000002H) → K1 (00000001H)

### Operation error

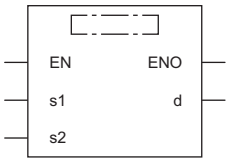
There is no operation error.

## D+(P)(_U) [using three operands]

These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DPLUS(EN,s1,s2,d);<br>ENO:=DPLUSP(EN,s1,s2,d); | ENO:=DPLUS_U(EN,s1,s2,d);<br>ENO:=DPLUSP_U(EN,s1,s2,d); |

**FBD/LD**




("DPLUS", "DPLUSP", "DPLUS_U", "DPLUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

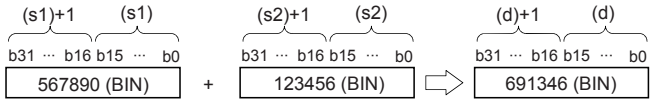
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | D+(P) | Augend data or the head device where the data to which another is added is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | D+(P) | Addend data or the head device where the data that is added to another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D+(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | D+(P)_U | | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the addition result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of D+(P)

K2147483647 (7FFFFFFFH) + K2 (00000002H) → K-2147483647 (80000001H) ····· Because the highest bit is 1, the value is negative.

K-2147483648 (80000000H) + K-2 (FFFFFFFEH) → K2147483646 (7FFFFFFEH) ····· Because the highest bit is 0, the value is positive.

In case of D+(P)(_U)

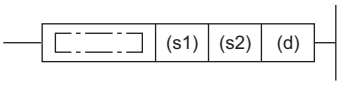
K4294967295 (FFFFFFFFH) + K2 (00000002H) → K1 (00000001H)

### Operation error

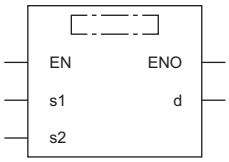
There is no operation error.

# DADD(P)(_U)

These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DADD(EN,s1,s2,d);<br>ENO:=DADDP(EN,s1,s2,d); | ENO:=DADD_U(EN,s1,s2,d);<br>ENO:=DADDP_U(EN,s1,s2,d); |

**FBD/LD**




## Setting data

### ■Descriptions, ranges, and data types

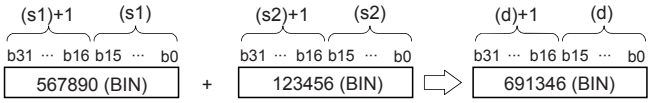
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DADD(P) | Addend data or the head device where the data that is added to another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DADD(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DADD(P) | Addend data or the head device where the data that is added to another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DADD(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DADD(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | DADD(P)_U | | — | 32-bit unsigned binary | ANY32_U |

### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

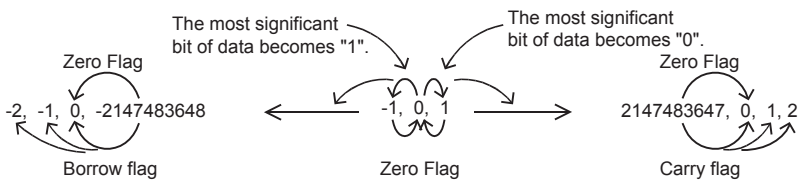
## Processing details

- These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the addition result in the device specified by (d).




### ■Relationship between the flag operation and the sign (positive or negative) of a numeric value

| Device | Name | Description |
|---|---|---|
| SM700, SM8022 | Carry | When the operation result exceeds the upper limit of the data setting range, the carry flag is turned ON. |
| SM8020 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8021 | Borrow | When the operation result is less than the lower limit of the data setting range, the borrow flag is turned ON. |



## Precautions

### ■When DADD instruction is used

When specifying word devices, a device for the lower-order 16-bits is specified first, and then a word device with the next device number is set for the higher-order 16 bits. To prevent number overlap, it is recommended to always specify an even number.

### ■When specifying the same device in the source and destination

The same device number can be specified for both the source and the destination. In this case, note that the addition result changes in every operation cycle if a continuous operation type ADD instruction is used.


### ■Difference between DADD(P) instruction, D+(P) instruction, and DINC(P) instruction in a program for adding "+1"

When DADD(P) instruction is used to add 1 to the contents of D0 every time X1 turns from OFF to ON, DADD(P) instruction is similar to D+(P) instruction and DINC(P) instruction described later except for the contents shown in the table below.

| | | DADD(P) instruction | D+(P) instruction, DINC(P) instruction |
|---|---|---|---|
| Flag (zero, borrow or carry) | | Operates | Does not operate |
| Operation result | (s)+1=(d) | +2147483647 → 0 → +1 → +2 →… | +2147483647 → -2147483648 → -2147483647 →… |

## Operation error

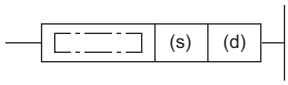
There is no operation error.

# Subtracting 32-bit binary data

D-(P)(_U) instruction and DSUB(P)(_U) instruction can be used for subtraction of 32-bit binary data.

## D-(P)(_U) [using two operands]

These instructions subtract the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

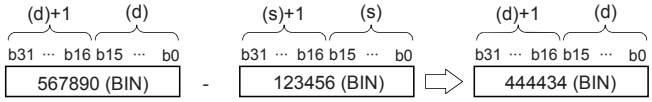
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | D-(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D-(P) | Head device where the data from which another is to be subtracted is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions subtract the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the subtraction result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of D-(P)

K-2147483648 (80000000H) - K2 (00000002H) → K2147483646 (7FFFFFFEH) ····· Because the highest bit is 0, the value is positive.

K2147483647 (7FFFFFFFH) - K-2 (FFFFFFFEH) → K-2147483647 (80000001H) ····· Because the highest bit is 1, the value is negative.

In case of D-(P)(_U)

K0 (00000000H) - K1 (00000001H) → K4294967295 (FFFFFFFFH)

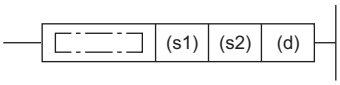
K0 (00000000H) - K4294967295 (FFFFFFFFH) → K1 (00000001H)

### Operation error

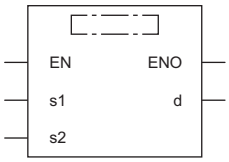
There is no operation error.

## D-(P)(_U) [using three operands]

These instructions subtract the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DMINUS(EN,s1,s2,d);<br>ENO:=DMINUSP(EN,s1,s2,d); | ENO:=DMINUS_U(EN,s1,s2,d);<br>ENO:=DMINUSP_U(EN,s1,s2,d); |

**FBD/LD**




("DMINUS", "DMINUSP", "DMINUS_U", "DMINUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

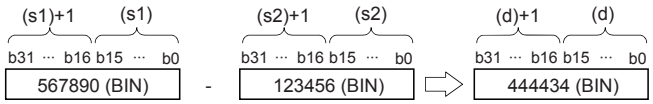
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | D-(P) | Minuend data or the head device where the data from which another is to be subtracted is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | D-(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D-(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | D-(P)_U | | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions subtract the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the subtraction result in the device specified by (d).




- When underflow or overflow occurs in the operation result, the following processing is executed. In this case, the carry flag (SM700, SM8022) does not turn ON.

In case of D-(P)

K-2147483648 (80000000H) - K2 (00000002H) → K2147483646 (7FFFFFFEH) ..... Because the highest bit is 0, the value is positive.

K2147483647 (7FFFFFFFH) - K-2 (FFFFFFFEH) → K-2147483647 (80000001H) ..... Because the highest bit is 1, the value is negative.

In case of D-(P)(_U)

K0 (00000000H) - K1 (00000001H) → K4294967295 (FFFFFFFFH)

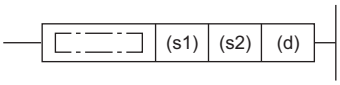
K0 (00000000H) - K4294967295 (FFFFFFFFH) → K1 (00000001H)

## Operation error

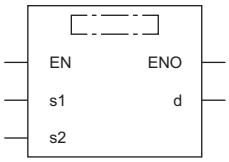
There is no operation error.

## DSUB(P)(_U)

These instructions subtract the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DSUB(EN,s1,s2,d);<br>ENO:=DSUBP(EN,s1,s2,d); | ENO:=DSUB_U(EN,s1,s2,d);<br>ENO:=DSUBP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

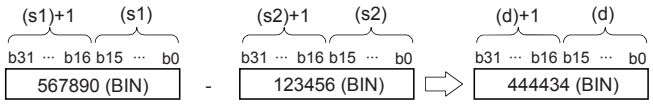
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DSUB(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DSUB(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DSUB(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DSUB(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DSUB(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | DSUB(P)_U | | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

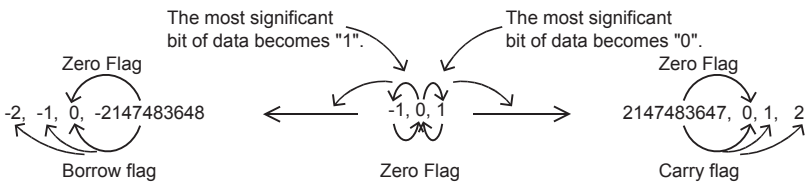
### Processing details

- These instructions subtract the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the subtraction result in the device specified by (d).




#### ■Relationship between the flag operation and the sign (positive or negative) of a numeric value

| Device | Name | Description |
|---|---|---|
| SM700, SM8022 | Carry | When the operation result exceeds the upper limit of the data setting range, the carry flag is turned ON. |
| SM8020 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8021 | Borrow | When the operation result is less than the lower limit of the data setting range, the borrow flag is turned ON. |



## Precautions

### ■When the DSUB instruction is used

When specifying word devices, a device is specified for the lower-order 16-bits first, and then a word device with the next device number is set for the higher-order 16 bits. To prevent number overlap, it is recommended to always specify an even number.

### ■When specifying the same device in the source and destination

The same device number can be specified for both the source and the destination. In this case, note that the subtraction result changes in every operation cycle if a continuous operation type SUB instruction is used.


### ■Difference between DSUB(P) instruction, D-(P) instruction, and DDEC(P) instruction in a program for subtracting "-1"

When DSUB(P) instruction is used to subtract 1 from the contents of D0 every time X1 turns from OFF to ON, SUB(P) instruction is similar to D-(P) instruction and DDEC(P) instruction described later except for the contents shown in the table below:

| | | DSUB(P) instruction | D-(P) instruction, DDEC(P) instruction |
|---|---|---|---|
| Flag (zero, borrow or carry) | | Operates | Does not operate |
| Operation result | (s)-1=(d) | -2147483648 → 0 → -1 → -2 →… | -2147483648 → +2147483647 → +2147483646 →… |

## Operation error

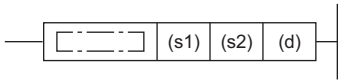
There is no operation error.

# Multiplying 16-bit binary data

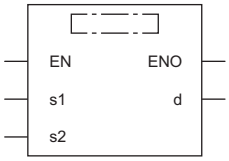
*(P)(_U) instruction and MUL(P)(_U) instruction can be used for multiplication of 16-bit binary data.

## *(P)(_U)

These instructions multiply the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("MULTI", "MULTIP", "MULTI_U", "MULTIP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

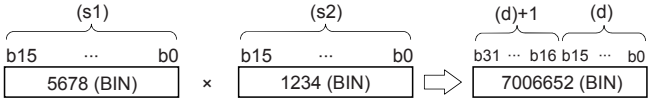
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | *(P) | Multiplicand data or the device where the data to be multiplied by another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | *(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | *(P) | Multiplier data or the device where the data by which another is to be multiplied is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | *(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | *(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | *(P)_U | | | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions multiply the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the multiplication result in the device specified by (d).




- When (d) is a bit device, lower-order bit is specified first.

Ex.

Multiplication result when (d) is a bit device

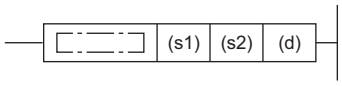
- K1 ... Lower 4 bits (b0 to b3)
- K4 ... Lower 16 bits (b0 to b15)
- K8 ... Lower 32 bits (b0 to b31)

## Operation error

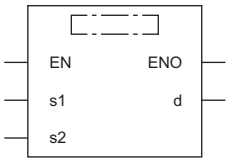
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds said device range. |

## MUL(P)(_U)

These instructions multiply the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text*1 | |
|---|---|---|
|  | ENO:=MULP(EN,s1,s2,d); | ENO:=MUL_U(EN,s1,s2,d);<br>ENO:=MULP_U(EN,s1,s2,d); |

**FBD/LD*1**




("MULP", "MUL_U", "MULP_U" enters □.)

*1 The MUL instruction is not supported by the ST language and the FBD/LD language. Use MUL of the standard function.
☞ Page 874 MUL(_E)

### Setting data

#### ■Descriptions, ranges, and data types

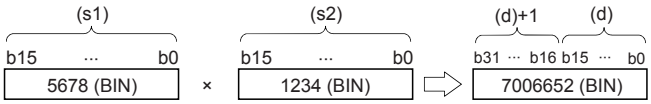
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | MUL(P) | Multiplicand data or the device where the data to be multiplied by another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | MUL(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | MUL(P) | Multiplier data or the device where the data by which another is to be multiplied is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | MUL(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | MUL(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | MUL(P)_U | | | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

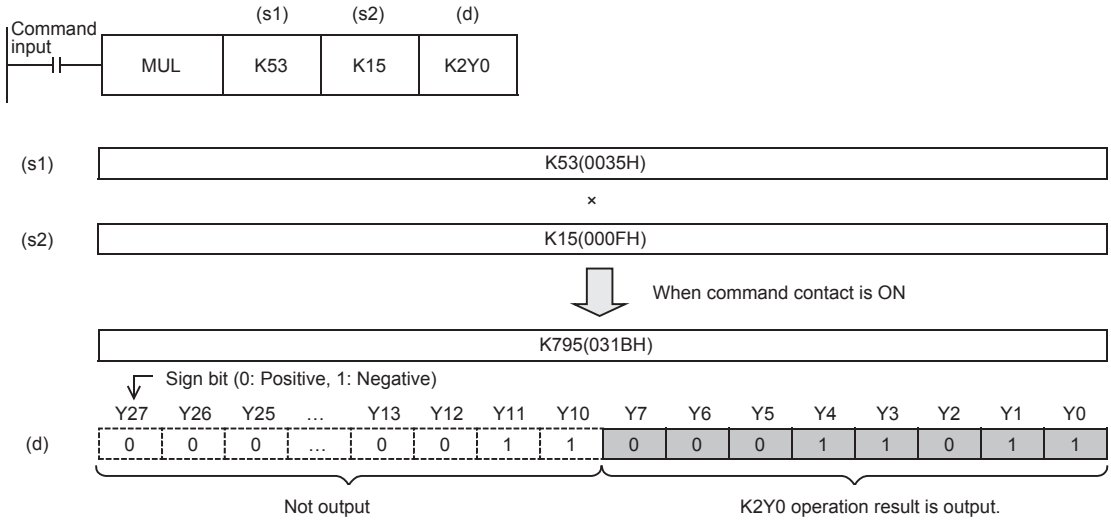
- These instructions multiply the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the multiplication result in the device specified by (d).



• Nibble can be specified ranging from K1 to K8 for (d).

**Ex.**

For example, when K2 is specified, only the lower-order 8 bits can be obtained out of the product (32 bits).




## ■Related flag

| Device | Name | Description |
|---|---|---|
| SM8304 | Zero | When the operation result is 0, the zero flag is turned ON. |

## Operation error

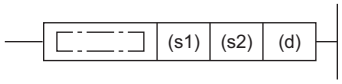
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds said device range. |

# Dividing 16-bit binary data

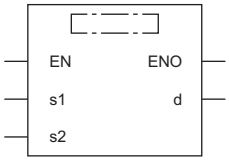
/(P)(_U) instruction and DIV(P)(_U) instruction can be used for division of 16-bit binary data.

## /(P)(_U)

These instructions divide the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DIVISON", "DIVISIONP", "DIVISION_U", "DIVISIONP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | /(P) | Dividend data or the device where the data to be divided by another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | /(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | /(P) | Divisor data or the device where the data by which another is to be divided is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | /(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | /(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | /(P)_U | | | 32-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions divide the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the division result in the device specified by (d).

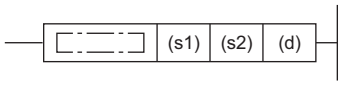

- For the division result, 32-bit is used for word device to store the quotient and remainder and 16-bit is used for bit device to store quotient only.
  - Quotient…… Stored in the lower 16 bits.
  - Remainder…… Stored in the upper 16 bits. (This data can be stored for word device only.)

## Operation error

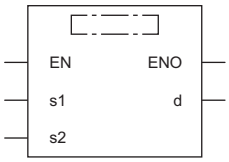
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |
| 3400H | 0 is specified for (s2) value. |
| 3403H | The operation result exceeds 32767, in case of signed operation. |

## DIV(P)(_U)

These instructions divide the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text*1 | |
|---|---|---|
|  | ENO:=DIVP(EN,s1,s2,d); | ENO:=DIV_U(EN,s1,s2,d);<br>ENO:=DIVP_U(EN,s1,s2,d); |

**FBD/LD*1**




("DIVP", "DIV_U", "DIVP_U" enters □.)

*1 The DIV instruction is not supported by the ST language and the FBD/LD language. Use DIV of the standard function.
☞ Page 878 DIV(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DIV(P) | Dividend data or the device where the data to be divided by another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | DIV(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | DIV(P) | Divisor data or the device where the data by which another is to be divided is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | DIV(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | DIV(P) | Head device for storing the operation result (quotient, remainder) | — | 32-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | DIV(P)_U | | | 32-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions divide the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the division result in the device specified by (d).


- Two devices in total starting from the one specified by (d) are used to store the division result. Make sure that these two devices are not used for another control.
  - Quotient…… Stored in the lower 16 bits.
  - Remainder…… Stored in the upper 16 bits.

### ■Related flag

| Device | Name | Description |
|---|---|---|
| SM700 | Carry | When the operation result of the signed operation exceeds 32767, the carry flag is turned ON. |
| SM8304 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8306 | Carry | When the operation result of the signed operation exceeds 32767, the carry flag is turned ON. |

## Precautions

### ■Operation result

- The most significant bit of the quotient and remainder indicates the sign (positive: 0, negative: 1), respectively.
- The quotient is negative when either (s1) or (s2) is negative. The remainder is negative when the (s1) is negative.

### ■Device specified by (d)

- The remainder is not obtained when a bit device is specified with nibble specification.

## Operation error

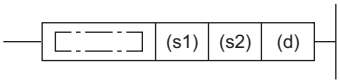
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |
| 3400H | 0 is specified for (s2) value. |
| 3403H | The data type of the data setting is signed data and the operation result exceeds 32767. |

# Multiplying 32-bit binary data

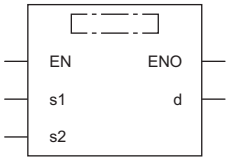
D*(P)(_U) instruction and DMUL(P)(_U) instruction can be used for multiplication of 32-bit binary data.

## D*(P)(_U)

These instructions multiply the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DMULTI", "DMULTIP", "DMULTI_U", "DMULTIP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | D*(P) | Multiplicand data or the head device where the data to be multiplied by another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D*(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | D*(P) | Multiplier data or the head device where the data by which another is to be multiplied is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D*(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D*(P) | Head device for storing the operation result | — | 64-bit signed binary | ANY32_S_ARRAY (Number of elements: 2) |
| | D*(P)_U | | | 64-bit unsigned binary | ANY32_U_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions multiply the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the multiplication result in the device specified by (d).


- When (d) is a bit device, only the lower 32 bits of the multiplication result are stored and the upper 32 bits cannot be specified. If the upper 32 bits data of the multiplication operation result are required, temporarily store the result in a word device, and transfer the data stored in word device ((d)+2) and ((d)+3) to the specified bit devices.

Ex.

Multiplication result when (d) is a bit device

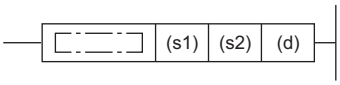
- K1 ... Lower 4 bits (b0 to b3)
- K4 ... Lower 16 bits (b0 to b15)
- K8 ... Lower 32 bits (b0 to b31)

## Operation error

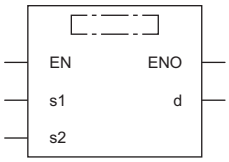
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |

## DMUL(P)(_U)

These instructions multiply the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DMUL(EN,s1,s2,d);<br>ENO:=DMULP(EN,s1,s2,d); | ENO:=DMUL_U(EN,s1,s2,d);<br>ENO:=DMULP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

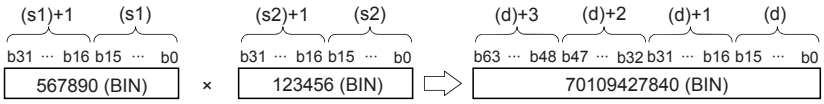
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DMUL(P) | Multiplicand data or the head device where the data to be multiplied by another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DMUL(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DMUL(P) | Multiplier data or the head device where the data by which another is to be multiplied is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DMUL(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DMUL(P) | Head device for storing the operation result | — | 64-bit signed binary | ANY32_S_ARRAY (Number of elements: 2) |
| | DMUL(P)_U | | | 64-bit unsigned binary | ANY32_U_ARRAY (Number of elements: 2) |

#### ■Applicable devices

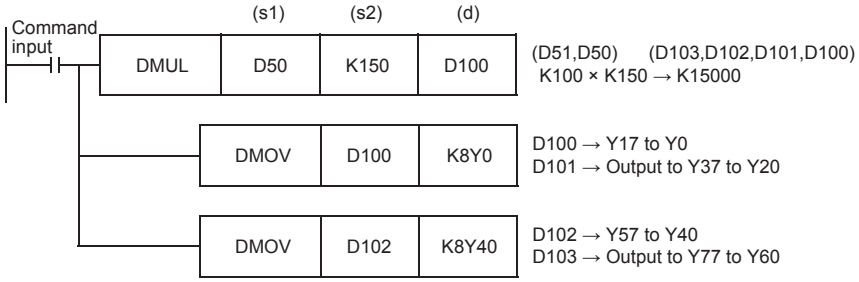
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

## Processing details

- These instructions multiply the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the multiplication result in the device specified by (d).




- When nibble is specified ranging from K1 to K8 for (d), the result is obtained only for the lower-order 32 bits, and is not obtained for the higher-order 32 bits. Transfer the data to word devices once, then execute the operation.




### ■Related flag

| Device | Name | Description |
|---|---|---|
| SM8304 | Zero | When the operation result is 0, the zero flag is turned ON. |

## Precautions

- Even if word devices are used, the operation result (64 bits binary data) cannot be monitored at one time. In such a case, a floating point operation is recommended.

## Operation error

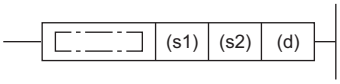
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |

# Dividing 32-bit binary data

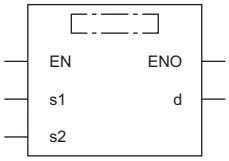
D/(P)(_U) instruction and DDIV(P)(_U) instruction can be used for division of 32-bit binary data.

## D/(P)(_U)

These instructions divide the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DDIVISION", "DDIVISIONP", "DDIVISION_U", "DDIVISIONP_U" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

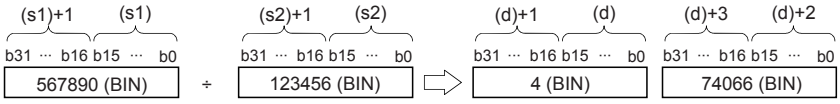
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | D/(P) | Dividend data or the head device where the data to be divided by another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D/(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | D/(P) | Divisor data or the head device where the data by which another is to be divided is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | D/(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | D/(P) | Head device for storing the operation result | — | 64-bit signed binary | ANY32_ARRAY (Number of elements: 2) |
| | D/(P)_U | | | 64-bit unsigned binary | |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions divide the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the division result in the device specified by (d).



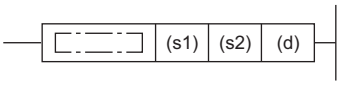

- For the division result of word device, 64-bit binary is used to store the quotient and remainder. For bit device, 32-bit binary is used to store quotient only.

## Operation error

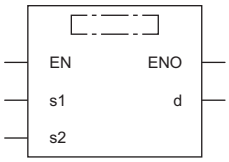
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |
| 3400H | 0 is specified for (s2) value. |
| 3403H | Signed operation is performed and the operation result exceeds 2147483647. |

## DDIV(P)(_U)

These instructions divide the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DDIV(EN,s1,s2,d);<br>ENO:=DDIVP(EN,s1,s2,d); | ENO:=DDIV_U(EN,s1,s2,d);<br>ENO:=DDIVP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

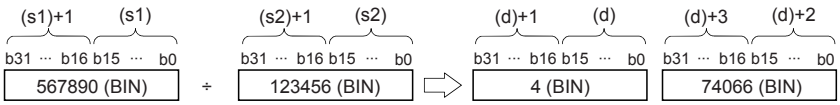
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DDIV(P) | Dividend data or the head device where the data to be divided by another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DDIV(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DDIV(P) | Divisor data or the head device where the data by which another is to be divided is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DDIV(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DDIV(P) | Head device for storing the operation result | — | 64-bit signed binary | ANY32_S_ARRAY (Number of elements: 2) |
| | DDIV(P)_U | | | 64-bit unsigned binary | ANY32_U_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions divide the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the division result in the device specified by (d).




#### ■Related flag

| Device | Name | Description |
|---|---|---|
| SM700 | Carry | When the operation result of the signed operation exceeds 32767, the carry flag is turned ON. |
| SM8304 | Zero | When the operation result is 0, the zero flag is turned ON. |
| SM8306 | Carry | When the operation result of the signed operation exceeds 32767, the carry flag is turned ON. |

## Precautions

### ■Operation result

- The most significant bit of the quotient and remainder indicates the sign (positive: 0, negative: 1), respectively.
- The quotient is negative when either (s1) or (s2) is negative. The remainder is negative when the (s1) is negative.

### ■Device specified by (d)

- The remainder is not obtained when a bit device is specified with nibble specification.

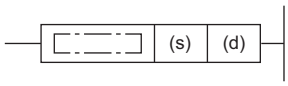
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of the device specified by (d) exceeds the range of said device. |
| 3400H | 0 is specified for (s2) value. |
| 3403H | Signed operation is performed and the operation result exceeds 2147483647. |

# Adding BCD 4-digit data

## B+(P) [using two operands]

These instructions add the BCD 4-digit data in the device specified by (d) and the BCD 4-digit data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

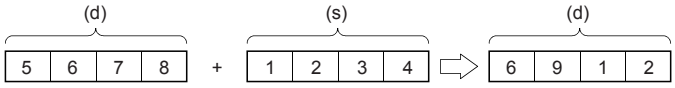
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Addend data or the device where the data that is added to another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Device where the data to which another is added is stored | 0 to 9999 | BCD 4-digit | ANY16 |

#### ■Applicable devices

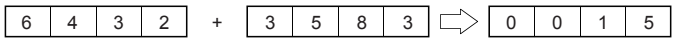
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions add the BCD 4-digit data in the device specified by (d) and the BCD 4-digit data in the device specified by (s), and store the addition result in the device specified by (d).




- If the addition result exceeds 9999, carry is ignored. In this case, the carry flag (SM700) does not turn ON.

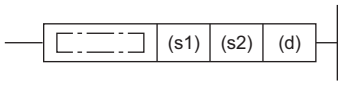



### Operation error

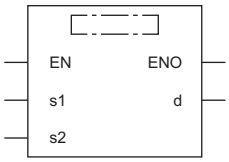
| Error code (SD/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (d) is outside of the valid range (0 to 9999). |

## B+(P) [using three operands]

These instructions add the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BPLUS(EN,s1,s2,d);<br>ENO:=BPLUSP(EN,s1,s2,d); |

**FBD/LD**




("BPLUS", "BPLUSP" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

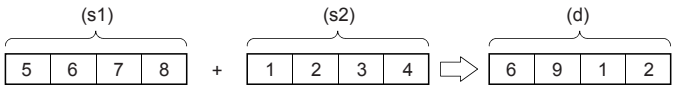
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Augend data or the device where the data to which another is added is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (s2) | Addend data or the device where the data that is added to another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Device for storing the operation result | 0 to 9999 | BCD 4-digit | ANY16 |

#### ■Applicable devices

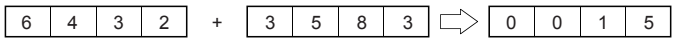
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions add the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the addition result in the device specified by (d).




- If the addition result exceeds 9999, carry is ignored. In this case, the carry flag (SM700) does not turn ON.



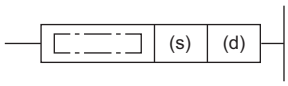

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 9999). |

# Subtracting BCD 4-digit data

## B-(P) [using two operands]

These instructions subtract the BCD 4-digit data in the device specified by (d) and the BCD 4-digit data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

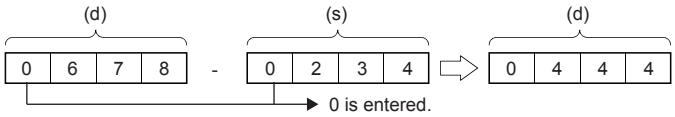
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Subtrahend data or the device where the data to be subtracted from another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Device where the data from which another is to be subtracted is stored | 0 to 9999 | BCD 4-digit | ANY16 |

#### ■Applicable devices

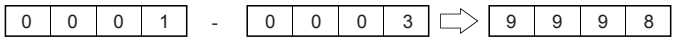
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s) and the BCD 4-digit data in the device specified by (d), and store the subtraction result in the device specified by (d).




- If an underflow occurs, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

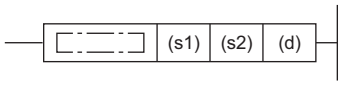



### Operation error

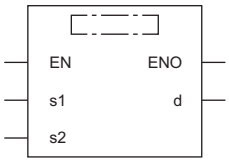
| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (d) is outside of the valid range (0 to 9999). |

## B-(P) [using three operands]

These instructions subtract the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BMINUS(EN,s1,s2,d);<br>ENO:=BMINUSP(EN,s1,s2,d); |

**FBD/LD**




("BMINUS", "BMINUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

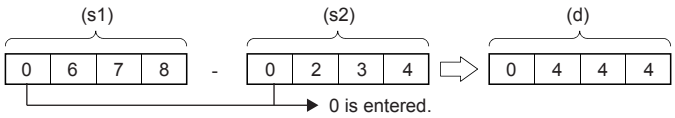
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Minuend data or the device where the data from which another is to be subtracted is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (s2) | Subtrahend data or the device where the data to be subtracted from another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Device for storing the operation result | 0 to 9999 | BCD 4-digit | ANY16 |

#### ■Applicable devices

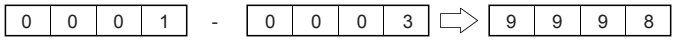
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the subtraction result in the device specified by (d).




- If an underflow occurs, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.



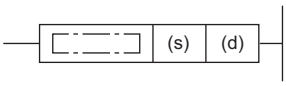

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 9999). |

# Adding BCD 8-digit data

## DB+(P) [using two operands]

These instructions add the BCD 8-digit data in the device specified by (d) and the BCD 8-digit data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

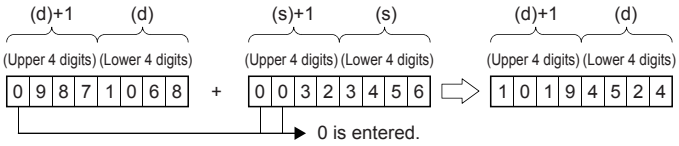
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Addend data or the head device where the data that is added to another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device where the data to which another is added is stored | 0 to 99999999 | BCD 8-digit | ANY32 |

#### ■Applicable devices

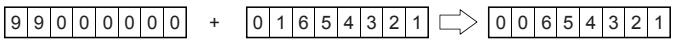
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions add the BCD 8-digit data in the device specified by (d) and the BCD 8-digit data in the device specified by (s), and store the addition result in the device specified by (d).




- If the addition result exceeds 99999999, carry is ignored. In this case, the carry flag (SM700) does not turn ON.

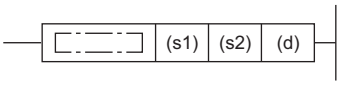



### Operation error

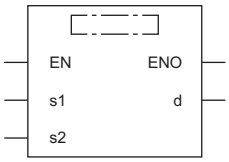
| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (d) is outside of the valid range (0 to 99999999). |

## DB+(P) [using three operands]

These instructions add the BCD 8-digit data in the device specified by (s1) and the BCD 8-digit data in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DBPLUS(EN,s1,s2,d);<br>ENO:=DBPLUSP(EN,s1,s2,d); |

**FBD/LD**




("DBPLUS", "DBPLUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

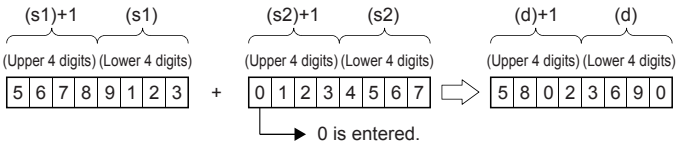
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Augend data or the head device where the data to which another is added is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (s2) | Addend data or the head device where the data that is added to another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device for storing the operation result | 0 to 99999999 | BCD 8-digit | ANY32 |

#### ■Applicable devices

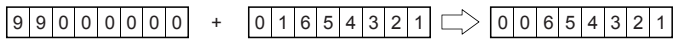
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions add the BCD 8-digit data in the device specified by (s1) and the BCD 8-digit data in the device specified by (s2), and store the addition result in the device specified by (d).




- If the addition result exceeds 99999999, carry is ignored. In this case, the carry flag (SM700) does not turn ON.



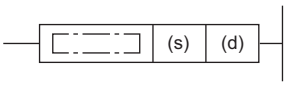

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 99999999). |

# Subtracting BCD 8-digit data

## DB-(P) [using two operands]

These instructions subtract the BCD 8-digit data in the device specified by (d) and the BCD 8-digit data in the device specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

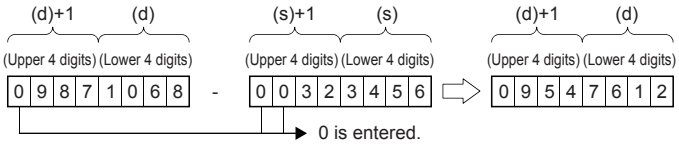
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Subtrahend data or the device where the data to be subtracted from another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Minuend data or the device where the data from which another is to be subtracted is stored | 0 to 99999999 | BCD 8-digit | ANY32 |

#### ■Applicable devices

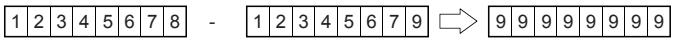
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions subtract the BCD 8-digit data specified by (d) and the BCD 8-digit data specified by (s), and store the results in the device specified by (d).




- If an underflow occurs, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

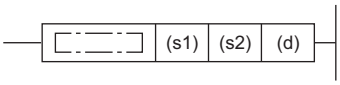



### Operation error

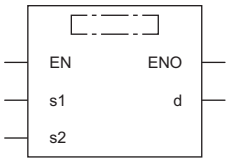
| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (d) is outside of the valid range (0 to 99999999). |

## DB-(P) [using three operands]

These instructions subtract the BCD 8-digit data specified by (s1) and the BCD 8-digit data specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DBMINUS(EN,s1,s2,d);<br>ENO:=DBMINUSP(EN,s1,s2,d); |

**FBD/LD**




("DBMINUS", "DBMINUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

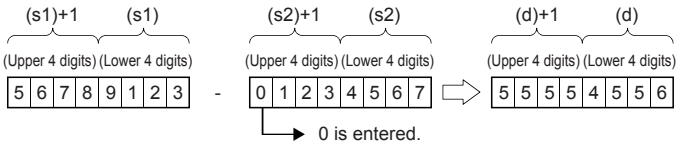
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Minuend data or the head device where the data from which another is to be subtracted is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (s2) | Subtrahend data or the head device where the data to be subtracted from another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device for storing the operation result | 0 to 99999999 | BCD 8-digit | ANY32 |

#### ■Applicable devices

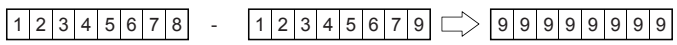
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions subtract the BCD 8-digit data specified by (s1) and the BCD 8-digit data specified by (s2), and store the results in the device specified by (d).




- If an underflow occurs, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.



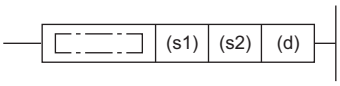

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 99999999). |

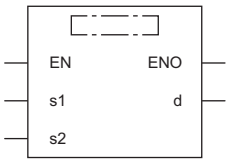
# Multiplying BCD 4-digit data

## B*(P)

These instructions multiply the BCD 4-digit data specified by (s1) and the BCD 4-digit data specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("BMULTI", "BMULTIP" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

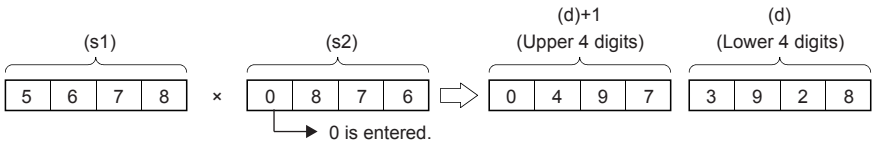
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Multiplicand data or the device where the data to be multiplied by another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (s2) | Multiplier data or the device where the data by which another is to be multiplied is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Head device for storing the operation result | — | BCD 8-digit | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions multiply the BCD 4-digit data specified by (s1) and the BCD 4-digit data specified by (s2), and store the multiplication results in the device specified by (d).



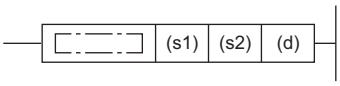

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Device specified by (d) exceeds the allowable device range |
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 9999). |

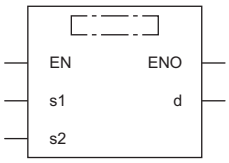
# Dividing BCD 4-digit data

## B/(P)

These instructions divide the BCD 4-digit data specified by (s1) by the BCD 4-digit data specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("BDIVISION", "BDIVISIONP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

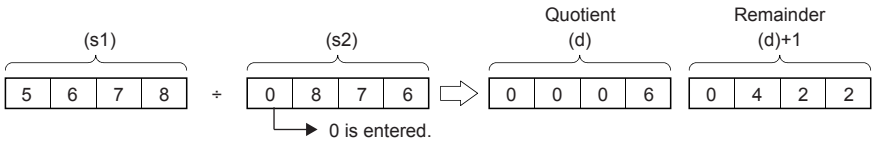
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Dividend data or the device where the data to be divided by another is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (s2) | Divisor data or the device where the data by which another is to be divided is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Head device for storing the operation result | — | BCD 8-digit | ANY16_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions divide the BCD 4-digit data specified by (s1) by the BCD 4-digit data specified by (s2), and store the results of division in the device specified by (d).




- The results of division are stored as quotient and remainder using 32 bit(s).
  - Quotient (BCD 4-digit): Stored in lower 16 bit(s).
  - Remainder (BCD 4-digit): Stored in upper 16 bit(s).
- If (d) is specified by bit device, remainder of division results is not stored.

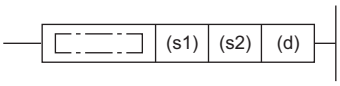
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Device specified by (d) exceeds the allowable device range |
| 3400H | 0 is specified for (s2) value. |
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 9999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 9999). |

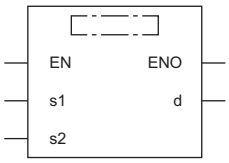
# Multiplying BCD 8-digit data

## DB*(P)

These instructions multiply the BCD 8-digit data specified by (s1) and the BCD 8-digit data specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DBMULTI", "DBMULTIP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

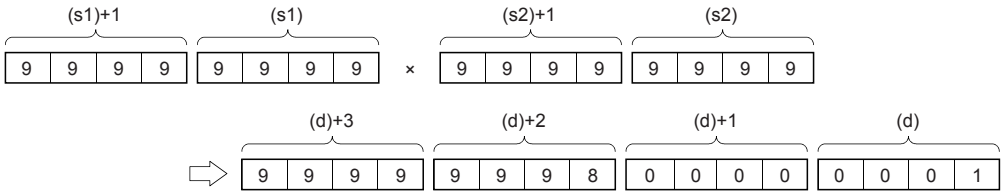
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Multiplicand data or the head device where the data to be multiplied by another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (s2) | Multiplier data or the head device where the data by which another is to be multiplied is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device for storing the operation result | — | BCD 16-digit | ANY32_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions multiply the BCD 8-digit data specified by (s1) and the BCD 8-digit data specified by (s2), and store the multiplication results in the device specified by (d).




- When (d) is a bit device, only the lower 8 nibbles (32 bits) of the multiplication result are stored, and the higher 8 nibbles (32 bits) cannot be specified.

Ex.

Multiplication result when (d) is a bit device

- K1 ... Lower 1 nibble (b0 to b3)
- K4 ... Lower 4 nibbles (b0 to b15)
- K8 ... Lower 8 nibbles (b0 to b31)

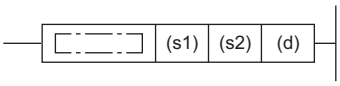
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Device specified by (d) exceeds the allowable device range |
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 99999999). |

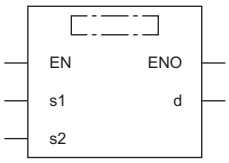
# Dividing BCD 8-digit data

## DB/(P)

These instructions divide the BCD 8-digit data specified by (s1) by the BCD 8-digit data specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DBDIVISION", "DBDIVISIONP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

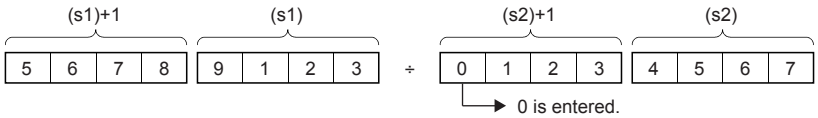
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Dividend data or the head device where the data to be divided by another is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (s2) | Divisor data or the head device where the data by which another is to be divided is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device for storing the operation result | — | BCD 16-digit | ANY32_ARRAY (Number of elements: 2) |

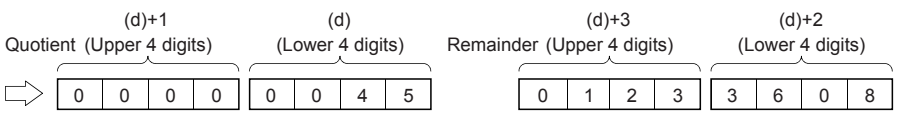
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions divide the BCD 8-digit data specified by (s1) by the BCD 8-digit data specified by (s2), and store the results of division in the device specified by (d).




- The results of division are stored as quotient and remainder using 64 bit(s) binary.
  - Quotient (BCD 8-digit): Stored in lower 32 bit(s).
  - Remainder (BCD 8-digit): Stored in upper 32 bit(s).
- If (d) is specified by bit device, remainder of division results is not stored.

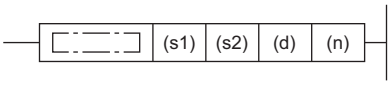
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Device specified by (d) exceeds the allowable device range |
| 3400H | 0 is specified for (s2) value. |
| 3405H | BCD data in the device specified by (s1) is outside of the valid range (0 to 99999999). |
| | BCD data in the device specified by (s2) is outside of the valid range (0 to 99999999). |

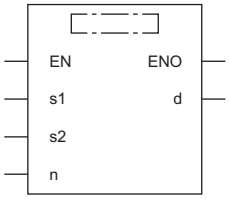
# Adding 16-bit binary block data

## BK+(P)(_U)

These instructions add (n) point(s) of 16-bit binary data from the device specified by (s1) and the (n) point(s) of 16-bit binary data from the device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("BKPLUS", "BKPLUSP", "BKPLUS_U", "BKPLUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | BK+(P) | Head device where the data to which another data is added is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BK+(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | BK+(P) | Addend data or the head device where the data that is added to another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BK+(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | BK+(P) | Head device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | BK+(P)_U | | — | 16-bit unsigned binary | ANY16_U |
| (n) | | Number of addition data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

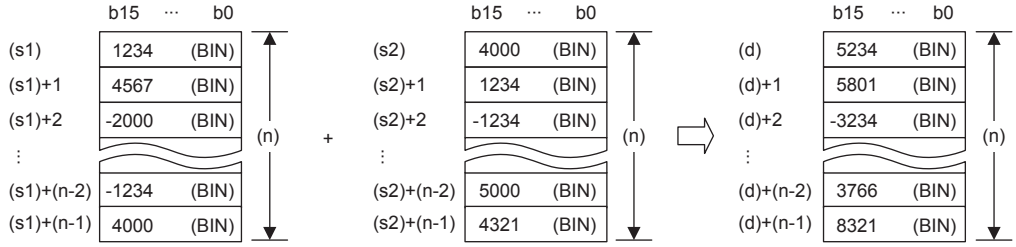
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

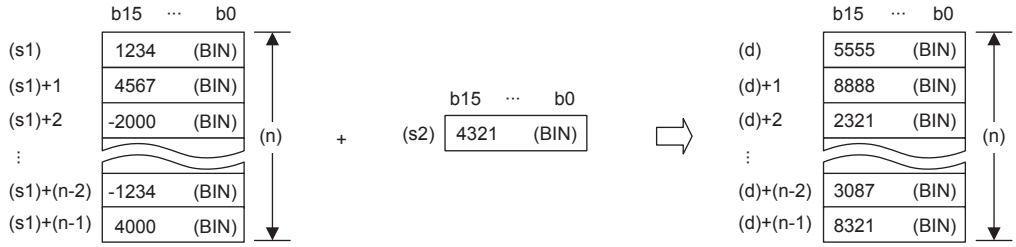
- These instructions add (n) point(s) of 16-bit binary data from the device specified by (s1) and the (n) point(s) of 16-bit binary data from the device specified by (s2), and store the results of addition in the device specified by (d).
- Block addition is performed in units of 16-bits.

**Ex.**

If device is specified for (s2) (signed)




If constant is specified for (s2) (signed)




- If an underflow or overflow occurs for operation result, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

| If signed is specified | If unsigned is specified |
|---|---|
| K32767 (7FFFH) + K2 (0002H) → K-32767 (8001H)<br>K-32767 (8001H) + K-2 (FFFEH) → K32767 (7FFFH) | K65535 (FFFFH) + K1 (0001H) → K0 (0000H) |

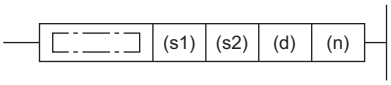
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | The device range for (n) point(s) beginning from (s1) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | The device range for (n) point(s) beginning from (s2) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

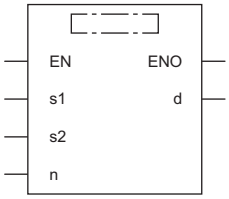
# Subtracting 16-bit binary block data

## BK-(P)(_U)

These instructions subtract (n) point(s) of 16-bit binary data from the device specified by (s1) and the (n) point(s) of 16-bit binary data from the device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("BKMINUS", "BKMINUSP", "BKMINUS_U", "BKMINUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | BK-(P) | Head device where the data from which another is to be subtracted is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BK-(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | BK-(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BK-(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | BK-(P) | Head device for storing the operation result | — | 16-bit signed binary | ANY16_S |
| | BK-(P)_U | | — | 16-bit unsigned binary | ANY16_U |
| (n) | | Number of subtraction data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

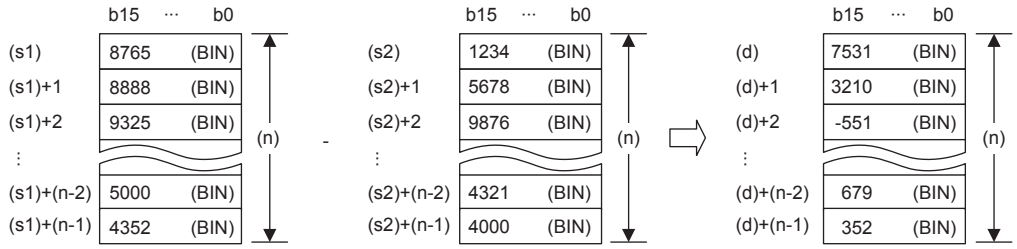
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

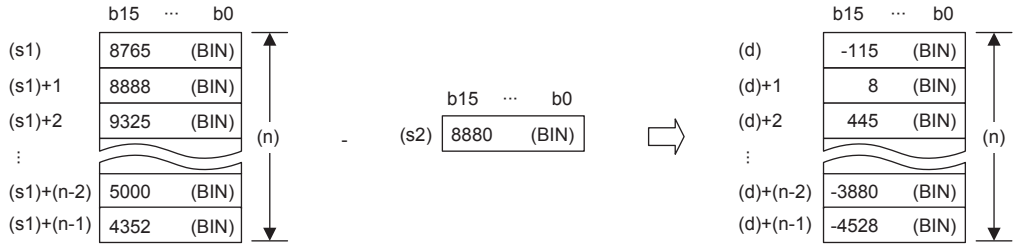
- These instructions subtract (n) point(s) of 16-bit binary data from the device specified by (s1) and the (n) point(s) of 16-bit binary data from the device specified by (s2), and store the subtraction results in the device specified by (d).
- Block subtraction is performed in 16-bit units.

**Ex.**

If device has been specified for (s2)




If constant is specified for (s2)




- If an underflow or overflow occurs for operation result, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

| If signed is specified | If unsigned is specified |
|---|---|
| K-32767 (8001H) - K2 (0002H) → K32766 (7FFEH)<br>K32767 (7FFFH) - K-2 (FFFEH) → K-32767 (8001H) | K0 (0000H) - K1 (0001H) → K65535 (FFFFH) |

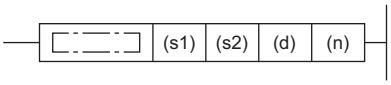
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | The device range for (n) point(s) beginning from (s1) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | The device range for (n) point(s) beginning from (s2) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

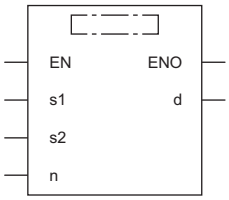
# Adding 32-bit binary block data

## DBK+(P)(_U)

These instructions add (n) point(s) of 32-bit binary data from the device specified by (s1) and the (n) point(s) of 32-bit binary data from the device specified by (s2), and store the results of addition in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DBKPLUS", "DBKPLUSP", "DBKPLUS_U", "DBKPLUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DBK+(P) | Head device where the data to which another is added is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBK+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DBK+(P) | Addend data or the head device where the data that is added to another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBK+(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DBK+(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | DBK+(P)_U | | — | 32-bit unsigned binary | ANY32_U |
| (n) | | Number of addition data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

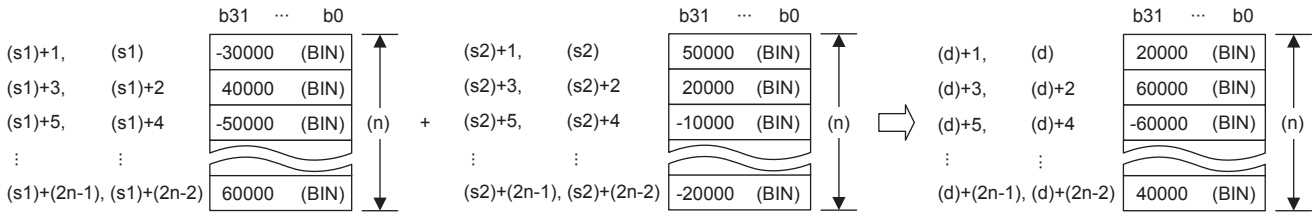
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | ○ | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

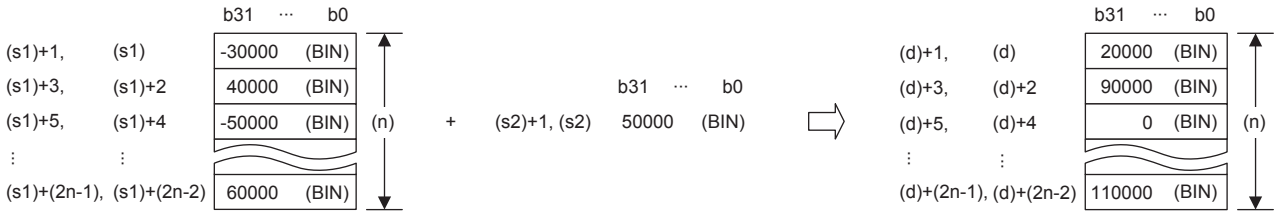
- These instructions add (n) point(s) of 32-bit binary data from the device specified by (s1) and the (n) point(s) of 32-bit binary data from the device specified by (s2), and store the results of addition in the device specified by (d).
- Block addition is performed in 32-bit units.

**Ex.**

If device is specified for (s2) (signed)



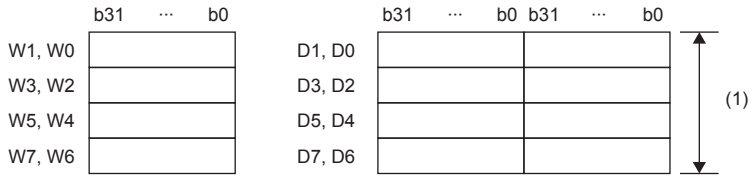

If constant is specified for (s2) (signed)




- Operation is enabled when (s1) or (s2) have been specified by same device as (d) (perfect match). An error occurs if the device range of (n) point(s) from (s1) or (s2) partially matches (overlaps) the device range of (n) point(s) from (d).

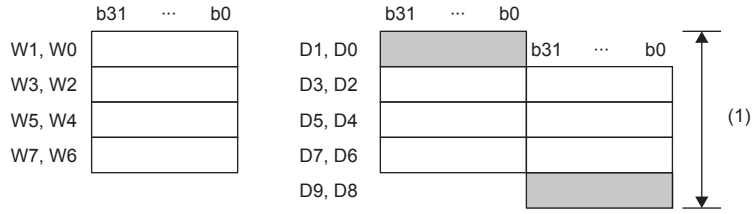
**Ex.**

If 4 points of the device from (s2) and (d) match




(1) Because it is a perfect match, operation is possible.

If 4 points of the device from (s2), (d) match partially




(1) An operation error occurs if they partially match.

- If the value specified for (n) is 0, processing is not performed.
- If an underflow or overflow occurs for operation result, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

| If signed is specified | | | | | If unsigned is specified | | | |
|---|---|---|---|---|---|---|---|---|
| K2147483647 (7FFFFFFFH) | + | K2 (00000002H) | ⇨ | K-2147483647 (80000001H) | K4294967295 (FFFFFFFFH) | + | K1 (00000001H) | ⇨ | K0 (00000000H) |
| K-2147483647 (80000001H) | + | K-2 (FFFFFFFEH) | ⇨ | K2147483647 (7FFFFFFFH) | | | | | |

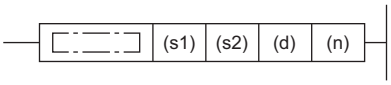
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | The device range for (n) point(s) beginning from (s1) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | The device range for (n) point(s) beginning from (s2) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

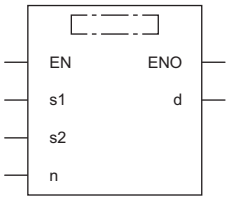
# Subtracting 32-bit binary block data

## DBK-(P)(_U)

These instructions subtract (n) point(s) of 32-bit binary data from the device specified by (s1) and the (n) point(s) of 32-bit binary data from the device specified by (s2), and store the results of subtraction in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("DBKMINUS", "DBKMINUSP", "DBKMINUS_U", "DBKMINUSP_U" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DBK-(P) | Head device where the data from which another is to be subtracted is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBK-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DBK-(P) | Subtrahend data or the head device where the data to be subtracted from another is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBK-(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DBK-(P) | Head device for storing the operation result | — | 32-bit signed binary | ANY32_S |
| | DBK-(P)_U | | — | 32-bit unsigned binary | ANY32_U |
| (n) | | Number of subtraction data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

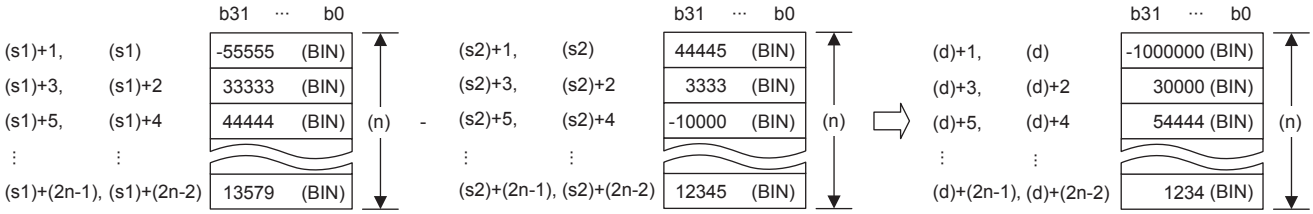
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | ○ | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

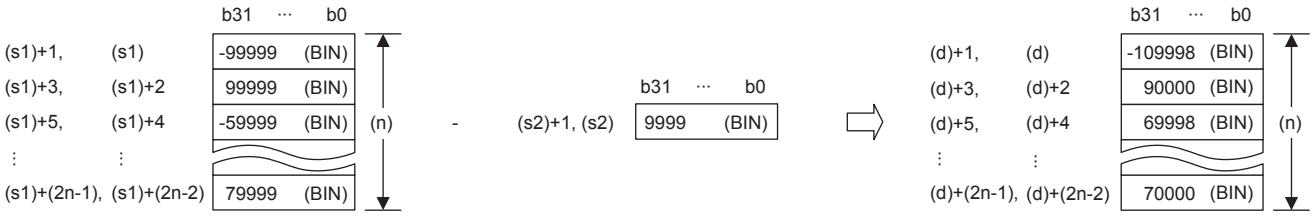
- These instructions subtract (n) point(s) of 32-bit binary data from the device specified by (s1) and the (n) point(s) of 32-bit binary data from the device specified by (s2), and store the results of subtraction in the device specified by (d).
- Block subtraction is performed in 32-bit units.

**Ex.**

If device is specified for (s2) (signed)



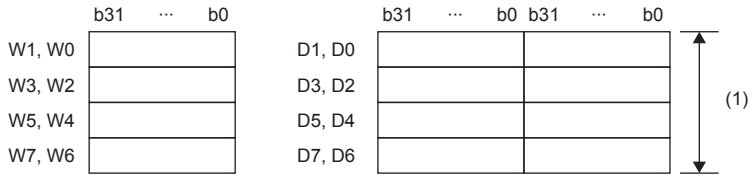

If constant is specified for (s2) (signed)




- Operation is enabled when (s1) or (s2) have been specified by same device as (d) (perfect match). An error occurs if the device range of (n) point(s) from (s1) or (s2) partially matches (overlaps) the device range of (n) point(s) from (d).

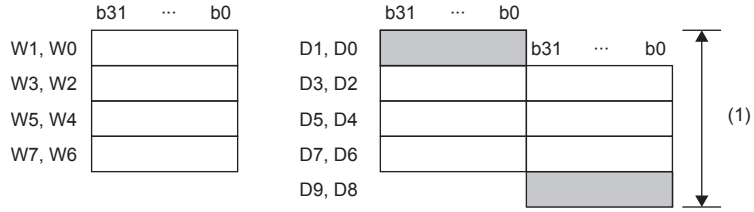
**Ex.**

If 4 points of the device from (s2) and (d) match




(1) Because it is a perfect match, operation is possible.

If 4 points of the device from (s2), (d) match partially




(1) An operation error occurs if they partially match.

- If the value specified for (n) is 0, processing is not performed.
- If an underflow or overflow occurs for operation result, the result will be as follows. In this case, the carry flag (SM700) does not turn ON.

| If signed is specified | If unsigned is specified |
|---|---|
| K2147483647 (7FFFFFFFH) - K-2 (FFFFFFFEH) ⇨ K-2147483647 (80000001H) | K0 (00000000H) - K1 (00000001H) ⇨ K4294967295 (FFFFFFFFH) |
| K-2147483647 (80000001H) - K2 (00000002H) ⇨ K2147483647 (7FFFFFFFH) | |

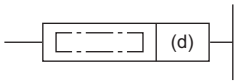
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | The device range for (n) point(s) beginning from (s1) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | The device range for (n) point(s) beginning from (s2) overlaps with that of (n) point(s) starting from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

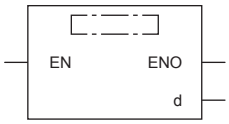
# Incrementing 16-bit binary data

## INC(P)(_U)

These instructions add +1 to the device (16-bit binary data) specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=INC(EN,d);<br>ENO:=INCP(EN,d); | ENO:=INC_U(EN,d);<br>ENO:=INCP_U(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

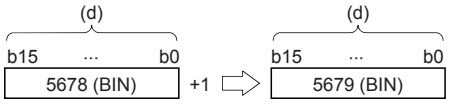
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (d) | INC(P) | Device to be incremented by +1 | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | INC(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions add +1 to the device (16-bit binary data) specified by (d).




- If INC(P) instruction is executed when contents of device specified by (d) is 32767, -32768 is stored in the device specified by (d). (If signed is specified)
- If INC(P)_U instruction is executed when contents of device specified by (d) is 65535, 0 is stored in the device specified by (d). (If unsigned is specified)
- Flags (zero, carry and borrow) are not activated at this time.

### Precautions

Note that data is incremented in every operation cycle in a continuous operation type (INC) instruction.

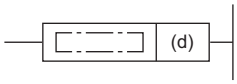
### Operation error

There is no operation error.

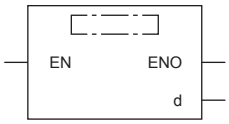
# Decrementing 16-bit binary data

## DEC(P)(_U)

These instructions subtract 1 from the device (16-bit binary data) specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DEC(EN,d);<br>ENO:=DECP(EN,d); | ENO:=DEC_U(EN,d);<br>ENO:=DECP_U(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

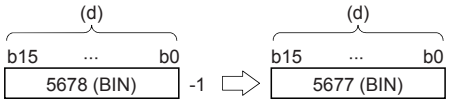
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (d) | DEC(P) | Device to be decremented by -1 | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | DEC(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions decrement device (16-bit binary data) specified by (d) by 1.




- If DEC(P) instruction is executed when contents of device specified by (d) is -32768, 32767 is stored in the device specified by (d). (If signed is specified)
- If DEC(P)_U instruction is executed when contents of device specified by (d) is 0, 65535 is stored in the device specified by (d). (If unsigned is specified)
- Flags (zero, carry and borrow) are not activated at this time.

### Precautions

Note that data is decremented in every operation cycle in a continuous operation type (DEC) instruction.

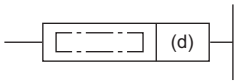
### Operation error

There is no operation error.

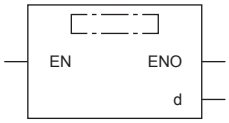
# Incrementing 32-bit binary data

## DINC(P)(_U)

These instructions add +1 to the device (32-bit binary data) specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DINC(EN,d);<br>ENO:=DINCP(EN,d); | ENO:=DINC_U(EN,d);<br>ENO:=DINCP_U(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

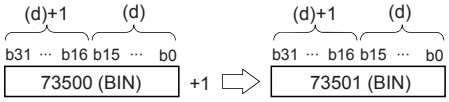
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (d) | DINC(P) | Head device to be incremented by +1 | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DINC(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions add +1 to the contents of device (32-bit binary data) specified by (d).




- If DINC(P) instruction is executed when contents of device specified by (d) is 2147483647, -2147483648 is stored in the device specified by (d). (If signed is specified)
- If DINC(P)_U instruction is executed when contents of device specified by (d) is 4294967295, 0 is stored in the device specified by (d). (If unsigned is specified)
- Flags (zero, carry and borrow) are not activated at this time.

### Precautions

Note that data is incremented in every operation cycle in a continuous operation type instruction.

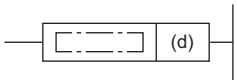
### Operation error

There is no operation error.

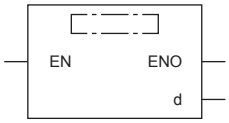
# Decrementing 32-bit binary data

## DDEC(P)(_U)

These instructions subtract 1 from the device (32-bit binary data) specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DDEC(EN,d);<br>ENO:=DDECP(EN,d); | ENO:=DDEC_U(EN,d);<br>ENO:=DDECP_U(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

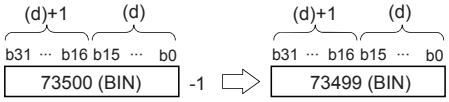
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (d) | DDEC(P) | Head device to be decremented by 1 | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DDEC(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions decrement contents of device (32-bit binary data) specified by (d) by 1.




- If DDEC(P) instruction is executed when contents of device specified by (d) is 0, -1 is stored in the device specified by (d). (If signed is specified)
- If DDEC(P)_U instruction is executed when contents of device specified by (d) is 0, 4294967295 is stored in the device specified by (d). (If unsigned is specified)
- Flags (zero, carry and borrow) are not activated at this time.

### Precautions

Note that data is decremented in every operation cycle in a continuous operation type (DDEC) instruction.

### Operation error

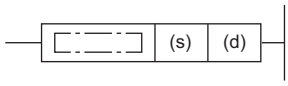
There is no operation error.

# 7.3 Logical Operation Instructions

## Performing an AND operation on 16-bit data

### WAND(P) [using two operands]

These instructions AND each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

#### Setting data

##### ■Descriptions, ranges, and data types

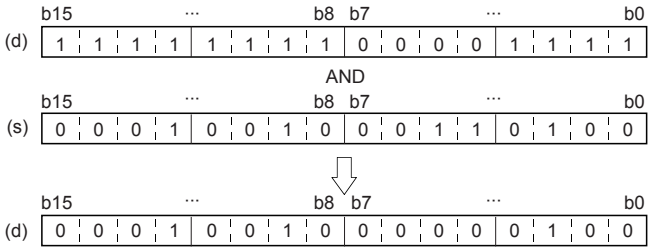
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for AND or device where the data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Device for storing AND results | -32768 to +32767 | 16-bit signed binary | ANY16 |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions AND each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).



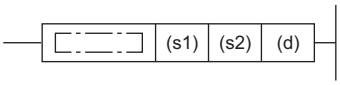

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

#### Operation error

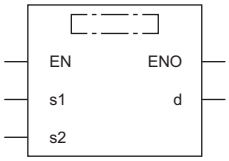
There is no operation error.

## WAND(P) [using three operands]

These instructions AND each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WAND(EN,s1,s2,d);<br>ENO:=WANDP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

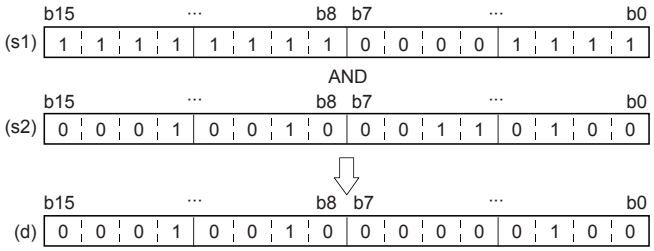
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for AND or device where the data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for AND or device where the data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Device for storing AND results | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions AND each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

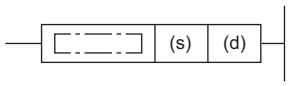
### Operation error

There is no operation error.

# Performing an AND operation on 32-bit data

## DAND(P) [using two operands]

These instructions AND each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

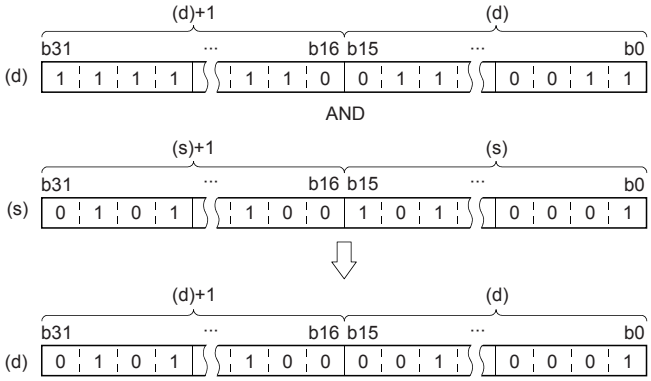
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for AND or head device where the data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing AND results | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions AND each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).



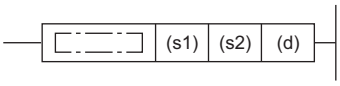

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

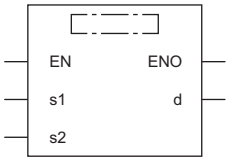
There is no operation error.

## DAND(P) [using three operands]

These instructions AND each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DAND(EN,s1,s2,d);<br>ENO:=DANDP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

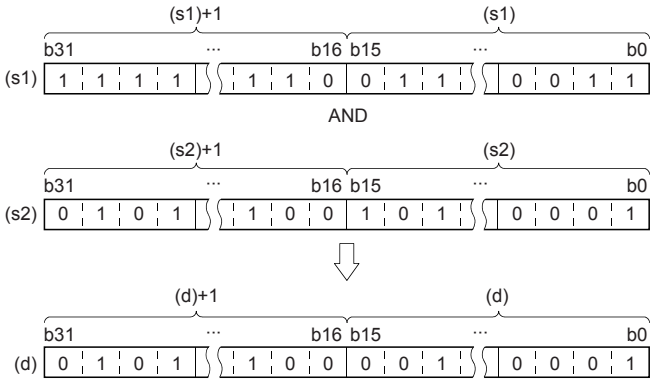
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for AND or head device where the data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Data for AND or head device where the data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing AND results | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions AND each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

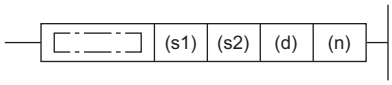
### Operation error

There is no operation error.

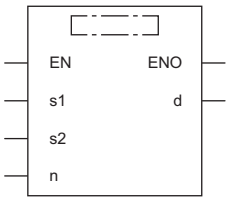
# Performing an AND operation on 16-bit block data

## BKAND(P)

These instructions AND contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BKAND(EN,s1,s2,n,d);<br>ENO:=BKANDP(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device that stores data for AND | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for AND or head device where the data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing AND results | — | 16-bit signed binary | ANY16 |
| (n) | Number of data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

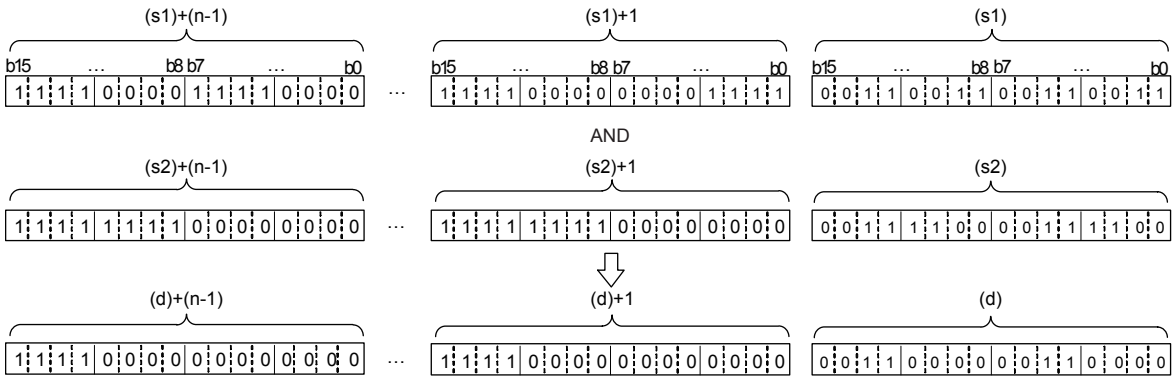
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2)*1 | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

### Processing details

- These instructions AND contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.



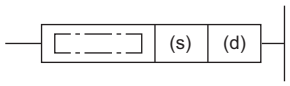
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | Device range of (n) point(s) from (s1) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | Device range of (n) point(s) from (s2) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

# Performing an OR operation on 16-bit data

## WOR(P) [using two operands]

These instructions OR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

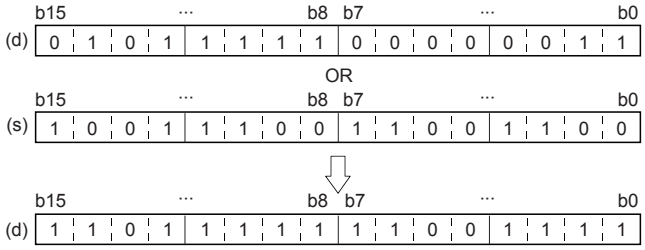
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the OR results | -32768 to +32767 | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions OR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).



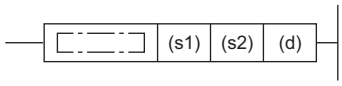

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

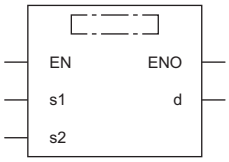
There is no operation error.

## WOR(P) [using three operands]

These instructions OR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WOR(EN,s1,s2,d);<br>ENO:=WORP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

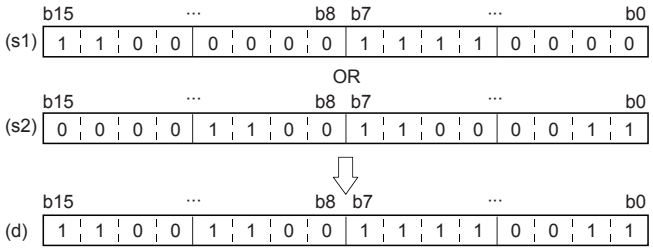
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the OR results | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions OR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

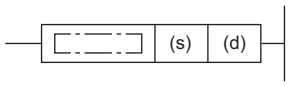
### Operation error

There is no operation error.

# Performing an OR operation on 32-bit data

## DOR(P) [using two operands]

These instructions OR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

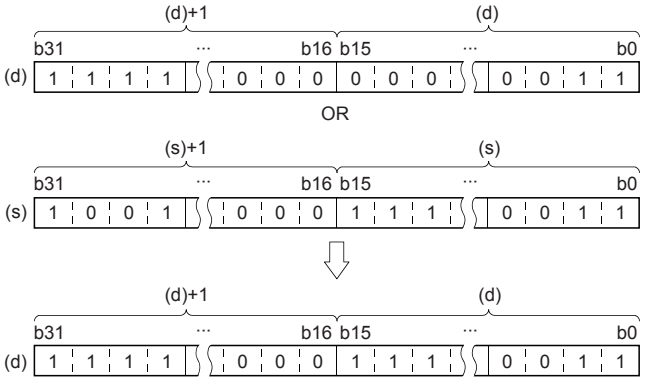
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing the OR results | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions OR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).



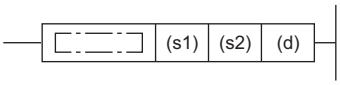

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

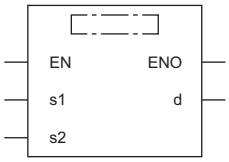
There is no operation error.

## DOR(P) [using three operands]

These instructions OR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DOR(EN,s1,s2,d);<br>ENO:=DORP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

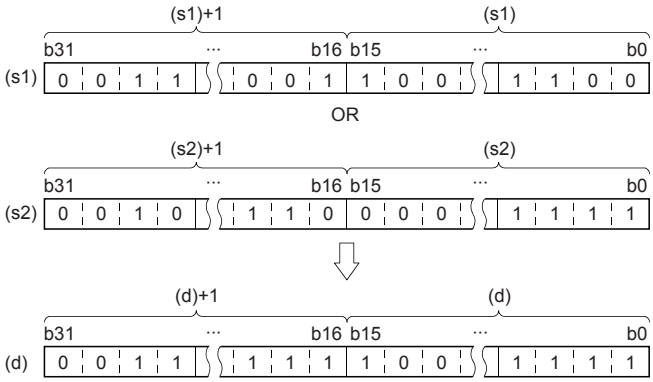
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Data for OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing the OR results | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions OR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

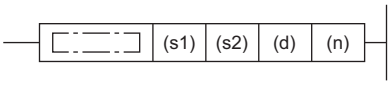
### Operation error

There is no operation error.

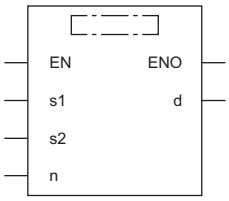
# Performing an OR operation on 16-bit block data

## BKOR(P)

These instructions OR contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BKOR(EN,s1,s2,n,d);<br>ENO:=BKORP(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device where the logical operation data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Logical operation data or the head device where the logical operation data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the operation result | — | 16-bit signed binary | ANY16 |
| (n) | Number of data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

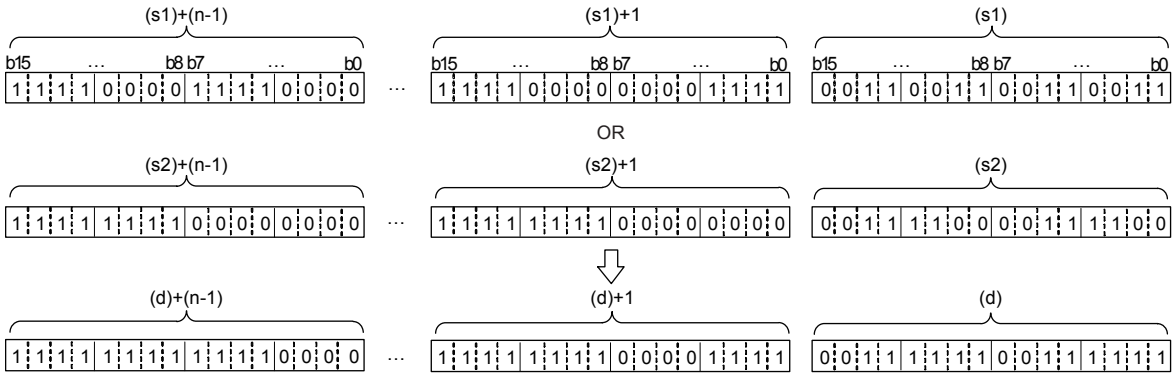
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2)*1 | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

### Processing details

- These instructions seek OR of contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.



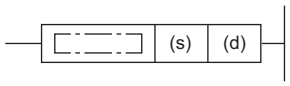
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | Device range of (n) point(s) from (s1) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | Device range of (n) point(s) from (s2) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

# Performing an XOR operation on 16-bit data

## WXOR(P) [using two operands]

These instructions exclusive OR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

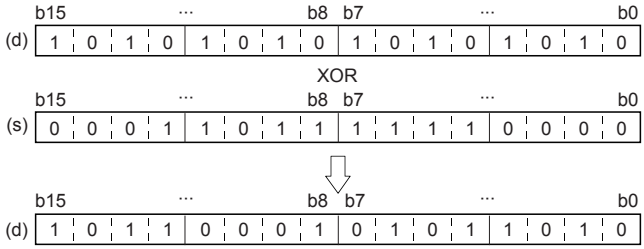
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for exclusive OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing exclusive OR results | -32768 to +32767 | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions exclusive OR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).



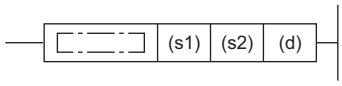

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

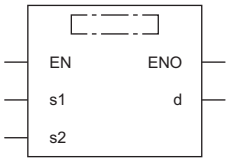
There is no operation error.

## WXOR(P) [using three operands]

These instructions exclusive OR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WXOR(EN,s1,s2,d);<br>ENO:=WXORP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

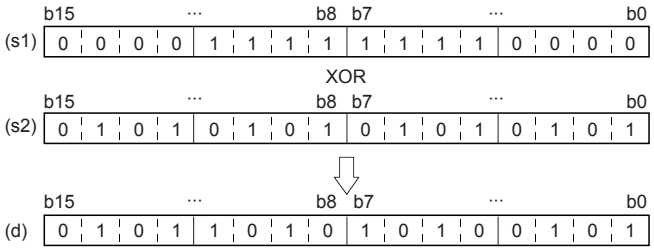
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for exclusive OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for exclusive OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing exclusive OR results | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions exclusive OR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

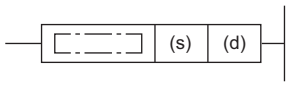
### Operation error

There is no operation error.

# Performing an XOR operation on 32-bit data

## DXOR(P) [using two operands]

These instructions exclusive OR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

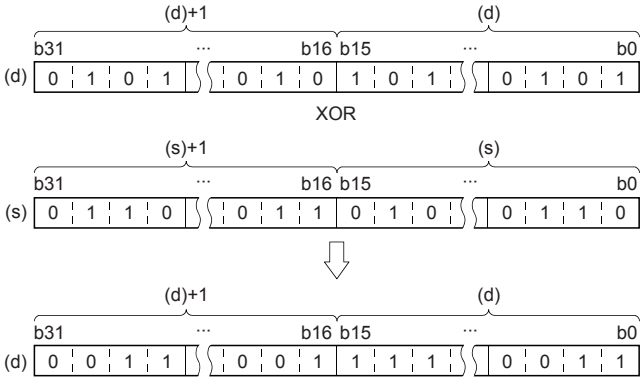
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for exclusive OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing exclusive OR results | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions exclusive OR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).



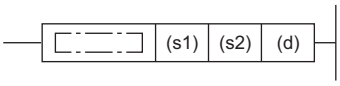

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

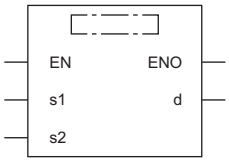
There is no operation error.

## DXOR(P) [using three operands]

These instructions exclusive OR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DXOR(EN,s1,s2,d);<br>ENO:=DXORP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

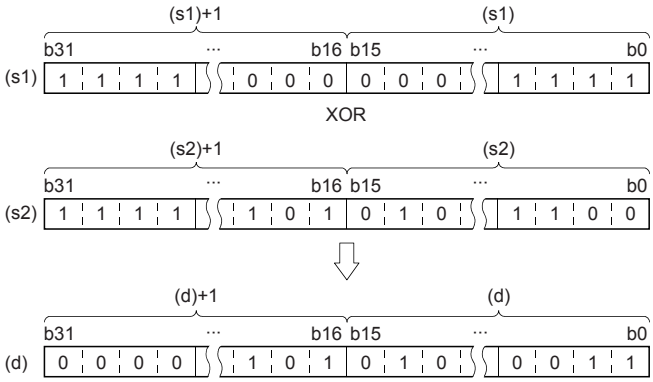
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for exclusive OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Data for exclusive OR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing exclusive OR results | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions exclusive OR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

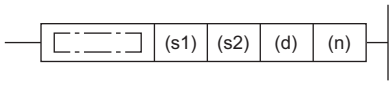
### Operation error

There is no operation error.

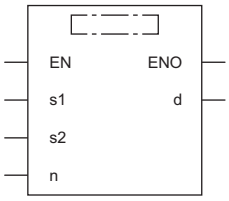
# Performing an XOR operation on 16-bit block data

## BKXOR(P)

These instructions seek exclusive OR of contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BKXOR(EN,s1,s2,n,d);<br>ENO:=BKXORP(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for exclusive OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for exclusive OR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the operation result | — | 16-bit signed binary | ANY16 |
| (n) | Number of data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

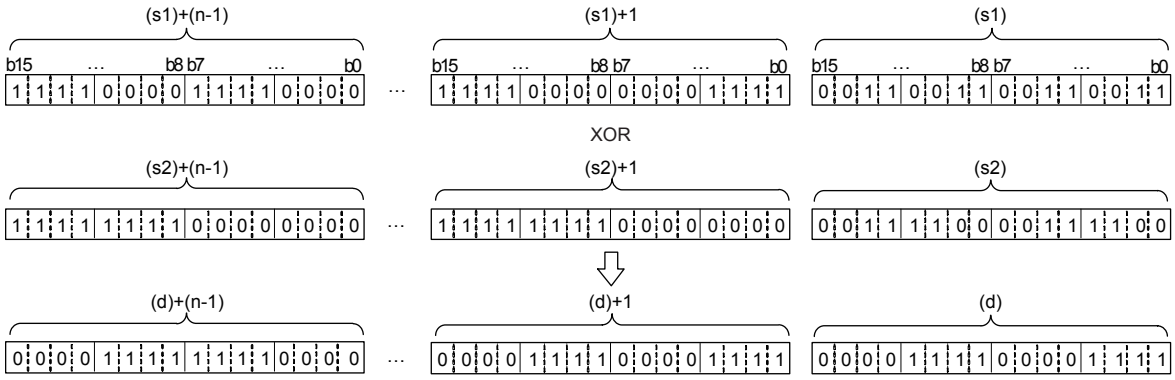
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2)*1 | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

### Processing details

- These instructions exclusive OR contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.



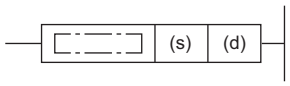
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | Device range of (n) point(s) from (s1) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | Device range of (n) point(s) from (s2) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

# Performing an XNOR operation on 16-bit data

## WXNR(P) [using two operands]

These instructions exclusive NOR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

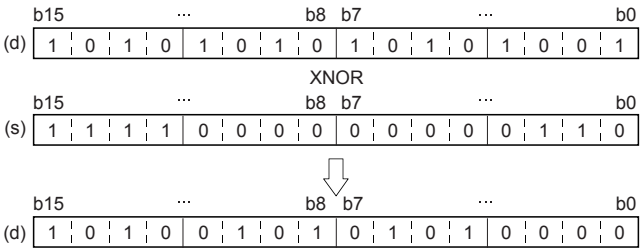
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for exclusive NOR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing exclusive NOR results | -32768 to +32767 | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions exclusive NOR each bit of 16-bit binary data from the device specified by (d) and each bit of 16-bit binary data from device specified by (s), and store the results in the device specified by (d).



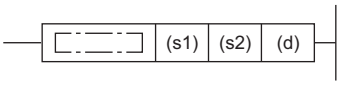

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

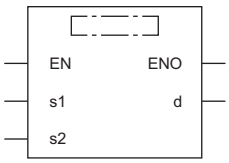
There is no operation error.

## WXNR(P) [using three operands]

These instructions exclusive NOR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WXNR(EN,s1,s2,d);<br>ENO:=WXNRP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for exclusive NOR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Data for exclusive NOR or head device where data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing exclusive NOR results | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions exclusive NOR each bit of 16-bit binary data from the device specified by (s1) and each bit of 16-bit binary data from device specified by (s2), and store the results in the device specified by (d).


- Bit devices subsequent to number of points by nibble specification are calculated as 0.

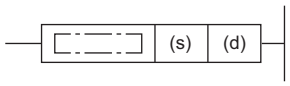
### Operation error

There is no operation error.

# Performing an XNOR operation on 32-bit data

## DXNR(P) [using two operands]

These instructions exclusive NOR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

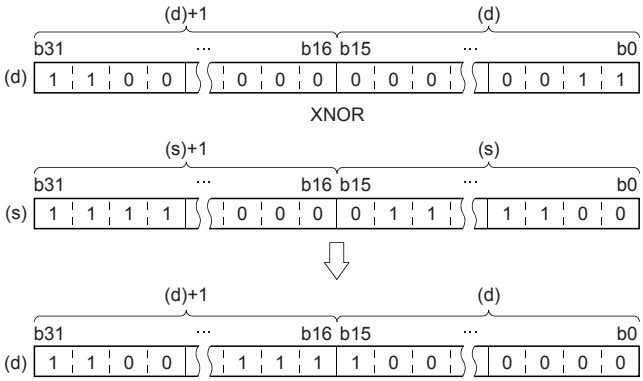
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data for exclusive NOR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing exclusive NOR results | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions exclusive NOR each bit of 32-bit binary data from the device specified by (d) and each bit of 32-bit binary data from device specified by (s), and store the results in the device specified by (d).



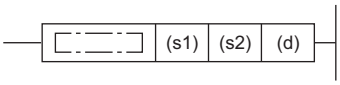

- Bit devices subsequent to number of points by nibble specification are calculated as 0.

### Operation error

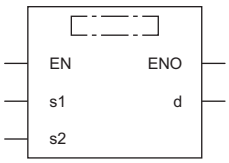
There is no operation error.

## DXNR(P) [using three operands]

These instructions exclusive NOR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DXNR(EN,s1,s2,d);<br>ENO:=DXNRP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

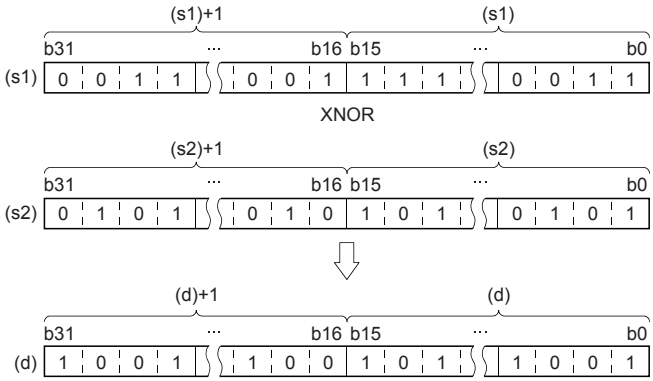
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data for exclusive NOR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Data for exclusive NOR or head device where data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing exclusive NOR results | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions exclusive NOR each bit of 32-bit binary data from the device specified by (s1) and each bit of 32-bit binary data from device specified by (s2), and store the results in the device specified by (d).




- Bit devices subsequent to number of points by nibble specification are calculated as 0.

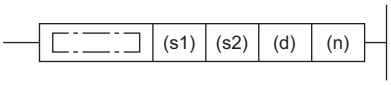
### Operation error

There is no operation error.

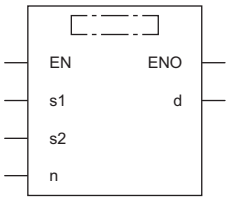
# Performing an XNOR operation on 16-bit block data

## BKXNR(P)

These instructions exclusive NOR contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BKXNR(EN,s1,s2,n,d);<br>ENO:=BKXNRP(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device where the logical operation data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Logical operation data or the head device where the logical operation data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the operation result | — | 16-bit signed binary | ANY16 |
| (n) | Number of data | 0 to 65535 | 16-bit unsigned binary | ANY16 |

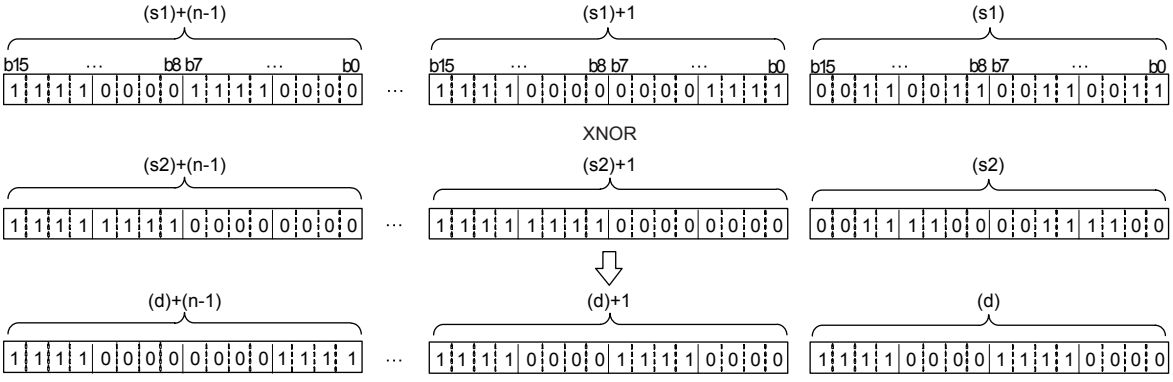
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2)*1 | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d)*1 | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

### Processing details

- These instructions exclusive NOR contents of (n) point(s) from the device specified by (s1) and (n) point(s) from the device specified by (s2), and store the results in the devices specified by (d) onward.



## Operation error

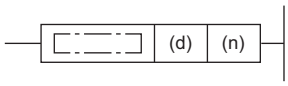
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s1), (s2), or (d) exceed the corresponding device range. |
| 2821H | Device range of (n) point(s) from (s1) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s1) and (d).) |
| | Device range of (n) point(s) from (s2) partially overlaps with device range of (n) point(s) from (d).<br>(Does not apply when same device has been specified for (s2) and (d).) |

# 7.4 Bit Processing Instructions

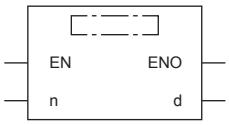
## Setting a bit in the word device

### BSET(P)

These instructions set (to 1) (n)th bit of word device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BSET(EN,n,d);<br>ENO:=BSETP(EN,n,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

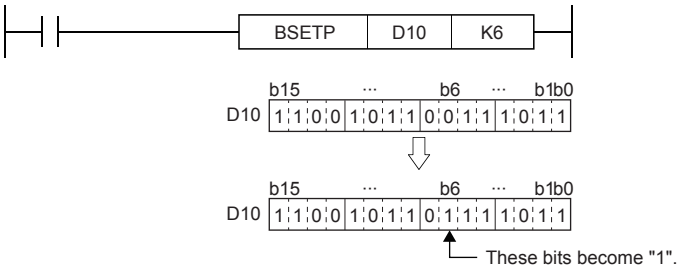
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for which bit is to be set | — | 16-bit signed binary | ANY16 |
| (n) | Number of bit(s) to be set | 0 to 15 | 16-bit unsigned binary | ANY16 |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

#### Processing details

- These instructions set (to 1) (n)th bit of word device specified by (d).
- If (n) exceeds 15, the processing will be done based on the lower 4 bits of (n).



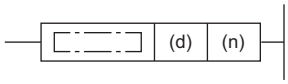

#### Operation error

There is no operation error.

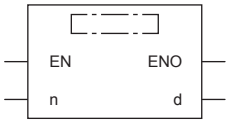
# Resetting a bit in the word device

## BRST(P)

These instructions reset (to 0) (n)th bit of word device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BRST(EN,n,d);<br>ENO:=BRSTP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

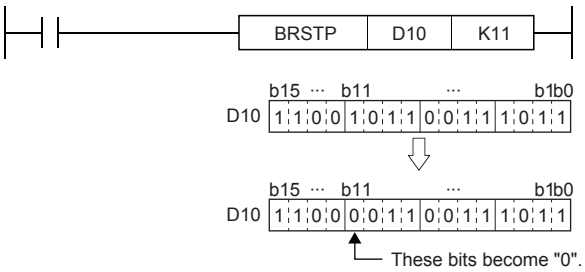
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for which bit is to be reset | — | 16-bit signed binary | ANY16 |
| (n) | Number of bit(s) to be reset | 0 to 15 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions reset (to 0) (n)th bit of word device specified by (d).
- If (n) exceeds 15, the processing will be done based on the lower 4 bits of (n).



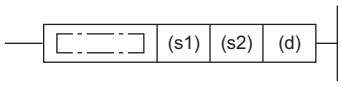

### Operation error

There is no operation error.

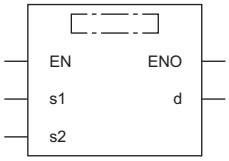
# Performing a 16-bit test

## TEST(P)

These instructions take bit data at position specified by (s2) from device specified by (s1) and write to bit device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TEST(EN,s1,s2,d);<br>ENO:=TESTP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Device number where bit data to be extracted is stored | — | 16-bit signed binary | ANY16 |
| (s2) | Position of bit data to be extracted | 0 to 15 | 16-bit unsigned binary | ANY16 |
| (d) | Bit device number where extracted bit data is to be stored | — | Bit | ANY_BOOL |

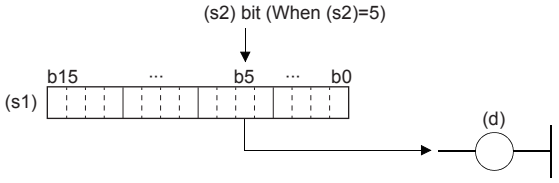
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | ○ | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions take bit data at position specified by (s2) from device specified by (s1) and write to bit device specified by (d).




- If relevant bit is "0", device specified by (d) is turned OFF, and if it is "1", device is turned ON.
- For (s2) specify the bit position (0 to 15) of word data. If 16 or more is specified for (s2), the value of the remainder of (s2)÷16 is the bit position.

Ex.

For (s2) = 18, the remainder for 18÷16 is "2", so it becomes data of b2.

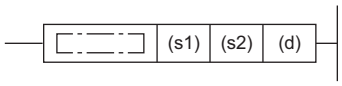
### Operation error

There is no operation error.

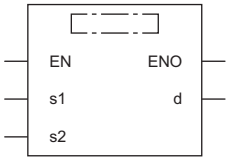
# Performing a 32-bit test

## DTEST(P)

These instructions take bit data at position specified by (s2) from device specified by (s1) and write to bit device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DTEST(EN,s1,s2,d);<br>ENO:=DTESTP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Device number where bit data to be extracted is stored | — | 32-bit signed binary | ANY32 |
| (s2) | Position of bit data to be extracted | 0 to 31 | 16-bit unsigned binary | ANY16 |
| (d) | Bit device number where extracted bit data is to be stored | — | Bit | ANY_BOOL |

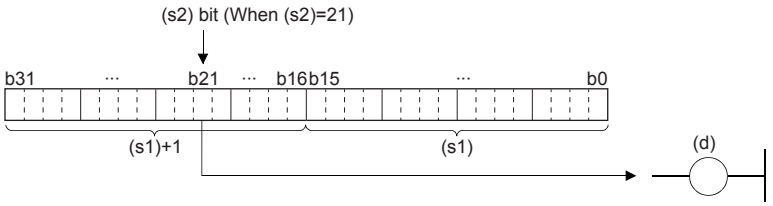
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | ○ | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions take bit data at position specified by (s2) from device specified by (s1), (s1) +1 and write to bit device specified by (d).




- If relevant bit is "0", device specified by (d) is turned OFF, and if it is "1", device is turned ON.
- For (s2) specify the bit position (0 to 31) of double word data. If 32 or more is specified for (s2), the value of the remainder of (s2)÷32 is the bit position.

Ex.

For (s2) = 34, the remainder for 34÷32 is "2", so it becomes data of b2.

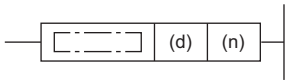
### Operation error

There is no operation error.

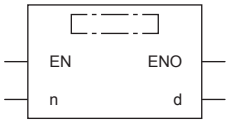
# Batch-resetting bit devices

## BKRST(P)

These instructions reset (n) point(s) bit devices from the bit device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BKRST(EN,n,d);<br>ENO:=BKRSTP(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device to be reset | — | Bit | ANY_BOOL |
| (n) | Number of devices to be reset | — | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | ○ | ○ | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions reset (n) point(s) bit devices from the bit device specified by (d).
- Reset status of bit device is as follows.

| Device | Status |
|---|---|
| Annunciator (F) | • (n) point(s) from annunciator (F) number specified by (d) are turned OFF.<br>• Annunciator numbers from SD64 to SD79 that were turned OFF are deleted and the subsequent numbers are shifted forward.<br>• The number of annunciators stored in SD64 to SD79 is stored in SD63. |
| Timer (T), Counter (C) | • Current value of (n) point(s) from timer (T) or counter (C) number specified by (d) is set to 0, and coil contact is turned OFF. |
| Bit devices other than given above | • Coils and contacts of (n) point(s) from the device specified by (d) are turned OFF. |

- If specified devices are OFF, device status remains unchanged.

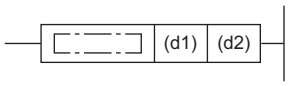
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | (n) point(s) of data starting from the device specified by (d) exceed the corresponding device range. |

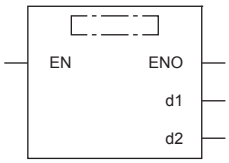
# Batch-resetting devices

## ZRST(P)

These instructions reset all data among devices of same type specified by (d1) and (d2). Use these instructions for restarting operation from the beginning after pause or after resetting control data.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ZRST(EN, d1, d2);<br>ENO:=ZRSTP(EN, d1, d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

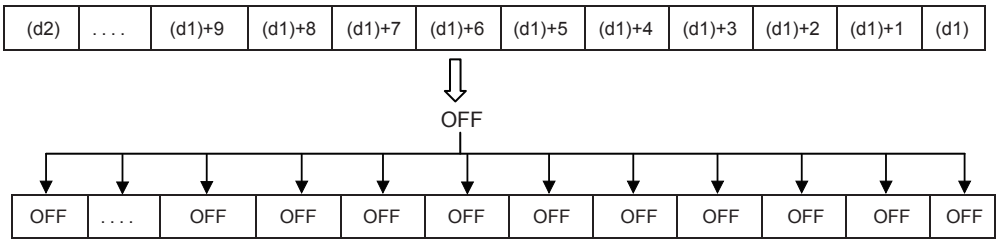
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d1) | Head bit or word device number to be reset | — | Bit/16-bit signed binary | ANY_ELEMENTARY |
| (d2) | Last bit or word device number to be reset | — | Bit/16-bit signed binary | ANY_ELEMENTARY |

#### ■Applicable devices

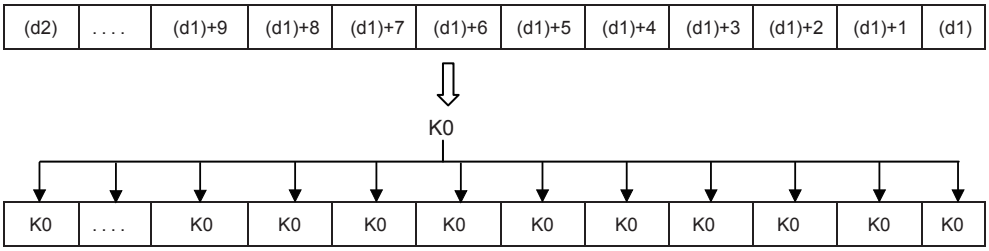
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

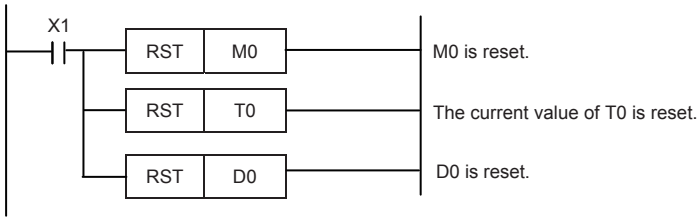
- These instructions reset all data among devices of same type specified by (d1) and (d2).
- OFF (reset) is written to the entire range of devices from (d1) to (d2) all at once if (d1) and/or (d2) are bit devices.



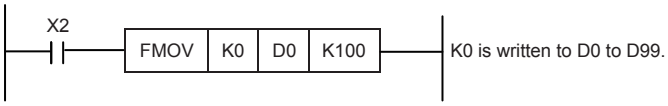

- K0 is written to the entire range of devices from (d1) to (d2) all at once if (d1) and/or (d2) are word devices.



- As a reset instruction for individual devices, the RST instruction can be used for bit devices and word devices.

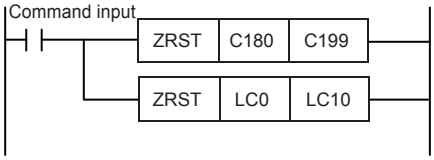



- The FMOV(P) instruction is a batch write instruction for a constant (K0 for example) that can write "0" for word devices (including nibble specification of bit devices).




## Precautions

- Specify the same type of device for (d1) and (d2) so that (d1) number is less than (d2) number. If the (d1) number≥(d2) number, only the device specified by (d1) is reset.
- The ZRST(P) instruction is a 16-bit instruction, but long counter (LC) and long index register (LZ) can be specified for (d1) and (d2).




## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of devices to be reset is 32768 or more when module access device has been specified for (d1) and/or (d2). |
| 3405H | Device type specified by (d1) differs from type specified by (d2). |
| | Module number for (d1) and (d2) differ when module access device is specified. |

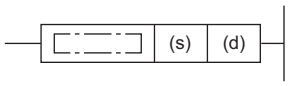
# 7.5 Data Conversion Instructions

## Converting binary data to BCD 4-digit data

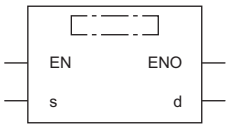
### BCD(P)

These instructions convert the binary data in the device specified by (s) to BCD data, and store the converted data in the device specified by (d).

Binary data is used in operations in CPU module. Use this instruction to display numeric values on seven-segment display unit equipped with BCD decoder.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BCD(EN,s,d);<br>ENO:=BCDP(EN,s,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

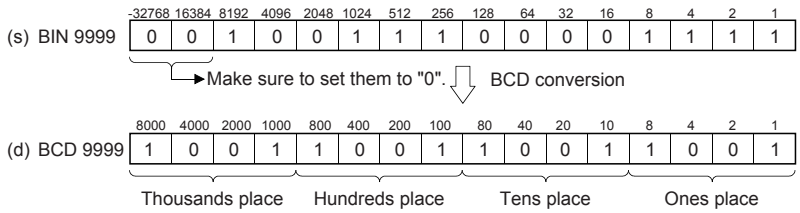
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Binary data or the head device where the binary data is stored | 0 to 9999 | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the BCD data | — | BCD 4-digit | ANY16 |

##### ■Applicable devices

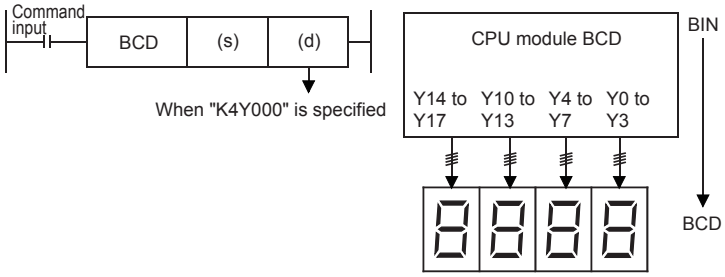
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions convert the 16-bit binary data (0 to 9999) in the device specified by (s) to BCD 4-digit data, and store the converted data in the device specified by (d).



- Data specified by (s) can be converted if it is within the range from K0 to K9999 BCD (decimal).
- The table below shows nibble specification for the data in the device specified by (s) and (d).




| (d) | Number of digits | Data range |
|---|---|---|
| K1Y0 | 1-digit | 0 to 9 |
| K2Y0 | 2-digit | 00 to 99 |
| K3Y0 | 3-digit | 000 to 999 |
| K4Y0 | 4-digit | 0000 to 9999 |

## Precautions

- Binary data is used in all operations in CPU module including arithmetic operations (+-×÷), increment and decrement instructions. When receiving digital switch information in binary-coded decimal (BCD) format into a CPU module, use the BIN(P) instructions (for converting BCD data into binary data). Furthermore, to output data to seven-segment display unit handling binary-coded decimal (BCD) data, use the BCD(P) instructions (for converting binary data into BCD data).

## Operation error

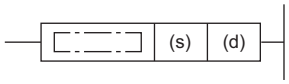
| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Data in the device specified by (s) is out of the valid range (0 to 9999). |

# Converting binary data to BCD 8-digit data

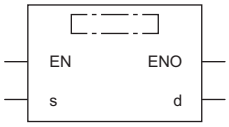
## DBCD(P)

These instructions convert the binary data in the device specified by (s) to BCD data, and store the converted data in the device specified by (d).

Binary data is used in operations in CPU module. Use this instruction to display numeric values on seven-segment display unit equipped with BCD decoder.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DBCD(EN,s,d);<br>ENO:=DBCDP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

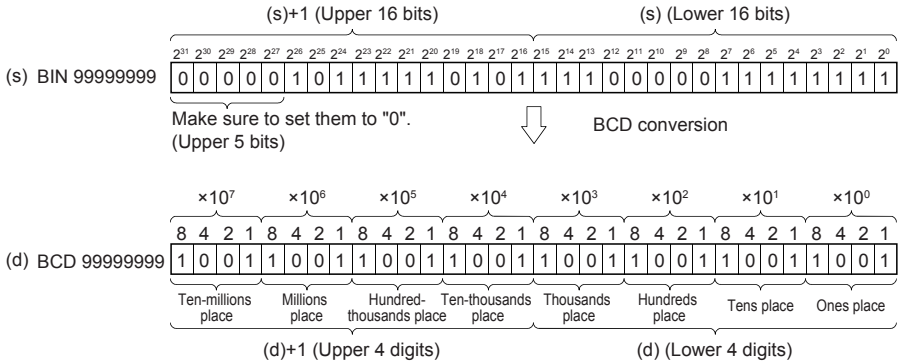
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Binary data or the head device where the binary data is stored | 0 to 99999999 | 32-bit signed binary | ANY32 |
| (d) | Head device for storing the BCD data | — | BCD 8-digit | ANY32 |

#### ■Applicable devices

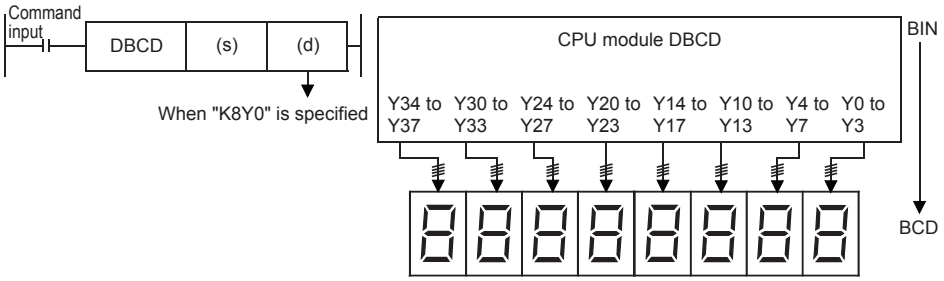
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert 32-bit binary data (0 to 99999999) in device specified by (s) to BCD 8-digit data, and store the converted data in the device specified by (d).



- Data specified by (s) can be converted if it is within the range from K0 to K99999999 BDC (decimal).
- The table below shows nibble specification for the data in the device specified by (s) and (d).




| (d)+1, (d) | Number of digits | Data range |
|---|---|---|
| K1Y0 | 1-digit | 0 to 9 |
| K2Y0 | 2-digit | 00 to 99 |
| K3Y0 | 3-digit | 000 to 999 |
| K4Y0 | 4-digit | 0000 to 9999 |
| K5Y0 | 5-digit | 00000 to 99999 |
| K6Y0 | 6-digit | 000000 to 999999 |
| K7Y0 | 7-digit | 0000000 to 9999999 |
| K8Y0 | 8-digit | 00000000 to 99999999 |

## Precautions

- When using the SEGL instruction, because BCD⇔binary conversion is automatically executed, the BCD(P) instruction do not have to be used.
- Binary data is used in all operations in CPU module including arithmetic operations (+-×÷), increment and decrement instructions. When receiving digital switch information in binary-coded decimal (BCD) format into a CPU module, use the BIN(P) instructions (for converting BCD data into binary data). Furthermore, to output data to seven-segment display unit handling binary-coded decimal (BCD) data, use the BCD(P) instructions (for converting binary data into BCD data).

## Operation error

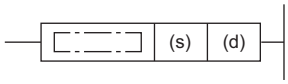
| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Data in the device specified by (s) is out of the valid range (0 to 99999999). |

# Converting BCD 4-digit data to binary data

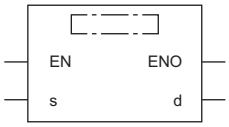
## BIN(P)

These instructions convert the binary-coded decimal data in the device specified by (s) to binary data, and store the converted data in the device specified by (d).

Use this instruction to convert a binary-coded decimal (BCD) value such as a value set by a digital switch into binary (BIN) data and to receive the converted binary data so that the data can be handled in operations in CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BIN(EN,s,d);<br>ENO:=BINP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

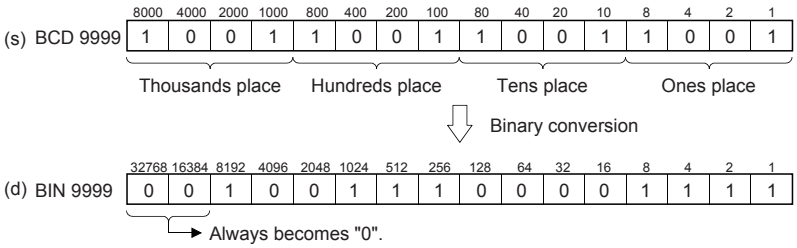
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Binary-coded decimal data or the head device where the binary-coded decimal data is stored | 0 to 9999 | BCD 4-digit | ANY16 |
| (d) | Head device for storing the binary data | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

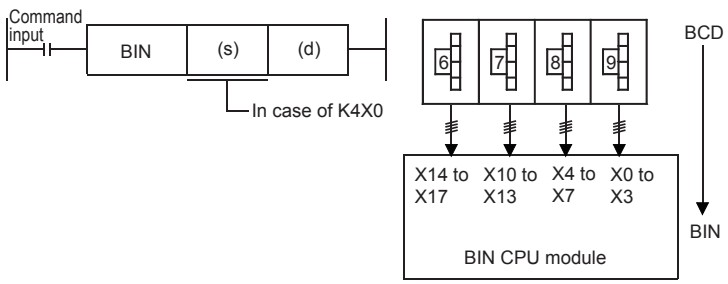
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the BCD 4-digit data (0 to 9999) in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).




- The data in the device specified by (s) can be converted if it is in the range from 0 to 9999 (BCD).
- The table below shows nibble specification for the data in the device specified by (s) and (d).



| (d) | Number of digits | Data range |
|---|---|---|
| K1X0 | 1-digit | 0 to 9 |
| K2X0 | 2-digit | 00 to 99 |
| K3X0 | 3-digit | 000 to 999 |
| K4X0 | 4-digit | 0000 to 9999 |

## Precautions

- Binary data is used in all operations in CPU module including arithmetic operations (+-×÷), increment and decrement instructions. When receiving digital switch information in binary-coded decimal (BCD) format into a CPU module, use the BIN(P) instructions (for converting BCD data into binary data). Furthermore, to output data to seven-segment display unit handling binary-coded decimal (BCD) data, use the BCD(P) instructions (for converting binary data into BCD data).

## Operation error

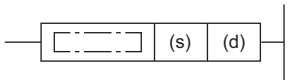
| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The value of each digit of the device specified by (s) is other than 0 to 9. (The data is not binary-coded decimal data.) |

# Converting BCD 8-digit data to binary data

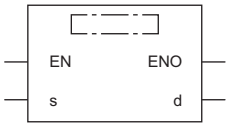
## DBIN(P)

These instructions convert the binary-coded decimal data in the device specified by (s) to binary data, and store the converted data in the device specified by (d).

Use this instruction to convert a binary-coded decimal (BCD) value such as a value set by a digital switch into binary (BIN) data and to receive the converted binary data so that the data can be handled in operations in CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DBIN(EN,s,d);<br>ENO:=DBINP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

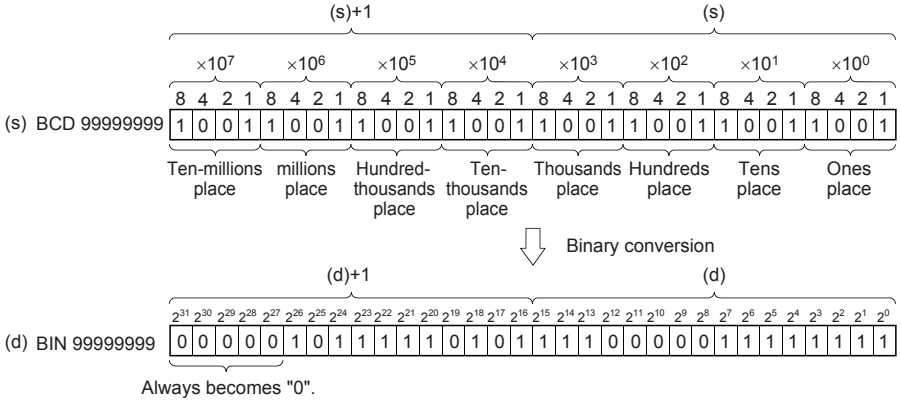
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Binary-coded decimal data or the head device where the binary-coded decimal data is stored | 0 to 99999999 | BCD 8-digit | ANY32 |
| (d) | Head device for storing the binary data | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

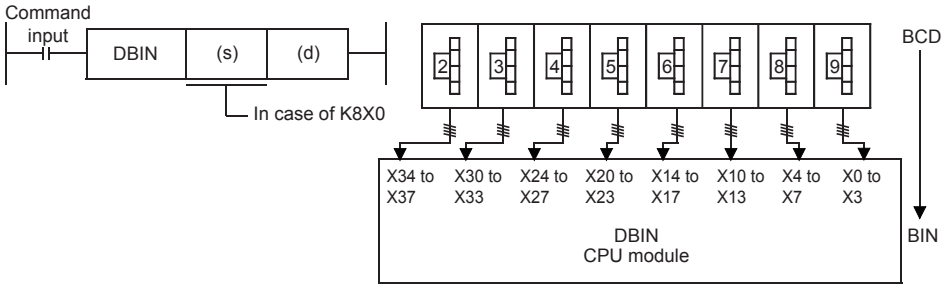
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the BCD 8-digit data (0 to 99999999) in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).



- The data in the device specified by (s) can be converted if it is in the range from 0 to 99999999 (BCD).
- The table below shows nibble specification for the data in the device specified by (s) and (d).




| (s)+1, (s) | Number of digits | Data range |
|---|---|---|
| K1X0 | 1-digit | 0 to 9 |
| K2X0 | 2-digit | 00 to 99 |
| K3X0 | 3-digit | 000 to 999 |
| K4X0 | 4-digit | 0000 to 9999 |
| K5X0 | 5-digit | 00000 to 99999 |
| K6X0 | 6-digit | 000000 to 999999 |
| K7X0 | 7-digit | 0000000 to 9999999 |
| K8X0 | 8-digit | 00000000 to 99999999 |

## Precautions

- Binary data is used in all operations in CPU module including arithmetic operations (+-×÷), increment and decrement instructions. When receiving digital switch information in binary-coded decimal (BCD) format into a CPU module, use the BIN instruction (for converting BCD data into binary data). Furthermore, to output data to seven-segment display unit handling binary-coded decimal (BCD) data, use the BCD instruction (for converting binary data into BCD data).

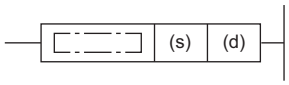
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The value of each digit of the device specified by (s) is other than 0 to 9. (The data is not binary-coded decimal data.) |

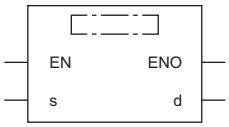
# Converting single-precision real number to 16-bit signed binary data

## FLT2INT(P)

These instructions convert the single-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

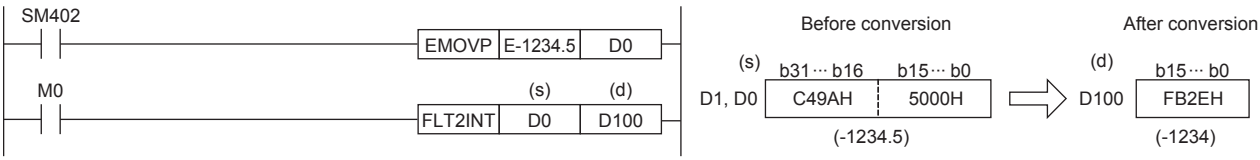
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -32768 to +32767 | Single-precision real number | ANYREAL_32 |
| (d) | Data after conversion | — | 16-bit signed binary | ANY16_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.



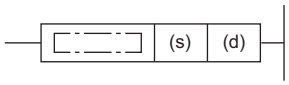

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (-32768 to +32767). |
| 3402H | When the contents of the specified device are outside the following range:<br>0, $2^{-126} \leq$ \|specified value (stored value)\| $< 2^{128}$ |
| | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

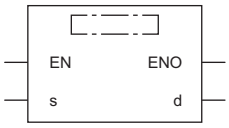
# Converting single-precision real number to 16-bit unsigned binary data

## FLT2UINT(P)

These instructions convert the single-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

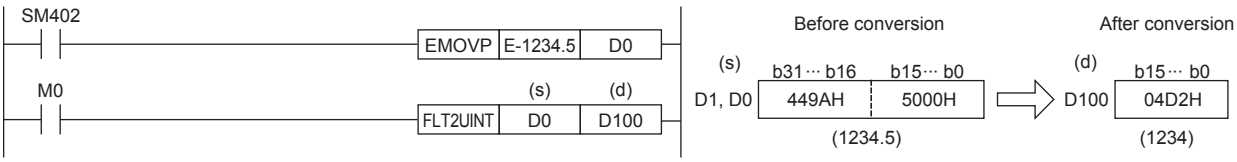
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 65535 | Single-precision real number | ANYREAL_32 |
| (d) | Data after conversion | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.



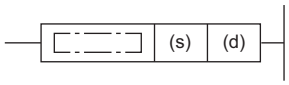

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (0 to 65535). |
| 3402H | When the contents of the specified device are outside the following range:<br>0, $2^{-126} \leq$ \|specified value (stored value)\| $< 2^{128}$ |
| | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

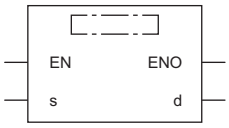
# Converting single-precision real number to 32-bit signed binary data

## FLT2DINT(P)

These instructions convert the single-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

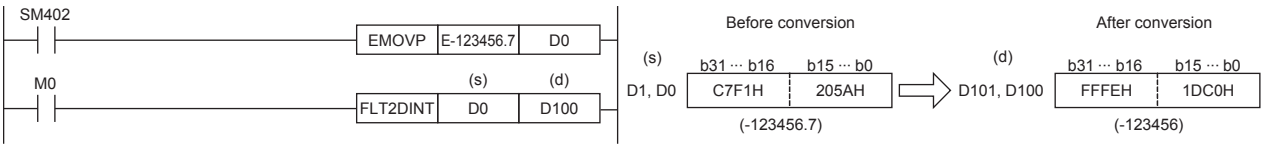
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -2147483648 to +2147483647 | Single-precision real number | ANYREAL_32 |
| (d) | Data after conversion | — | 32-bit signed binary | ANY32_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.



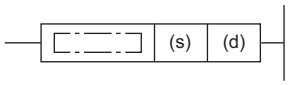

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (-2147483648 to +2147483647). |
| 3402H | When the contents of the specified device are outside the following range:<br>0, $2^{-126} \leq$ \|specified value (stored value)\| $< 2^{128}$ |
| | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

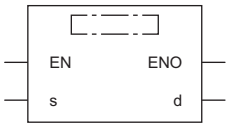
# Converting single-precision real number to 32-bit unsigned binary data

## FLT2UDINT(P)

These instructions convert the single-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

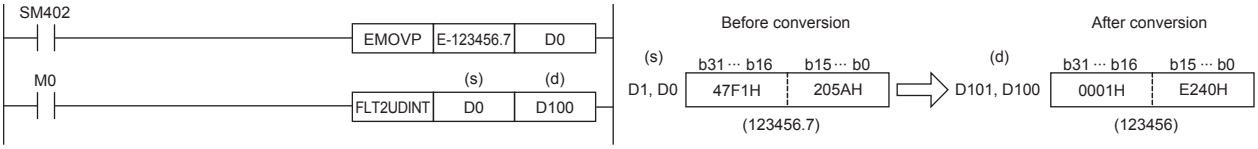
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 4294967295 | Single-precision real number | ANYREAL_32 |
| (d) | Data after conversion | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d). After conversion, the digits after the decimal point of the single-precision real number, specified with (s), are rounded off.



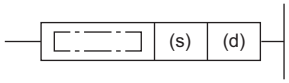

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (0 to 4294967295). |
| 3402H | When the contents of the specified device are outside the following range:<br>0, $2^{-126} \leq$ \|specified value (stored value)\| $< 2^{128}$ |
| | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

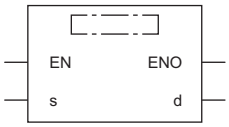
# Converting 16-bit signed binary data to 16-bit unsigned binary data

## INT2UINT(P)

These instructions convert the 16-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

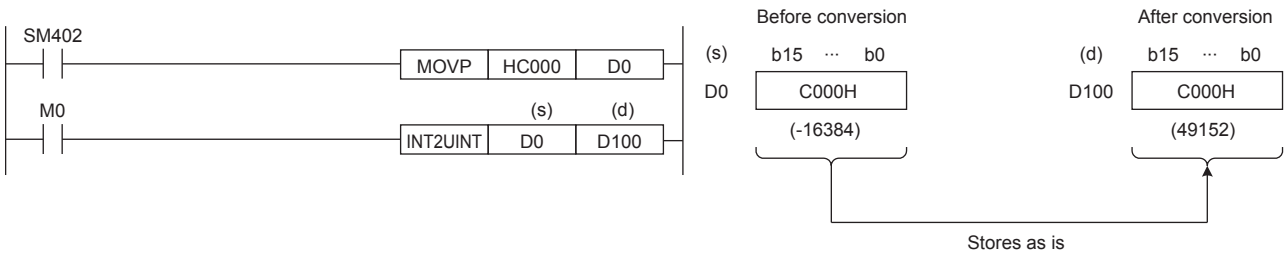
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| (d) | Data after conversion | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the result in the device specified by (d).



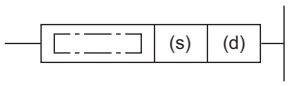

### Operation error

There is no operation error.

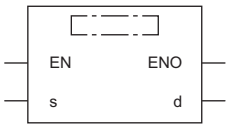
# Converting 16-bit signed binary data to 32-bit signed binary data

## INT2DINT(P)

These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

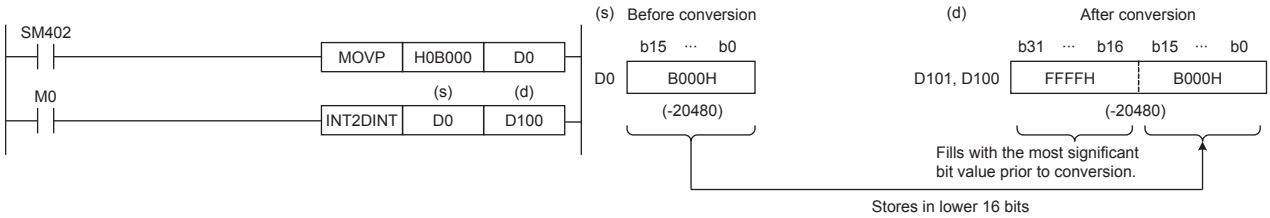
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| (d) | Data after conversion | — | 32-bit signed binary | ANY32_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).



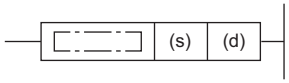

### Operation error

There is no operation error.

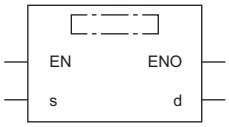
# Converting 16-bit signed binary data to 32-bit unsigned binary data

## INT2UDINT(P)

These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

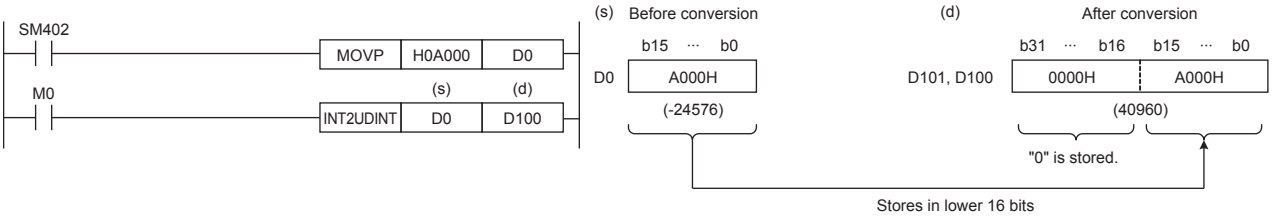
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| (d) | Data after conversion | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).



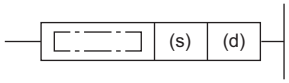

### Operation error

There is no operation error.

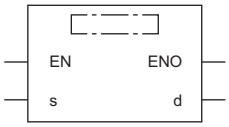
# Converting 16-bit unsigned binary data to 16-bit signed binary data

## UINT2INT(P)

These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

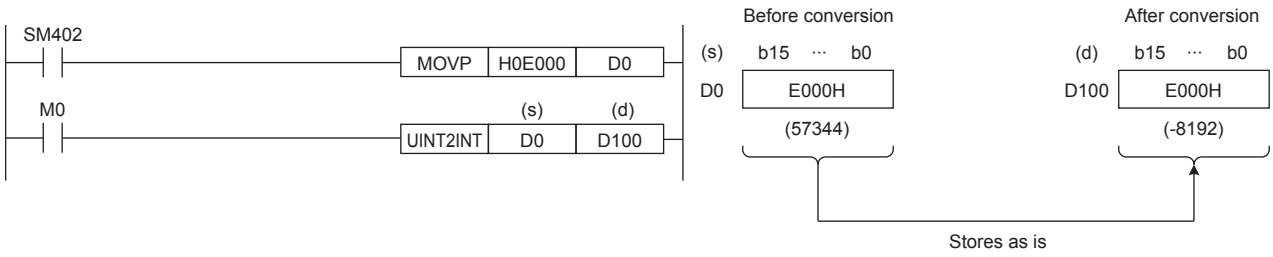
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 ro 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | Data after conversion | — | 16-bit signed binary | ANY16_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).



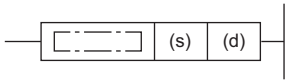

### Operation error

There is no operation error.

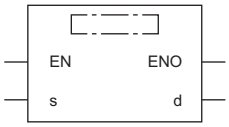
# Converting 16-bit unsigned binary data to 32-bit signed binary data

## UINT2DINT(P)

These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

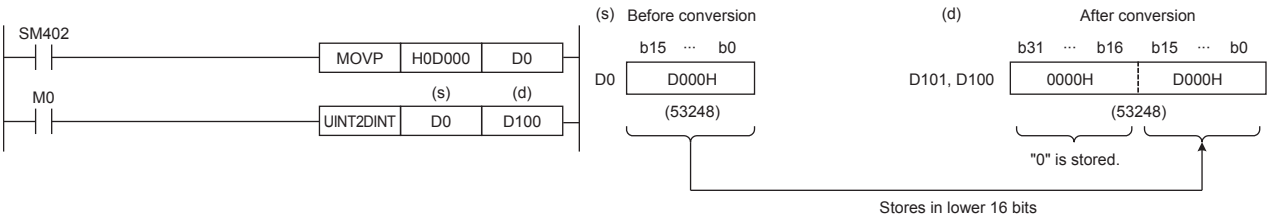
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | Data after conversion | — | 32-bit signed binary | ANY32_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).



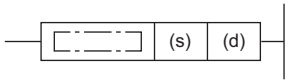

### Operation error

There is no operation error.

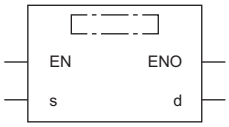
# Converting 16-bit unsigned binary data to 32-bit unsigned binary data

## UINT2UDINT(P)

These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

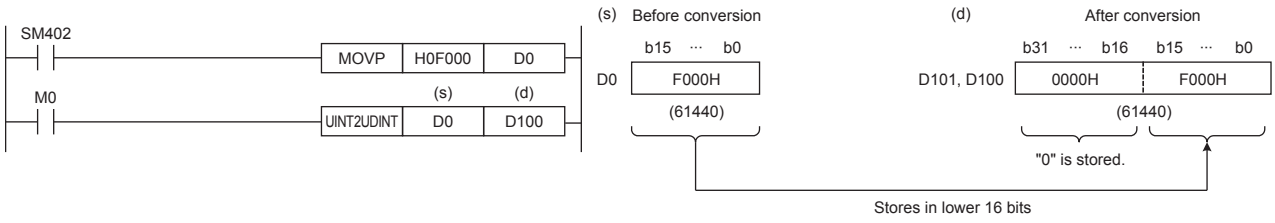
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | Data after conversion | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).



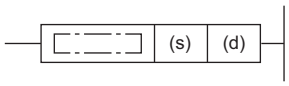

### Operation error

There is no operation error.

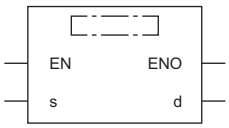
# Converting 32-bit signed binary data to 16-bit signed binary data

## DINT2INT(P)

These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

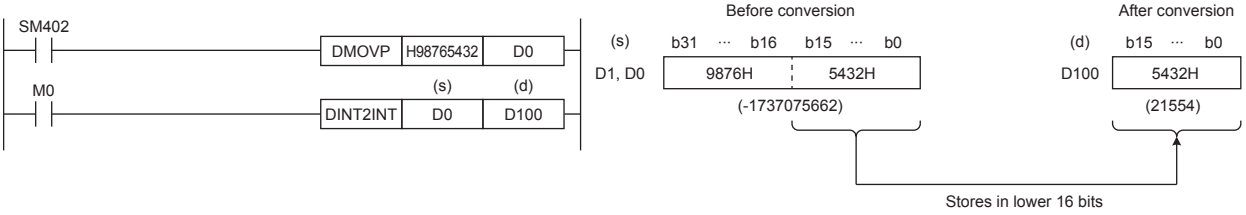
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| (d) | Data after conversion | — | 16-bit signed binary | ANY16_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).



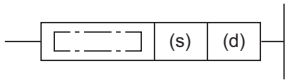

### Operation error

There is no operation error.

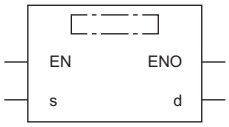
# Converting 32-bit signed binary data to 16-bit unsigned binary data

## DINT2UINT(P)

These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

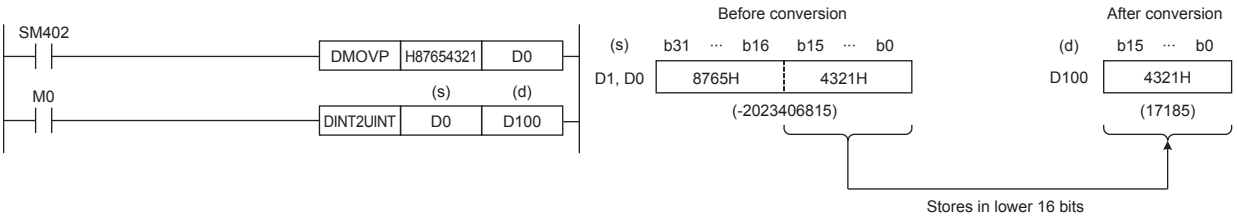
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| (d) | Data after conversion | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).



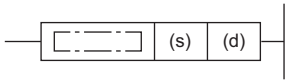

### Operation error

There is no operation error.

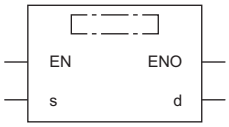
# Converting 32-bit signed binary data to 32-bit unsigned binary data

## DINT2UDINT(P)

These instructions convert the 32-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

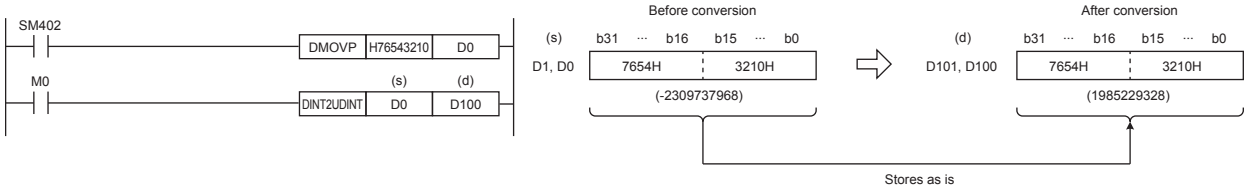
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| (d) | Data after conversion | — | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).



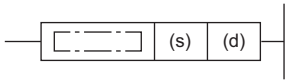

### Operation error

There is no operation error.

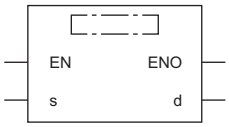
# Converting 32-bit unsigned binary data to 16-bit signed binary data

## UDINT2INT(P)

These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

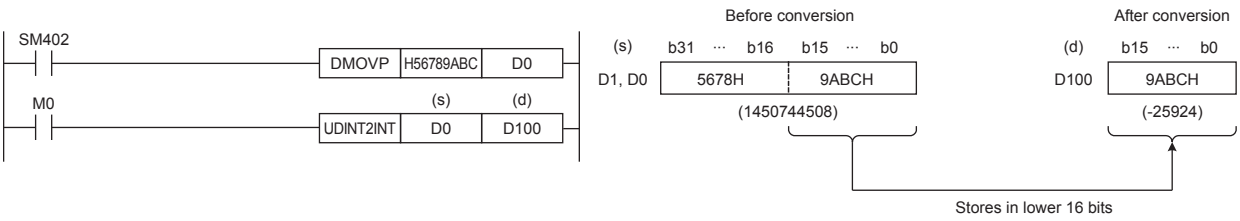
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | Data after conversion | — | 16-bit signed binary | ANY16_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).



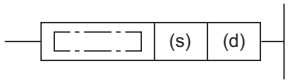

### Operation error

There is no operation error.

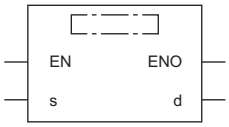
# Converting 32-bit unsigned binary data to 16-bit unsigned binary data

## UDINT2UINT(P)

These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

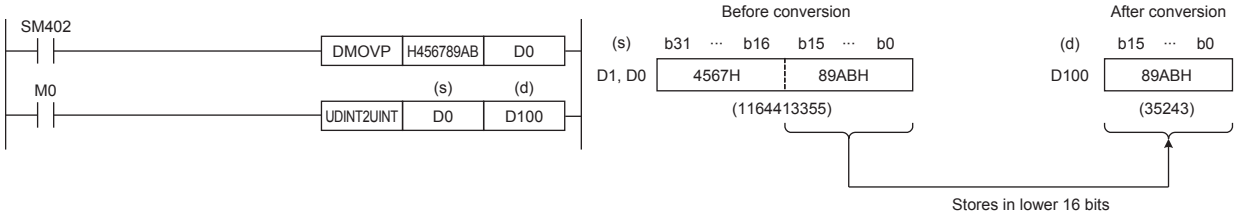
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | Data after conversion | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).



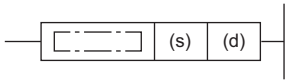

### Operation error

There is no operation error.

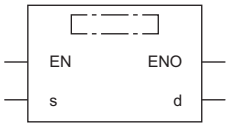
# Converting 32-bit unsigned binary data to 32-bit signed binary data

## UDINT2DINT(P)

These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

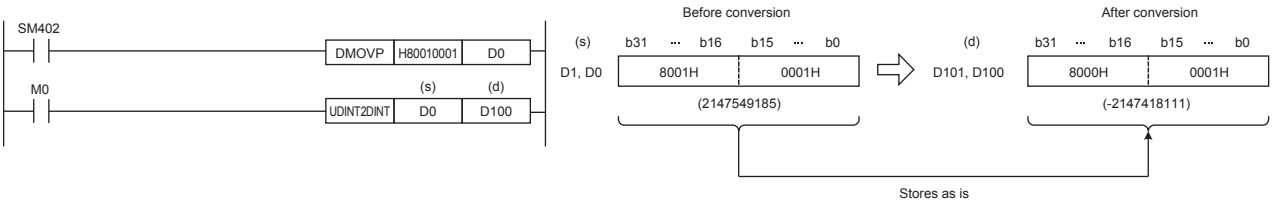
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | Data after conversion | — | 32-bit signed binary | ANY32_S |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).



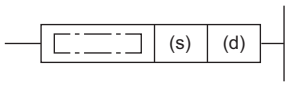

### Operation error

There is no operation error.

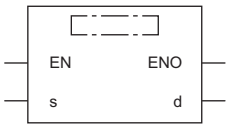
# Converting 16-bit binary data to Gray code

## GRY(P)(_U)

These instructions convert the 16-bit binary data in the device specified by (s) to 16-bit binary gray code data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=GRY(EN,s,d);<br>ENO:=GRYP(EN,s,d); | ENO:=GRY_U(EN,s,d);<br>ENO:=GRYP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

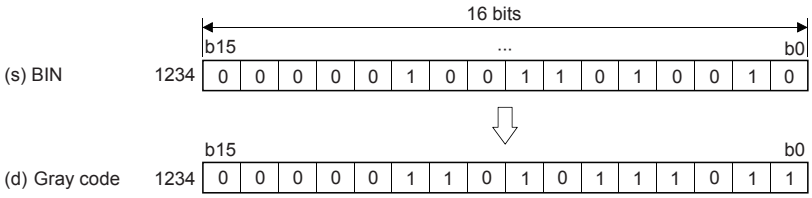
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | GRY(P) | Binary data or the head device where the binary data is stored | 0 to 32767 | 16-bit signed binary | ANY16_S |
| | GRY(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | GRY(P) | Head device for storing the gray code data after conversion | — | 16-bit signed binary | ANY16_S |
| | GRY(P)_U | | | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit binary data in the device specified by (s) to 16-bit binary gray code data, and store the converted data in the device specified by (d).




### Precautions

The data conversion speed depends on the scan time of the CPU module.

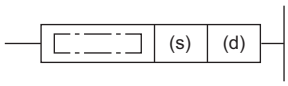
### Operation error

There is no operation error.

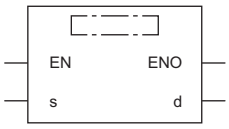
# Converting 32-bit binary data to Gray code

## DGRY(P)(_U)

These instructions convert the 32-bit binary data in the device specified by (s) to 32-bit binary gray code data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DGRY(EN,s,d);<br>ENO:=DGRYP(EN,s,d); | ENO:=DGRY_U(EN,s,d);<br>ENO:=DGRYP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

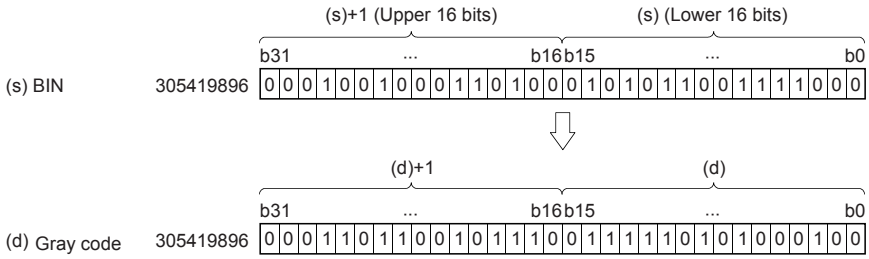
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DGRY(P) | Binary data or the head device where the binary data is stored | 0 to 2147483647 | 32-bit signed binary | ANY32_S |
| | DGRY(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DGRY(P) | Head device for storing the gray code data after conversion | — | 32-bit signed binary | ANY32_S |
| | DGRY(P)_U | | | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit binary data in the device specified by (s) to 32-bit binary gray code data, and store the converted data in the device specified by (d).




### Precautions

The data conversion speed depends on the scan time of the CPU module.

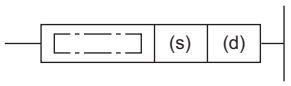
### Operation error

There is no operation error.

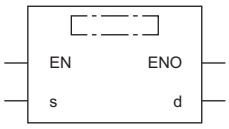
# Converting Gray code to 16-bit binary data

## GBIN(P)(_U)

These instructions convert the 16-bit binary gray code data in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=GBIN(EN,s,d);<br>ENO:=GBINP(EN,s,d); | ENO:=GBIN_U(EN,s,d);<br>ENO:=GBINP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

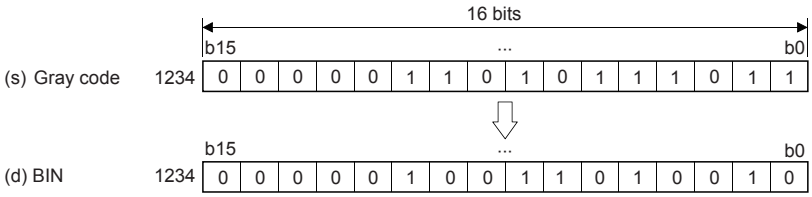
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | GBIN(P) | Gray code data or head device storing the gray code data | 0 to 32767 | 16-bit signed binary | ANY16_S |
| | GBIN(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | GBIN(P) | Head device for storing the binary data after conversion | — | 16-bit signed binary | ANY16_S |
| | GBIN(P)_U | | | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit binary gray code data in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).




### Precautions

When an input relay (X) is specified as (s), the response delay will be "Scan time of CPU module + Input filter constant".

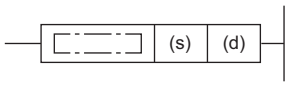
### Operation error

There is no operation error.

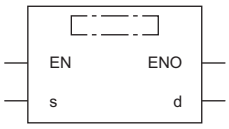
# Converting Gray code to 32-bit binary data

## DGBIN(P)(_U)

These instructions convert the 32-bit binary gray code data in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DGBIN(EN,s,d);<br>ENO:=DGBINP(EN,s,d); | ENO:=DGBIN_U(EN,s,d);<br>ENO:=DGBINP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

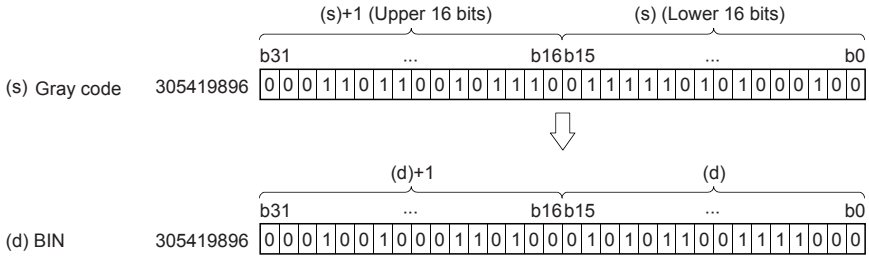
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DGBIN(P) | Gray code data or head device storing the gray code data | 0 to 2147483647 | 32-bit signed binary | ANY32_S |
| | DGBIN(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DGBIN(P) | Head device for storing the binary data after conversion | — | 32-bit signed binary | ANY32_S |
| | DGBIN(P)_U | | | 32-bit unsigned binary | ANY32_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit binary gray code data in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).




### Precautions

When an input relay (X) is specified as (s), the response delay will be "Scan time of CPU module + Input filter constant".

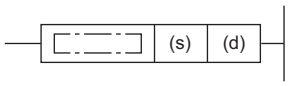
### Operation error

There is no operation error.

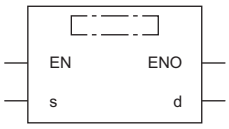
# Converting decimal ASCII to 16-bit binary data

## DABIN(P)(_U)

These instructions convert the decimal ASCII data in the device areas specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DABIN(EN,s,d);<br>ENO:=DABINP(EN,s,d); | ENO:=DABIN_U(EN,s,d);<br>ENO:=DABINP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | | ASCII data or the head device where the ASCII data is stored | — | Character string | ANYSTRING_SINGLE |
| (d) | DABIN(P) | Head device for storing the converted data | — | 16-bit signed binary | ANY16_S |
| | DABIN(P)_U | | | 16-bit unsigned binary | ANY16_U |

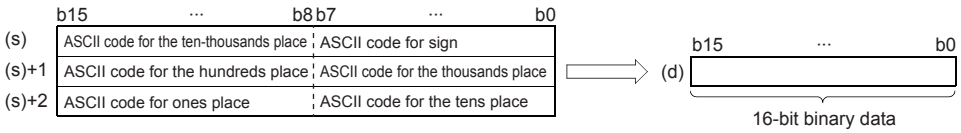
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

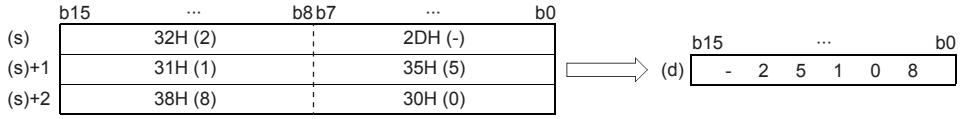
### Processing details

- These instructions convert the decimal ASCII data in the device areas specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).




**Ex.**

When the ASCII data, -25108 (signed), is specified by (s)




- The ASCII data that can be specified by (s) to (s)+2 is -32768 to +32767 for signed data, and 0 to 65535 for unsigned data.
- As signed data, "20H" is stored if the ASCII data is positive, and "2DH" is stored if the data is negative. (If a value other than "20H" and "2DH" is set, the data will be processed as positive data.) (DABIN(P))
- A value "30H" to "39H" can be set in the each place of the ASCII code.
- If a value "20H" or "00H" is set, the value will be processed as "30H".

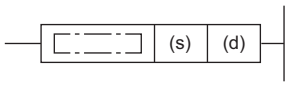
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (s) exceeds the corresponding device range. |
| 3401H | The signed data is other than 20H, 2DH. |
| | A value specified by (s) to (s)+2 for each place of the ASCII code is other than "30H" to "39H", "20H", and "00H". |
| | The ASCII data in the device specified by (s) to (s)+2 is out of the valid range (-32768 to +32767) (when a signed data is specified). |
| | The ASCII data in the device specified by (s) to (s)+2 is out of the valid range (0 to 65535) (when unsigned data is specified). |

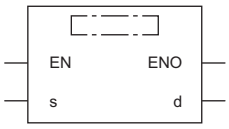
# Converting decimal ASCII to 32-bit binary data

## DDABIN(P)(_U)

These instructions convert the decimal ASCII data in the device numbers specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DDABIN(EN,s,d);<br>ENO:=DDABINP(EN,s,d); | ENO:=DDABIN_U(EN,s,d);<br>ENO:=DDABINP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | | ASCII data or the head device where the ASCII data is stored | — | Character string | ANYSTRING_SINGLE |
| (d) | DDABIN(P) | Head device for storing the converted data | — | 32-bit signed binary | ANY32_S |
| | DDABIN(P)_U | | | 32-bit unsigned binary | ANY32_U |

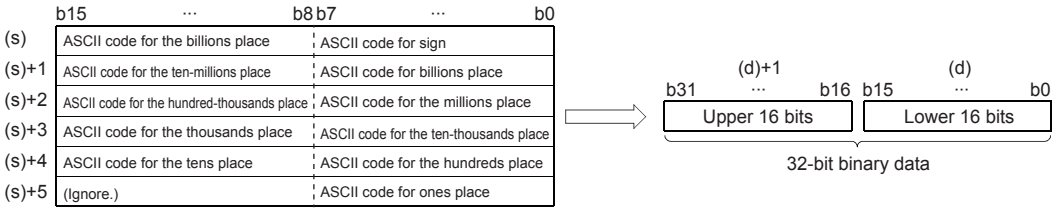
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

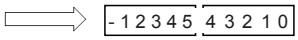
- These instructions convert the decimal ASCII data in the device numbers specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).



**Ex.**

When the ASCII data, -1234543210 (signed), is specified by (s)

| | b15 ··· b8 | b7 ··· b0 |
|---|---|---|
| (s) | 31H (1) | 2DH (-) |
| (s)+1 | 33H (3) | 32H (2) |
| (s)+2 | 35H (5) | 34H (4) |
| (s)+3 | 33H (3) | 34H (4) |
| (s)+4 | 31H (1) | 32H (2) |
| (s)+5 | | 30H (0) |




- The ASCII data that can be specified by (s) to (s)+5 is -2147483648 to +2147483647 for signed data, and 0 to 429496729 for unsigned data. The data stored in the high-order byte of (s)+5 is ignored.
- As signed data, "20H" is stored if the ASCII data is positive, and "2DH" is stored if the data is negative. (If a value other than "20H" and "2DH" is set, the data will be processed as positive data.) (DABIN(P))
- A value "30H" to "39H" can be set in the each place of the ASCII code.
- If a value "20H" or "00H" is set, the value will be processed as "30H".

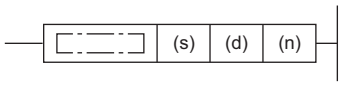
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (s) exceeds the corresponding device range. |
| 3401H | The signed data is other than 20H, 2DH. |
| | A value specified by (s) to (s)+2 for each place of the ASCII code is other than "30H" to "39H", "20H", and "00H". |
| | The ASCII data in the device specified by (s) to (s)+5 is out of the valid range (-2147483648 to +2147483647) (when a signed data is specified). |
| | The ASCII data in the device specified by (s) to (s)+5 is out of the valid range (0 to 4294967295) (when unsigned data is specified). |

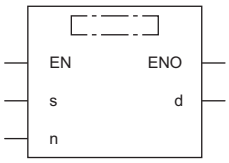
# Converting ASCII to HEX

## HEXA(P)

These instructions convert the ASCII data stored in the number of characters specified by (n) in the device numbers specified by (s) and later to HEX code data, and store the converted data in the device numbers specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=HEXA(EN,s,n,d);<br>ENO:=HEXAP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the ASCII data to be converted to hexadecimal code | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device for storing the hexadecimal code after conversion | — | 16-bit signed binary | ANY16 |
| (n) | Number of characters (number of bytes) of ASCII data to be converted | 1 to 16383 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

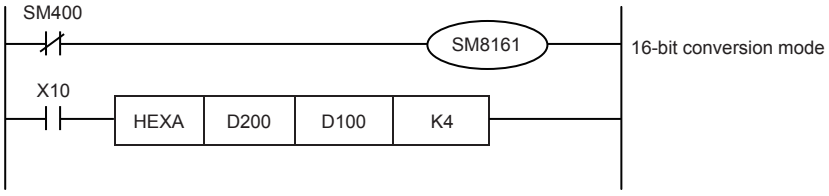
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions convert the ASCII data stored in the number of characters specified by (n) in the device numbers specified by (s) and later to HEX code data, and store the converted data in the device numbers specified by (d) and later. 16-bit conversion mode and 8-bit conversion mode options are available for these instructions. For operation in each mode, refer to the succeeding pages.
  - 16-bit conversion mode (while SM8161 is OFF)

    The ASCII data stored in high-order 8 bits and low-order 8 bits (byte) of the device specified by (s) is converted to hexadecimal code, and transferred to the device specified by (d) in units of 4 digits. The number of characters to be converted is specified by (n).
    SM8161 is also used for the RS2, ASCI(P), CCD(P), and CRC(P) instructions. When using the 16-bit conversion mode, set SM8161 to normally OFF.
    SM8161 is cleared when the CPU module mode is changed from RUN to STOP.
    Moreover, when using the 16-bit conversion mode, the ASCII data must also be stored in high-order 8 bits of the device specified by (s).
    In the following program, conversion is executed as follows:



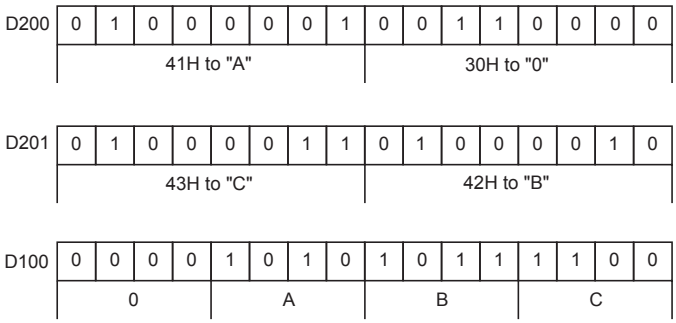
Conversion source data

| (s) | ASCII data | Hexadecimal code |
|---|---|---|
| D200 low-order | 30H | 0 |
| D200 high-order | 41H | A |
| D201 low-order | 42H | B |
| D201 high-order | 43H | C |
| D202 low-order | 31H | 1 |
| D202 high-order | 32H | 2 |
| D203 low-order | 33H | 3 |
| D203 high-order | 34H | 4 |
| D204 low-order | 35H | 5 |

Number of specified characters and conversion result "·" indicates "0".

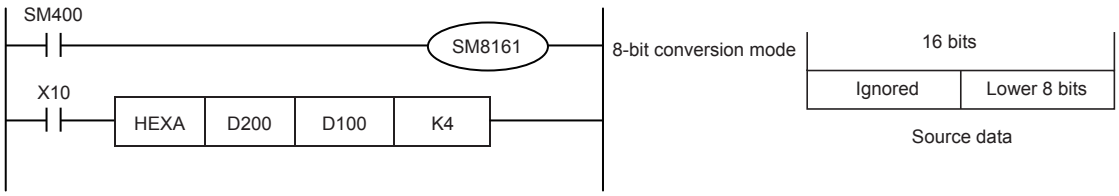
| (n) | (d) | | |
|---|---|---|---|
| | D102 | D101 | D100 |
| 1 | Does not change | | ··· 0H |
| 2 | | | ·· 0AH |
| 3 | | | · 0ABH |
| 4 | | | 0ABCH |
| 5 | | ··· 0H | ABC1H |
| 6 | | ·· 0AH | BC12H |
| 7 | | · 0ABH | C123H |
| 8 | | 0ABCH | 1234H |
| 9 | ··· 0H | ABC1H | 2345H |

When (n)=K4




• 8-bit conversion mode (while SM8161 is on)

The ASCII data stored in low-order 8 bits of the device specified by (s) is converted to hexadecimal code, and transferred to the device specified by (d) in units of 4 digits.
The number of characters to be converted is specified by (n).
SM8161 is also used for the RS2, ASCI(P), CCD(P), and CRC(P) instructions. When using the 8-bit conversion mode, set SM8161 to normally on.
SM8161 is cleared when the CPU module mode is changed from RUN to STOP.
In the following program, conversion is executed as follows:



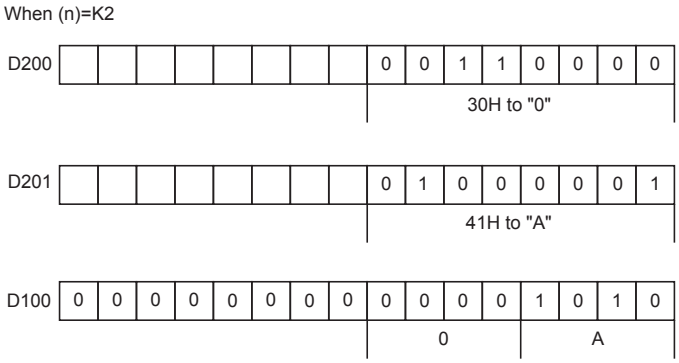

Conversion source data

| (s) | ASCII data | Hexadecimal code |
|---|---|---|
| D200 | 30H | 0 |
| D201 | 41H | A |
| D202 | 42H | B |
| D203 | 43H | C |
| D204 | 31H | 1 |
| D205 | 32H | 2 |
| D206 | 33H | 3 |
| D207 | 34H | 4 |
| D208 | 35H | 5 |

Number of specified characters and conversion result "." indicates "0".

| (n) | (d) | | |
|---|---|---|---|
| | D102 | D101 | D100 |
| 1 | Does not change | | ··· 0H |
| 2 | | | ·· 0AH |
| 3 | | | · 0ABH |
| 4 | | | 0ABCH |
| 5 | | ··· 0H | ABC1H |
| 6 | | ·· 0AH | BC12H |
| 7 | | · 0ABH | C123H |
| 8 | | 0ABCH | 1234H |
| 9 | ···0H | ABC1H | 2345H |

When (n)=K2




## Precautions

- Make sure that only ASCII codes "0" to "9" and "A" to "F" are stored in the device specified by (s).
- If ASCII data is not stored in the device specified for (s) by the HEXA(P) instructions, an operation error occurs and conversion into hexadecimal code is disabled. Especially, note that when SM8161 is OFF (16-bit conversion mode), ASCII code should be stored in high-order 8 bits of the device specified by (s).
- The number of points occupied by the device specified by (d) varies depending on the ON/OFF status of SM8161. When SM8161 is on (8-bit conversion mode), as many points as the number of characters are occupied, and when SM8161 is OFF (16-bit conversion mode) as many points as the (number of characters ÷2) are occupied.
- The SM8161 flag is also used for the RS2, ASCI(P), CCD(P) and CRC(P) instructions. When using these instructions and the HEXA(P) instructions in the same program, make sure to set SM8161 to ON or OFF just before each instruction so that SM3161 does not affect another instruction.

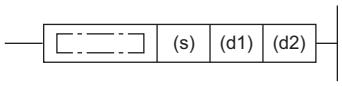
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) number of devices specified by (s) and (d) exceeds the corresponding device range. |
| 2821H | The range specified by (s) and (d) overlaps. |
| 3401H | An ASCII code other than 30H to 39H, and 41H to 46H is set in the device specified by (s). |
| 3405H | The value specified in (n) is outside the range specified below.<br>1 to 16383 |

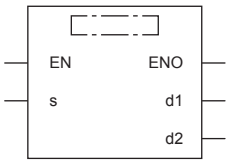
# Converting character string to 16-bit binary data

## VAL(P)(_U)

These instructions convert the character string in the device numbers specified by (s) and later to 16-bit binary data, and store the number of digits in the device specified by (d1) and the binary data in the device specified by (d2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=VAL(EN,s,d1,d2);<br>ENO:=VALP(EN,s,d1,d2); | ENO:=VAL_U(EN,s,d1,d2);<br>ENO:=VALP_U(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | | Character string to be converted to binary data, or head device for storing the character string. | — | Character string | ANYSTRING_SINGLE |
| (d1) | VAL(P) | Head device for storing the number of digits of the binary data after conversion | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | VAL(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |
| (d2) | VAL(P) | Head device for storing the binary data after conversion | — | 16-bit signed binary | ANY16_S |
| | VAL(P)_U | | | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

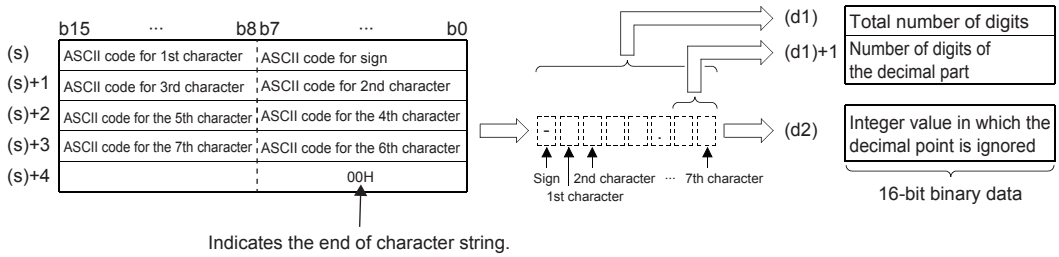
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d1) | ○ | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

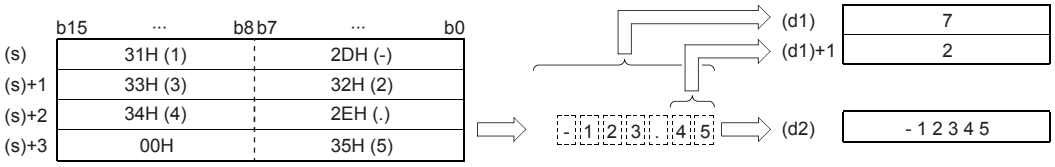
- These instructions convert the character string in the device numbers specified by (s) and later to 16-bit binary data, and store the number of digits in the device specified by (d1) and the binary data in the device specified by (d2). When converting a character string into binary data, the data from the device number specified by (s) to a device number storing "00H" is handled as a character string.

- The total number of digits stored in (d1) is the total number of characters (including the sign and decimal point) representing the numeric value. The number of digits in the decimal part stored in (d1)+1 is the number of characters representing the decimal part after 2EH (.). The 16-bit binary data stored in (d2) is binary value converted from a character string with the decimal point ignored.




**Ex.**

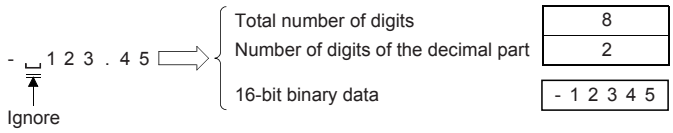
When the character string "-123.45" (signed) is specified by (s) and later




- The total number of characters of the character string specified by (s) is 2 to 8 characters.
- In the character string specified by (s), the number of characters that form the decimal part is 0 to 5 characters. However, be sure to specify "Total number of digits - 3" or below.
- The range of the character string of the numeric value that can be converted to a binary value is -32768 to +32767 for a signed value with the decimal point ignored, and 0 to 65535 for an unsigned value. A character string of a numeric value excluding the sign and decimal point can be specified only within the range of 30H to 39H. (Value with the decimal point ignored ... "-12345.6" becomes "-123456".)
- When representing a positive numeric value, 20H is set in the sign, and when representing a negative numeric value, 2DH is set.
- 2EH is set in the decimal point.
- When "20H (space)" or "30H (0)" exists between the sign and the first non-zero number in a character string specified by (s), "20H" or "30H" is ignored during conversion to a binary value.

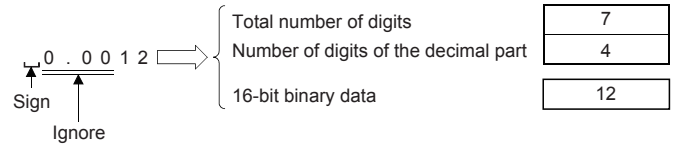
**Ex.**

When "20H" exists between the sign and the first non-zero number (a signed value is specified)




**Ex.**

When "30H" exists between the sign and the first non-zero number




## Precautions

- Store signed data, "space (20H)" or "- (2DH)" only in the 1st byte (low-order 8 bits of the head device set in (s)). Only the ASCII data "0 (30H)" to "9 (39H)", "space (20H)" and "decimal point (2EH)" can be stored from the 2nd byte to the "00H" at the end of the character string in (s). If "- (2DH)" is stored in the 2nd byte or later, an operation error occurs.

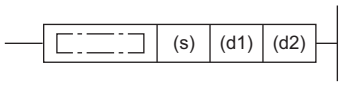
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (d1) exceeds the corresponding device range. |
| | When "00H" is not set in the corresponding device range after the device specified in (s). |
| 3401H | The number of characters of the character string specified by (s) is other than 2 to 8 characters. |
| | The number of characters of the decimal part of the character string specified by (s) is other than 0 to 5 characters. |
| | The relationship between the total number of characters specified by (s) and the number of characters of the decimal part is other than that described below.<br>Total number of characters - 3≥Number of characters in the decimal part |
| | An ASCII code other than 20H, 2DH is set in the sign. (a signed value is specified) |
| | An ASCII code other than 30H to 39H, and 2EH (decimal point) is set in the digits of each number |
| | Two or more decimal points are set. |
| | The converted binary value exceeds the range that can be converted by each instruction.<br>Signed operation: -32768 to +32767, unsigned operation: 0 to 65535 |

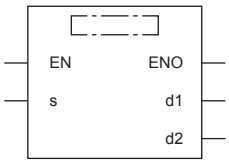
# Converting character string to 32-bit binary data

## DVAL(P)(_U)

These instructions convert the character string in the device numbers specified by (s) and later to 32-bit binary data, and store the number of digits in the device specified by (d1) and the binary data in the device specified by (d2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DVAL(EN,s,d1,d2);<br>ENO:=DVALP(EN,s,d1,d2); | ENO:=DVAL_U(EN,s,d1,d2);<br>ENO:=DVALP_U(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | | Character string to be converted to binary data, or head device for storing the character string. | — | Character string | ANYSTRING_SINGLE |
| (d1) | DVAL(P) | Head device for storing the number of digits of the binary data after conversion | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | DVAL(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |
| (d2) | DVAL(P) | Head device for storing the binary data after conversion | — | 32-bit signed binary | ANY32_S |
| | DVAL(P)_U | | | 32-bit unsigned binary | ANY32_U |

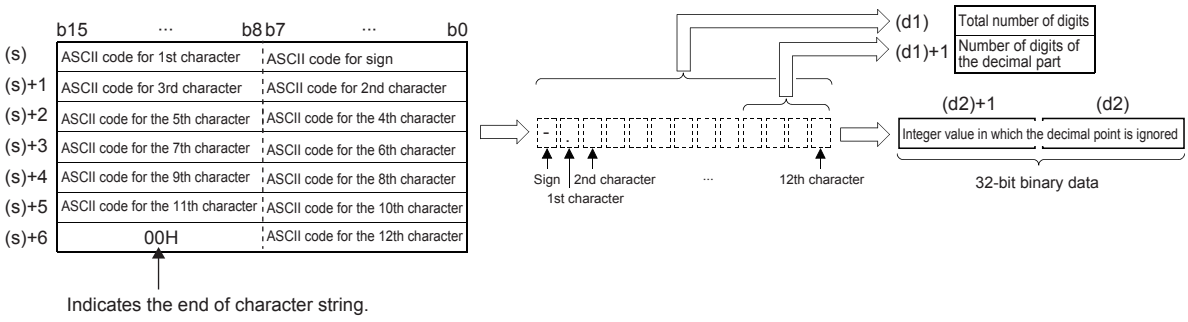
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d1) | ○ | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

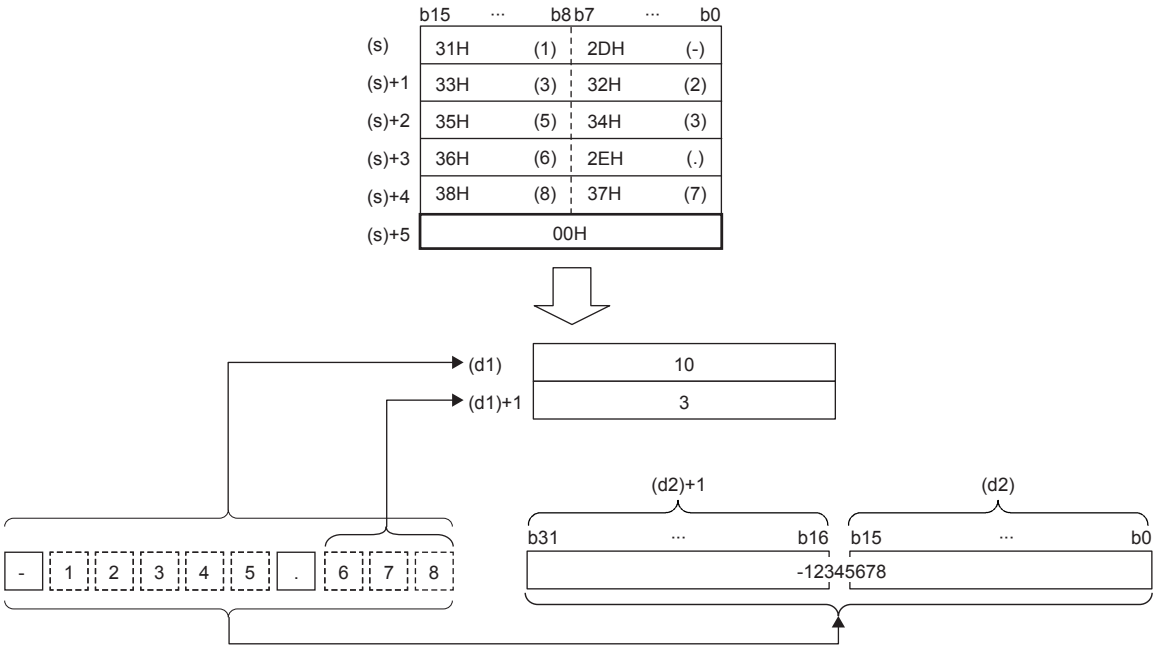
## Processing details

- These instructions convert the character string in the device numbers specified by (s) and later to 32-bit binary data, and store the number of digits in the device specified by (d1) and the binary data in the device specified by (d2). When converting a character string into binary data, the data from the device number specified by (s) to a device number storing "00H" is handled as a character string.
- The total number of digits stored in (d1) is the total number of characters (including the sign and decimal point) representing the numeric value. The number of digits in the decimal part stored in (d1)+1 is the number of characters representing the decimal part after 2EH (.). The 32-bit binary data stored in (d2) is binary value converted from a character string with the decimal point ignored.




**Ex.**

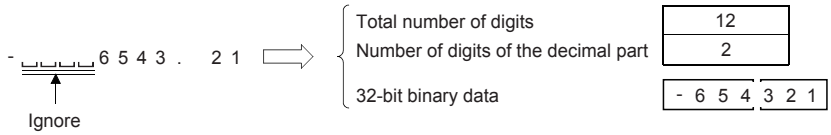
When the character string "-12345.678" (signed) is specified by (s) and later




- The total number of characters of the character string specified by (s) is 2 to 13 characters.
- In the character string specified by (s), the number of characters that form the decimal part is 0 to 10 characters. However, be sure to specify "Total number of digits - 3" or below.
- The range of the character string of the numeric value that can be converted to a binary value is -2147483648 to 2147483647 for a signed value with the decimal point ignored, and 0 to 4294967295 for an unsigned value. A character string of a numeric value excluding the sign and decimal point can be specified only within the range of 30H to 39H. (Value with the decimal point ignored ... "-12345.6" becomes "-123456".)
- When representing a positive numeric value, 20H is set in the sign, and when representing a negative numeric value, 2DH is set.
- Set 2EH in the decimal point.
- When "20H (space)" or "30H (0)" exists between the sign and the first non-zero number in a character string specified by (s), "20H" or "30H" is ignored during conversion to a binary value.

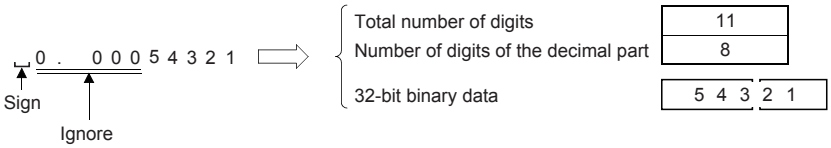
Ex.

When "20H" exists between the sign and the first non-zero number (a signed value is specified)




Ex.

When "30H" exists between the sign and the first non-zero number




## Precautions

- Store sign data, "space (20H)" or "- (2DH)" in the 1st byte (low-order 8 bits of the head device set in (s)). Only the ASCII data "0 (30H)" to "9 (39H)", "space (20H)" and "decimal point (2EH)" can be stored from the 2nd byte to the "00H" at the end of the character string in (s). If "- (2DH)" is stored in the 2nd byte or later, an operation error occurs.

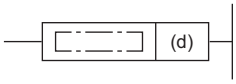
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (d1) exceeds the corresponding device range. |
| | When "00H" is not set in the corresponding device range after the device specified in (s). |
| 3401H | The number of characters of the character string specified by (s) is other than 2 to 13 characters. |
| | The number of characters of the decimal part of the character string specified by (s) is other than 0 to 10 characters. |
| | The relationship between the total number of characters specified by (s) and the number of characters of the decimal part is other than that described below.<br>Total number of characters - 3≥Number of characters in the decimal part |
| | An ASCII code other than 20H, 2DH is set in the sign. |
| | Two or more decimal points are set. |
| | The converted binary value exceeds the range that can be converted by each instruction.<br>Signed operation: -2147483648 to +2147483647, unsigned operation: 0 to 4294967295 |

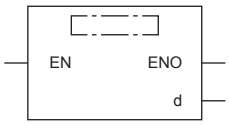
# Two's complement of 16-bit binary data (sign inversion)

## NEG(P)

These instructions invert the sign of the 16-bit binary data in the device specified by (d), and store the resultant data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=NEG(EN,d);<br>ENO:=NEGP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

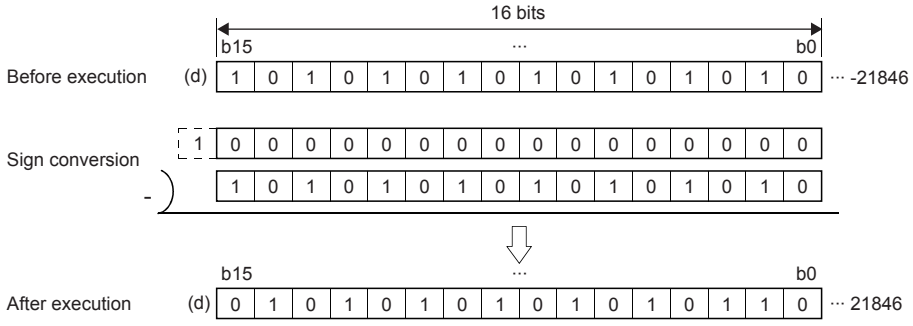
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for storing the data that performs two's complement | -32768 to +32767 | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions invert the sign of the 16-bit binary data in the device specified by (d), and store the resultant data in the device specified by (d).
- They are used when a positive or negative sign is to be inverted.




### Precautions

Note that data is inverted in every operation cycle in a continuous operation type (NEG) instruction.

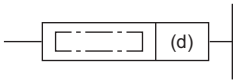
### Operation error

There is no operation error.

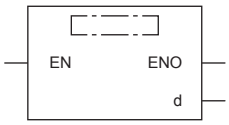
# Two's complement of 32-bit binary data (sign inversion)

## DNEG(P)

These instructions invert the sign of the 32-bit binary data in the device specified by (d), and store the resultant data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DNEG(EN,d);<br>ENO:=DNEGP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

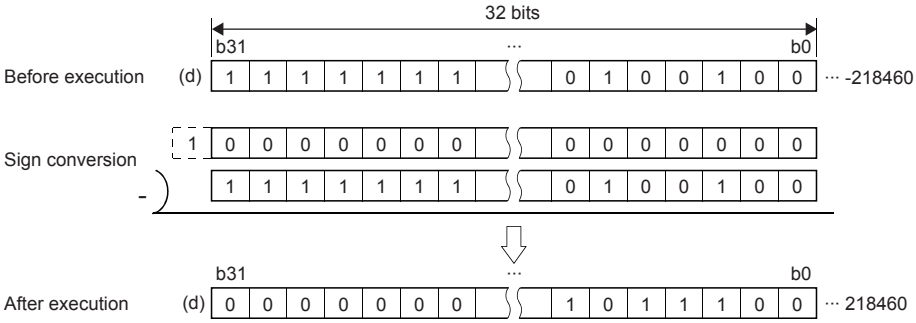
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for storing the data that performs two's complement | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions invert the sign of the 32-bit binary data in the device specified by (d), and store the resultant data in the device specified by (d).
- They are used when a positive or negative sign is to be inverted.




### Precautions

Note that data is inverted in every operation cycle in a continuous operation type (DNEG) instruction.

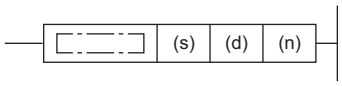
### Operation error

There is no operation error.

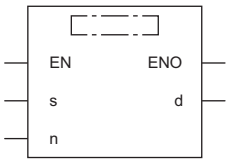
# Decoding from 8 to 256 bits

## DECO(P)

These instructions decode the lower-order (n) bits of the device specified by (s), and store the result in the 2 (to the power (n)) bit from the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DECO(EN,s,n,d);<br>ENO:=DECOP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Decode data or device number for storing the decode data | — | Bit/16-bit signed binary | ANY_ELEMENTARY |
| (d) | Head device for storing the decode result | — | Bit/16-bit unsigned binary | ANY_ELEMENTARY*1 |
| (n) | Valid bit length | 1 to 8 | 16-bit unsigned binary | ANY16 |

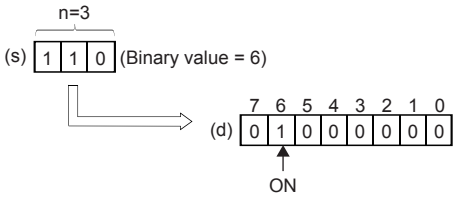
*1 When using a bit-type label, use a global label assigned to a bit device.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions turn ON the bit position of the device specified by (d) in correspondence to the BIN value specified by the lower-order (n) bits of (s).




- When (n) is 0, no processing is performed, and the contents of the device specified by (d) do not change.
- The bit device is handled as a device storing one-bit data and the word device is handled as a device storing 16-bit data.

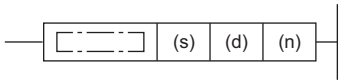
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (s) exceeds the corresponding device range. |
| | The device specified by (d) exceeds the corresponding device range. |
| 3401H | (d) is specified as a bit device and (n) is other than 0 to 8. |
| | (d) is specified as a word device and (n) is other than 0 to 4. |

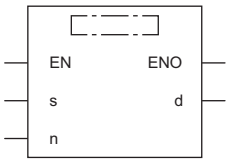
# Encoding from 256 to 8 bits

## ENCO(P)

These instructions encode the $2^{(n)}$ bits of data from the device specified by (s), and store it in (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ENCO(EN,s,n,d);<br>ENO:=ENCOP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the encode data | — | Bit/16-bit unsigned binary | ANY_ELEMENTARY*1 |
| (d) | Device number for storing the encoding result | — | 16-bit signed binary | ANY_ELEMENTARY |
| (n) | Valid bit length | 1 to 8 | 16-bit unsigned binary | ANY16 |

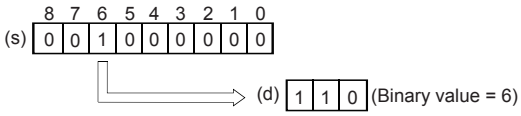
*1 When using a bit-type label, use a global label assigned to a bit device.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions store into (d) the binary value corresponding to the bit whose value is 1 in the data with $2^{(n)}$ bits.




- When (n) is 0, no processing is performed, and the contents of the device specified by (d) do not change.
- The bit device is handled as a device storing one-bit data and the word device is handled as a device storing 16-bit data.
- If two or more bits are 1, the higher bit position is processed.

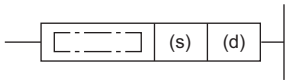
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (s) exceeds the corresponding device range. |
| 3401H | The entire data from (s) to $2^{(n)}$ number of bits is 0. |
| | (s) is specified as a bit device and (n) is other than 0 to 8. |
| | (s) is specified as a word device and (n) is other than 0 to 4. |

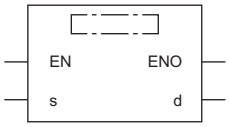
# Seven-segment decoding

## SEGD(P)

This instruction decodes data, and turns the seven-segment display unit (1 digit) ON.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SEGD(EN,s,d);<br>ENO:=SEGDP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device to be decoded | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Device number storing the data to be displayed in the seven-segment display unit | — | 16-bit signed binary | ANY16 |

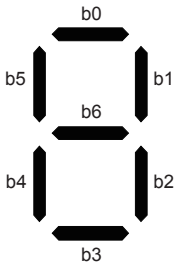
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

*1 X cannot be used.

## Processing details

- "0" to "F" (hexadecimal numbers) in low-order 4 bits (1 digit) of (s) are decoded to data for the seven-segment display unit, and stored in the low-order 8 bits of (d). Low-order 8 bits of (d) are occupied, and high-order 8 bits do not change.
- Seven-segment decode table is as follows.

| (s) | | | | | Seven-segment data components | (d) | | | | | | | | | | | Display data |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hexadecimal | b3 | b2 | b1 | b0 | | b15 | to | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | |
| 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |  |
| 2 | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |  |
| 3 | 0 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |  |
| 4 | 0 | 1 | 0 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |  |
| 5 | 0 | 1 | 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |  |
| 6 | 0 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |  |
| 7 | 0 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |  |
| 8 | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
| 9 | 1 | 0 | 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |  |
| A | 1 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |  |
| B | 1 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  |
| C | 1 | 1 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |  |
| D | 1 | 1 | 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |  |
| E | 1 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |  |
| F | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |  |

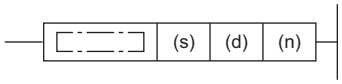
## Operation error

There is no operation error.

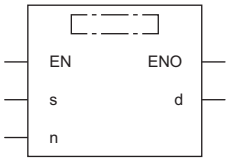
# Seven Segment With Latch

## SEGL

This instruction controls one or two sets of 4-digit seven-segment display units having the latch function.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SEGL(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device converted into the BCD format | 0 to 9999 | 16-bit signed binary | ANY16 |
| (d) | Head Y number to be output | — | bit | ANY_BOOL |
| (n) | Parameter number | 0 to 7 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |

*1 Only Y can be used.

## Processing details

The 4-digit numeric value stored in (s) is converted into BCD data, and each digit is output to the seven-segment display unit with the BCD decoder by the time division method. For (s), binary data ranging from 0 to 9999 is valid.

### ■When using one set of 4 digits (n = K0 to K3)

A 4-digit numeric value stored in (s) is converted from binary into BCD, and each digit is output in turn from (d) to (d)+3 by the time division method. The strobe signal is output in turn from (d)+4 to (d)+7 by the time division method also to latch one set of 4-digit seven-segment display unit.

### ■When using two sets of 4 digits (n = K4 to K7)

1st set of 4 digits

A 4-digit numeric value stored in (s) is converted from binary into BCD, and its each digit is output in turn from (d) to (d)+3 by the time division method. The strobe signal is output in turn from (d)+4 to (d)+7 by the time division method also to latch the first set of 4-digit seven-segment display unit.

2nd set of 4 digits

A 4-digit numeric value stored in (s)+1 is converted from binary into BCD, and its each digit is output in turn from (d)+10 to (d)+13 by the time division method. The strobe signal is output in turn from (d)+4 to (d)+7 by the time division method also to latch the second set of 4-digit seven-segment display unit.

For the connection example of two seven-segment display units, refer to the following manual.

MELSEC iQ-F FX5U User's Manual (Hardware)
MELSEC iQ-F FX5UC User's Manual (Hardware)

## Precautions

- The scan time (operation cycle) multiplied by 12 is required to update (one or two sets of) the 4-digit display.
- While the command input is ON, the operation is repeated. When the command contact is set to OFF in the middle of an operation, the operation is paused. When the command contact is set to ON again, the operation is started from the beginning.
- When one set of 4 digits is used, one device is occupied from the device specified in (s) and eight devices are occupied from the device specified in (d).
- When two sets of 4 digits are used, two devices are occupied from the device specified in (s) and twelve devices are occupied from the device specified in (d).
- SEGL instruction is executed in synchronization with the scan time (operation cycle) of the CPU module. For achieving a series of display, the scan time of the PLC should be 10 ms or more. If the scan time is less than 10 ms, use the constant scan mode so that the scan time exceeds 10 ms.
- Use a transistor output type CPU module.
- The SEGL instruction can only be executed four times in a program.
- When a constant (K or H) is specified as (s), operate as shown below.

  When one set of 4 digits is used : It operates considering the constant specified by (s) as the 1st set.

  When two sets of 4 digits are used : It operates considering the constant specified by (s) as the 1st set, and the 2nd set fixed to 0.

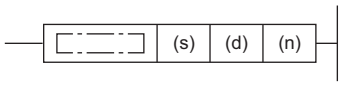
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s), (d) exceeds the corresponding device range. |
| 3405H | (n) is other than 0 to 7. |
| | The value specified by (s), (s)+1 is other than 0 to 9999. |
| 1811H | The number of the SEGL instructions which are used simultaneously exceeds four. |

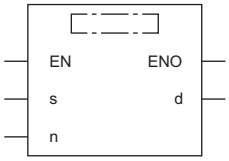
# Separating 4 bits from 16-bit data

## DIS(P)

These instructions store the data equivalent of the (n) nibbles (1-nibble/ 4-bits) of the 16-bit binary data specified by (s) in to the lower-order 4 bits of (n) number of devices starting from the one specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DIS(EN,s,n,d);<br>ENO:=DISP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

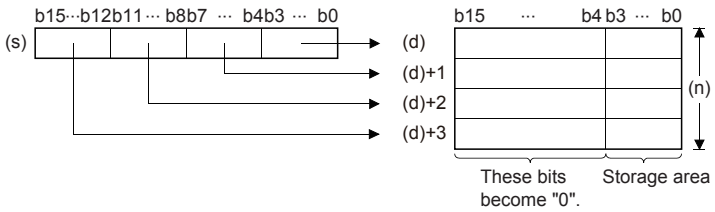
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the data to be separated | — | 16-bit signed binary | ANY16 |
| (d) | Head device storing separated data | — | 16-bit signed binary | ANY16 |
| (n) | Number of separations (0 indicates no processing is performed) | 1 to 4 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions store the data equivalent of the (n) nibbles (1-nibble/ 4-bits) of the 16-bit binary data specified by (s) in to the lower-order 4 bits of (n) number of devices starting from the one specified by (d).




- The higher-order 12 bits of (n) number of devices starting from the one specified by (s) becomes 0.
- When (n) is 0, no processing is performed, and the contents of the (n) number of devices starting from the one specified by (d) do not change.

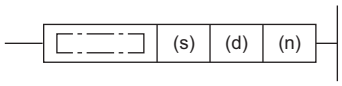
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) number of points from (d) exceed the corresponding device range. |
| 3401H | (n) is other than 0 to 4. |

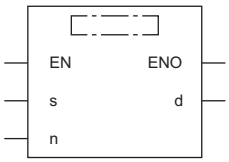
# Connecting 4 bits to 16-bit data

## UNI(P)

These instructions link the lower-order 4 bits of the 16-bit binary data of the (n) number of devices starting from the one specified by (s) to the device storing 16-bit binary data specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=UNI(EN,s,n,d);<br>ENO:=UNIP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

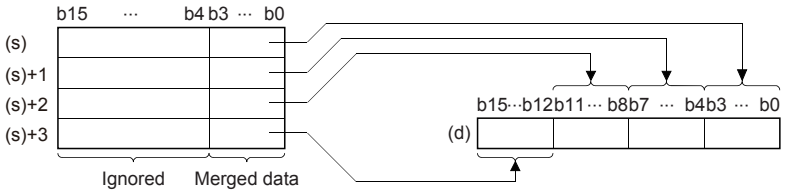
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the data to be linked | — | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the linked data | — | 16-bit signed binary | ANY16 |
| (n) | Number of links | 1 to 4 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions link the lower-order 4 bits of the 16-bit binary data of the (n) number of devices starting from the one specified by (s) to the device storing 16-bit binary data specified by (d).




- The higher-order (4-n) nibble bits of the device specified by (d) becomes 0.
- When (n) is 0, no processing is performed, and the contents of the device specified by (d) do not change.

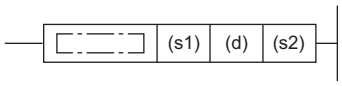
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) number of points from (d) exceed the corresponding device range. |
| 3401H | (n) is other than 0 to 4. |

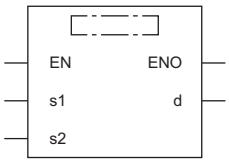
# Separating the specified number of bits

## NDIS(P)

These instructions separate each bit of the data in the device numbers specified by (s1) onwards into bit units specified by (s2), and store the separated data in the device number specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=NDIS(EN,s1,s2,d);<br>ENO:=NDISP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

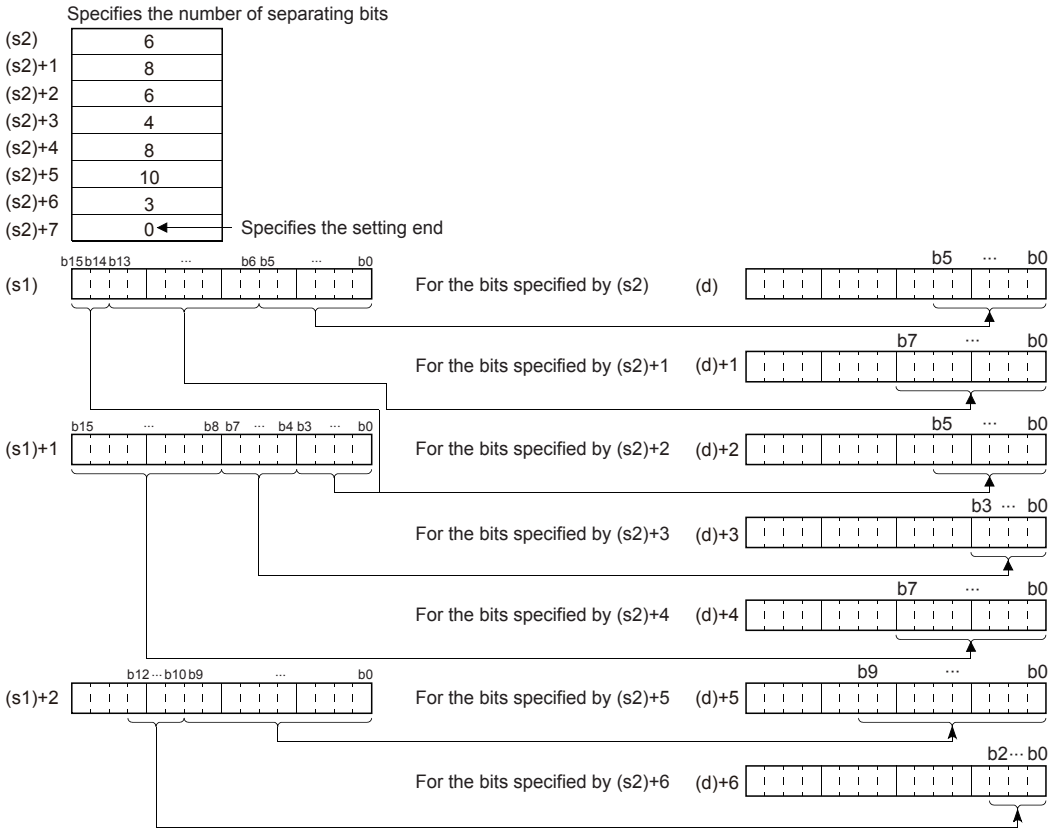
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device for storing the data to be separated | — | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the separated data | — | 16-bit signed binary | ANY16 |
| (s2) | Head device for storing the separation unit | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

## Processing details

- These instructions separate each bit of the data in the device numbers specified by (s1) and later into bit units specified by (s2), and store the separated data in the device numbers starting from the one specified by (d).




- The number of separation bits specified by (s2) can be specified within the range of 1 to 16 bits.
- The number of bits specified in devices from the device number specified by (s2) up to the device number in which "0" is stored are processed as the number of separation bits.
- If the device numbers specified by (s1), (s2), (d) are partially overlapping, an operation error occurs.

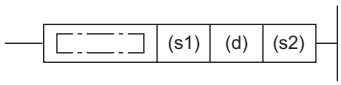
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The usage range of the device specified by (s1) or (d) exceeds the corresponding device range due to the specification of the number of separation bits specified by (s2). |
| 2821H | The (s1), (s2) devices are overlapping. |
| | The (s1), (d) devices are overlapping. |
| | The (s2), (d) devices are overlapping. |
| 3401H | The specification of the number of separation bits specified by (s2) is not set within the range of 1 to 16 bits. |
| | 0 is not set in the range between the device specified by (s2) up to the corresponding device range. |

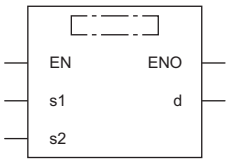
# Connecting the specified number of bits

## NUNI(P)

These instructions link each bit of the data in the device numbers specified by (s1) onwards into bit units specified by (s2), and store the connected data in the device number specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=NUNI(EN,s1,s2,d);<br>ENO:=NUNIP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

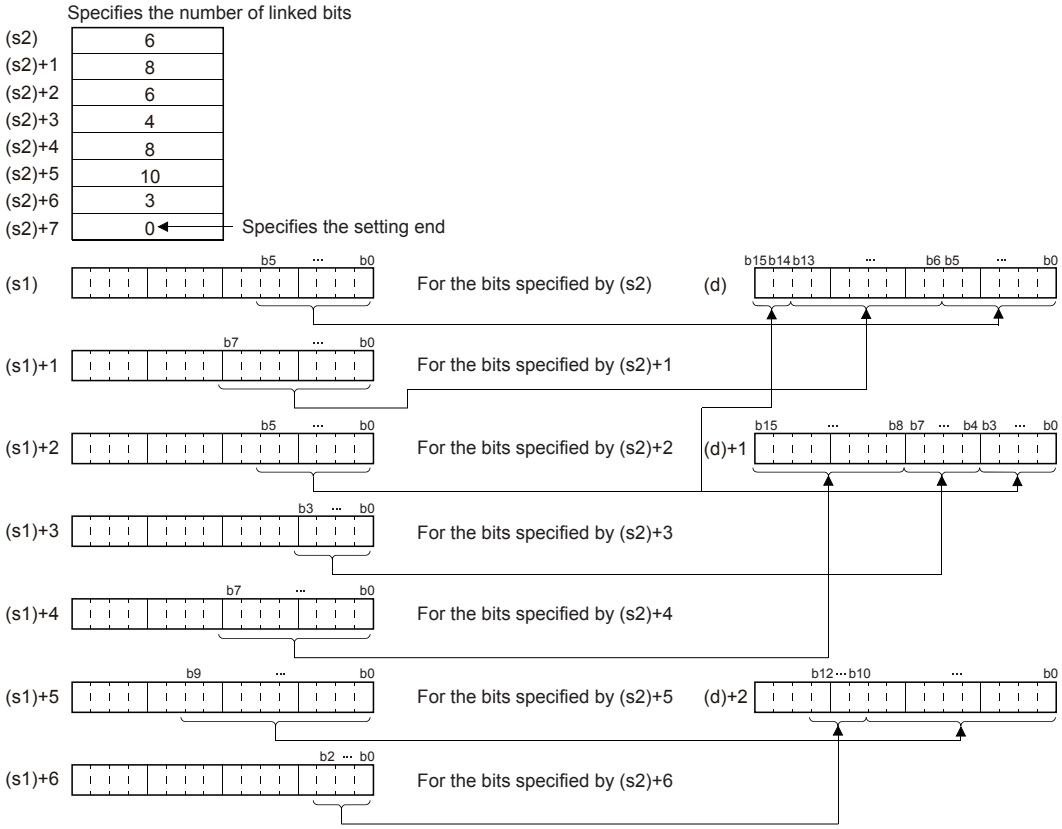
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device for storing the data to be linked | — | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the linked data | — | 16-bit signed binary | ANY16 |
| (s2) | Head device for storing the link unit size | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

## Processing details

- These instructions link each bit of the data in the device numbers specified by (s1) onwards into bit units specified by (s2), and store the linked data in the device number specified by (d).




- The number of link bits specified by (s2) can be specified within the range of 1 to 16 bits.
- The number of bits specified in devices from the device number specified by (s2) up to the device number in which "0" is stored are processed as the number of connection bits.
- If the device numbers specified by (s1), (s2), (d) are partially overlapping, an operation error occurs.

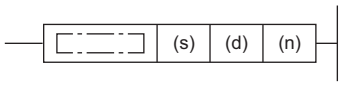
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The usage range of the device specified by (s1) or (d) exceeds the corresponding device range due to the specification of the number of link bits specified by (s2). |
| 2821H | The (s1), (s2) devices are overlapping. |
| | The (s1), (d) devices are overlapping. |
| | The (s2), (d) devices are overlapping. |
| 3401H | The specification of the number of link bits specified by (s2) is not set within the range of 1 to 16 bits. |
| | 0 is not set in the range between the device specified by (s2) up to the corresponding device range. |

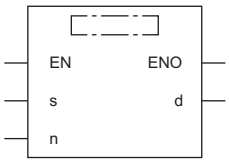
# Separating data in byte units

## WTOB(P)

These instructions separate the 16-bit binary data in the device numbers starting from the one specified by (s) onwards into (n) byte units, and store the separated data in the device number specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WTOB(EN,s,n,d);<br>ENO:=WTOBP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

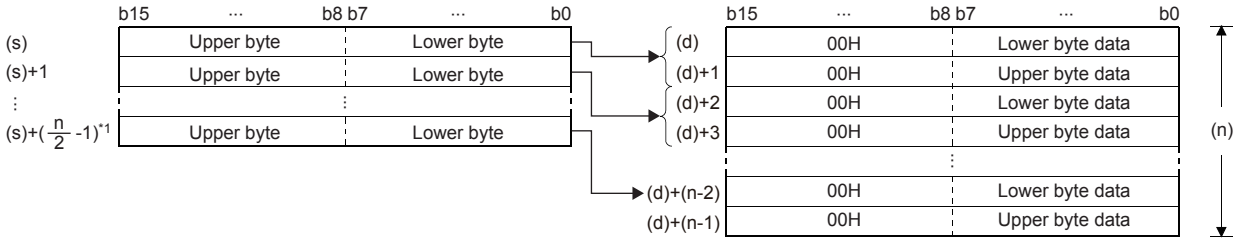
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device where the separation target data is stored | — | 16-bit signed binary | ANY16 |
| (d) | Head device for storing the result of separation in byte unit | — | 16-bit signed binary | ANY16 |
| (n) | Number of byte units | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

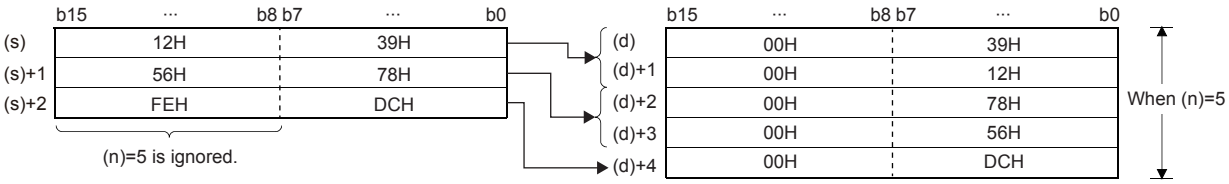
- These instructions separate the 16-bit binary data in the device numbers starting from the one specified by (s) onwards into (n) byte units, and store the separated data in the device number specified by (d) onwards.




*1 Values after the decimal point are rounded up.

**Ex.**

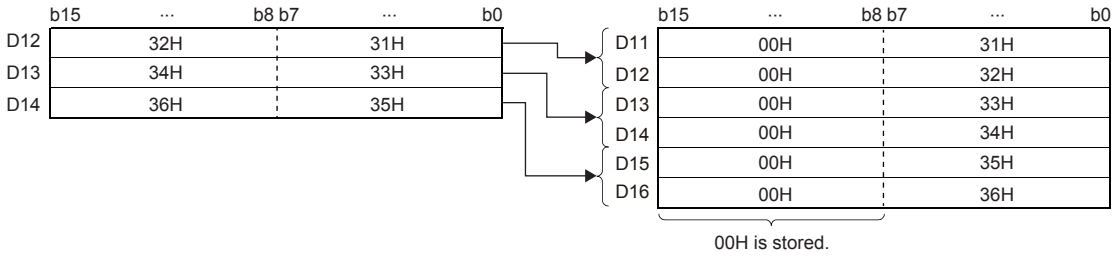
For example, when (n) is 5, data starting from (s) to the lower 8 bits of (s)+2 is stored into (d) through (d)+4.




- Setting the number of bytes by (n) automatically determines the 16-bit binary data range specified by (s) and the device range specified by (d) for storing the separated byte data.
- If (n) is 0, no processing is performed.
- In the upper byte of the devices specified by (d) to hold byte data, 00Hs are automatically stored.

**Ex.**

To store data in D12 to D14 into the lower 8 bits of D11 to D16




- Even if the device range of the data to be separated and the device range for storing the separated data overlap, the processing is performed normally.

| Device range where the data to be separated is stored | Device range for storing the separated data |
|---|---|
| (s) to (s)+($\frac{n}{2}$ -1) | (d)+0 to (d)+(n)-1 |

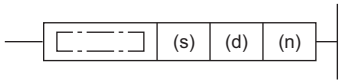
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of no. of bytes specified in (n) from the device number specified in (s) onwards exceed the corresponding device range. |
| | The range of (n) points of devices from the device number specified in (d) onwards exceed the corresponding device range. |

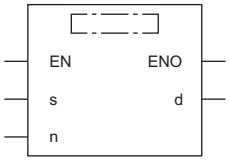
# Connecting data in byte units

## BTOW(P)

These instructions link the lower-order 8 bits of the 16-bit binary data of (n) number of bytes stored in the device numbers starting from the one specified by (s) onwards into word units, and store the linked data in the device numbers starting from the one specified by (d) onwards.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BTOW(EN,s,n,d);<br>ENO:=BTOWP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

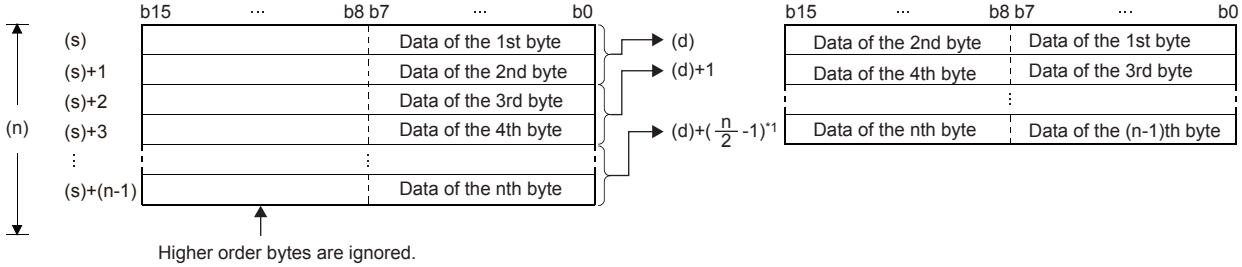
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the data to be linked in byte units | — | 16-bit signed binary | ANY16 |
| (d) | Head device storing data acquired by combination in byte units | — | 16-bit signed binary | ANY16 |
| (n) | Number of byte data to be linked | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

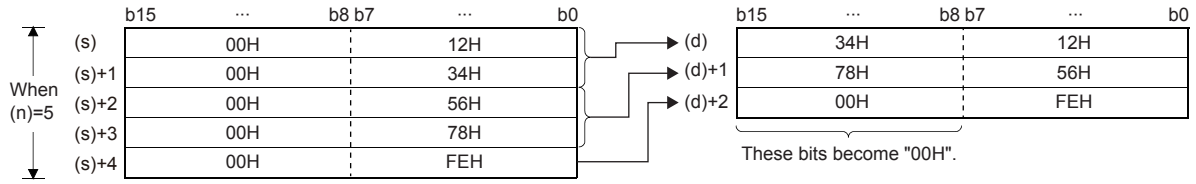
- These instructions link the lower-order 8 bits of the 16-bit binary data of (n) number of bytes stored in the device numbers starting from the one specified by (s) onwards, and store the linked data in the device numbers starting from the one specified by (d) onwards.
- The higher-order 8 bits of the data of (n) words stored in device numbers starting from the one specified by (s) are ignored. If (n) is an odd number, 0 is stored in the higher-order 8 bits of the device for storing the data of the (n)th byte.




*1 Values after the decimal point are rounded up.

**Ex.**

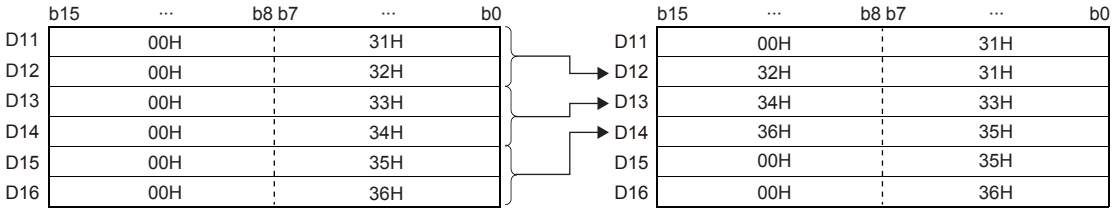
For example, when (n) is 5, lower 8 bits of data from (s) through (s+4) is stored into (d) through (d)+2.




- Setting the number of bytes by (n) automatically determines the byte data range specified by (s) and the device range specified by (d) for storing the linked data.
- If (n) is 0, no processing is performed.
- The higher-order 8 bits of the device specified by (s) for storing byte data are ignored, and only the lower-order 8 bits are applicable.

**Ex.**

To store data in lower 8 bits of D11 to D16 into D12 to D14




- Even if the device range of the data to be linked and the device range for storing the linked data overlap, the processing is performed normally.

| Device range where the data to be linked is stored | Device range for storing the linked data |
|---|---|
| (s)+0 to (s)+(n)-1 | (d) to (d)+($\frac{n}{2}$ -1) |

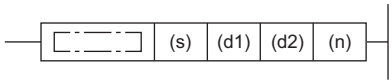
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) points of devices from the device number specified in (s) onwards exceed the corresponding device range. |
| | The range of no. of bytes specified in (n) from the device number specified in (d) onwards exceed the corresponding device range. |

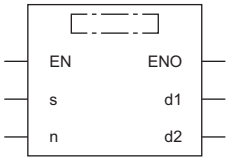
# 7.6 Digital Switch (Thumbwheel Input)

## DSW

This instruction reads the set value of digital switches. This instruction can read a set of 4 digits (n = K1) or two sets of 4 digits (n = K2).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSW(EN,s,n,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number connected to a digital switch | — | Bit | ANYBIT_ARRAY (Number of element: 4) |
| (d1) | Head device number to which the strobe signal is output | — | Bit | ANYBIT_ARRAY (Number of element: 4) |
| (d2) | Device number storing the numeric value of a digital switch | 0 to 9999 | 16-bit signed binary | ANY16 |
| (n) | Total number of 4-digit switch sets | 1, 2 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d1) | ○*2 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | — | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |

*1 Only X can be used.
*2 Only Y can be used.

### Processing details

The value of each digital switch connected to (s) is input by the time division method (in which the value is input in turn from the 1st digit by 100 ms interval output signal), and stored to (d2). A numeric value from 0 to 9999 (up to 4 digits) can be read, the first set is stored to (d2), and the second set is stored to (d2)+1.

When using one set of 4 digits (n = K1)

A 4-digit BCD digital switch connected to (s) to (s)+3 is read in turn by the strobe signal (d1) to (d1)+3, and stored in binary format to (d2).

When using two sets of 4 digits (n = K2)

A 4-digit BCD digital switch connected to (s) to (s)+3 is read in turn by the strobe signal (d1) to (d1)+3. (s) to (s3)+3 is stored in (d2) and (s)+4 to (s)+7 is stored in (d2)+1 as BIN format.

For the connection example of digital switch, refer to the following manual.

MELSEC iQ-F FX5U User's Manual (Hardware)
MELSEC iQ-F FX5UC User's Manual (Hardware)

## Precautions

- Though the contents of (d2) do not change, all of (d1) to (d1)+3 turn OFF.
- When one set of 4 digits (n = K1) are used, four devices are occupied starting from (s).
- When two sets of 4 digits (n = K2) are used, eight devices are occupied starting from (s), and two devices are occupied starting from (d2).
- When connecting a digital switch of less than 4 digits, it is not necessary to wire the strobe signal (output for digit specification) (d1) to unused digits. Because unused digits are occupied also by this instruction, however, they cannot be used for any other purpose.
- For continuously receiving digital switch values, make sure to use a transistor output type CPU module.
- Use BCD output type digital switches.
- The DSW instruction can only be executed four times in a program.

## Operation error

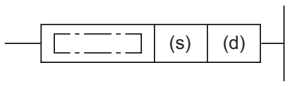
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s), (d1), (d2) exceeds the corresponding device range. |
| 3405H | (n) is other than 1 or 2. |
| | The value specified by (s) to (s)+1 and (s)+4 to (s)+7 is other than 0 to 9. |
| 1811H | The number of the DSW instructions which are used simultaneously exceeds four. |

# 7.7 Data Transfer Instructions

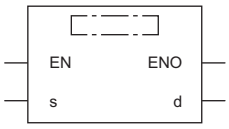
## Transferring 16-bit data

### MOV(P)

These instructions transfer the 16-bit binary data in the device specified by (s) to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MOV(EN,s,d);<br>ENO:=MOVP(EN,s,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

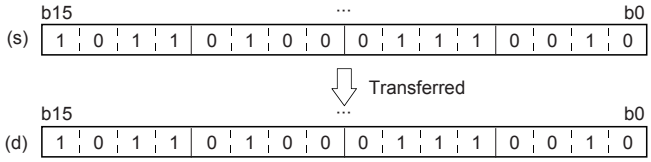
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Transfer source data or device number for storing data | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Transfer destination device number | — | 16-bit signed binary | ANY16 |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions transfer the 16-bit binary data in the device specified by (s) to the device specified by (d).



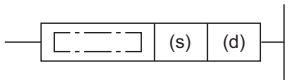

#### Operation error

There is no operation error.

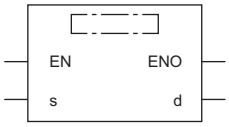
# Transferring 32-bit data

## DMOV(P)

These instructions transfer the 32-bit binary data in the device specified by (s) to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DMOV(EN,s,d);<br>ENO:=DMOVP(EN,s,d) |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

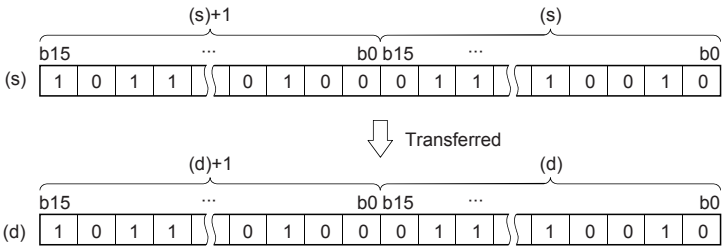
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Transfer source data or device number for storing data | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Transfer destination device number | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the device specified by (d).



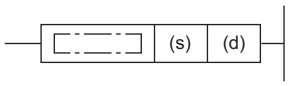

### Operation error

There is no operation error.

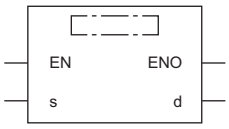
# Inverting and transferring 16-bit data

## CML(P)

These instructions invert each bit of the 16-bit binary data in the device specified by (s), and transfer the result to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=CML(EN,s,d);<br>ENO:=CMLP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

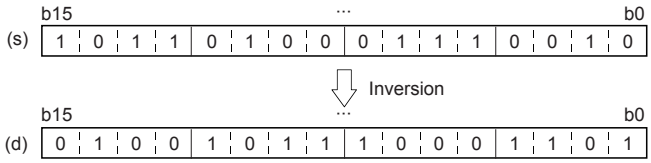
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data to be inverted or device number in which data is stored | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Device number for storing the inversion result | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions invert each bit of the 16-bit binary data in the device specified by (s), and transfer the result to the device specified by (d).



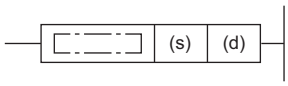

### Operation error

There is no operation error.

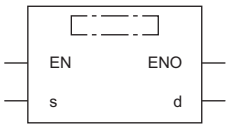
# Inverting and transferring 32-bit data

## DCML(P)

These instructions invert each bit of the 32-bit binary data in the device specified by (s), and transfer the result to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DCML(EN,s,d);<br>ENO:=DCMLP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

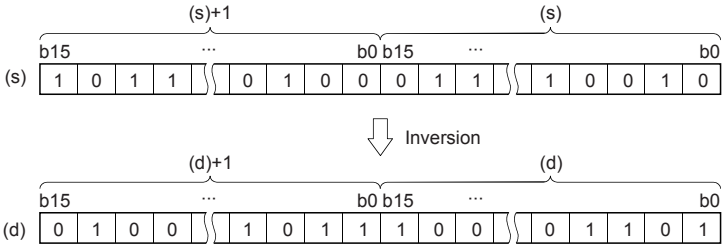
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data to be inverted or device number in which data is stored | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Device number for storing the inversion result | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions invert each bit of the 32-bit binary data in the device specified by (s), and store the result in the device specified by (d).



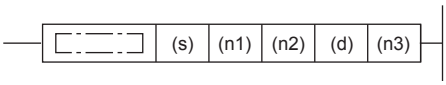

### Operation error

There is no operation error.

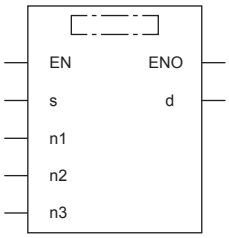
# Digit move

## SMOV(P)

These instructions distribute and compose data in units of nibble (4 bits).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SMOV(EN,s,n1,n2,n3,d);<br>ENO:=SMOVP(EN,s,n1,n2,n3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Word device number storing data whose nibbles will be moved | — | 16-bit signed binary | ANY16 |
| (n1)*1 | Head nibble position to be moved | 1 to 4 | 16-bit unsigned binary | ANY16_U |
| (n2)*1 | Number of nibbles to be moved | 1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d) | Word device number storing data whose nibbles are moved | — | 16-bit signed binary | ANY16 |
| (n3)*1 | Head digit position of movement destination | 1 to 4 | 16-bit unsigned binary | ANY16_U |

*1 Set so that n2≤n1, n2≤n3.

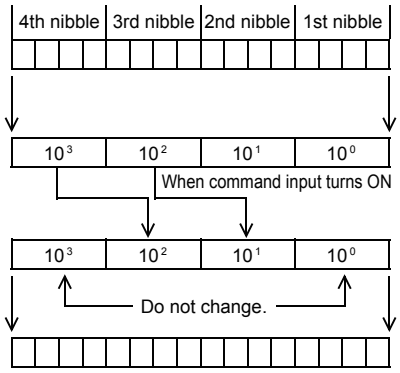
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

These instructions distribute and compose data in units of nibble (4 bits). The contents of the transfer source (s) and transfer destination (d) are converted into 4-digit BCD (0000 to 9999). (n2) nibbles starting from the (n1)th nibble are transferred to the transfer destination (d) starting from the (n3)th nibble, converted into binary, and then stored to the transfer destination (d).

- While the command input is OFF, the transfer destination (d) does not change.
- When the command input turns ON, only the specified digits in the transfer destination (d) are changed. The transfer source (s) and unspecified digits in the transfer destination (d) do not change.




In the case of "n1 = 4, n2 = 2, n3 = 3".

(s) (16-bit binary data)

↓ Data is automatically converted(1)

(s)' (4-digit BCD data)

↓ Nibbles are moved(2)

(d)' (4-digit BCD data)

↓ Data is automatically converted(3)

(d) (16-bit binary data)

(1): (s) is converted from binary to → BCD data.

(2): (n2) digits starting from the (n1)th digit are transferred (combined) to (d)' starting from the (n3)th digit. The first and fourth digits of (d)' are not affected even if data is transferred from (s)'.

(3): The combined data (BCD) is converted into binary, and stored to (d).

### ■Extension function

When SM8168 is set to ON first and then SMOV instruction is executed, conversion from binary to BCD is not executed. Data is moved in units of 4 bits.

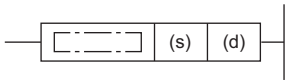
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | Any one of (n1), (n2), (n3) is 0. |
| | Either (s) or (d) is other than 0 to 9999 when SM8168 is OFF. |
| | Either (n1) or (n3) is larger than 4. |
| | (n2) is larger than (n1) or (n3). |

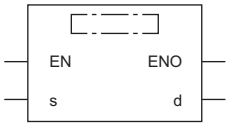
# Inverting and transferring 1-bit data

## CMLB(P)

These instructions invert the bit data in the device specified by (s), and transfer the result into the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=CMLB(EN,s,d);<br>ENO:=CMLBP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Transfer-source data | — | Bit | ANY_BOOL |
| (d) | Transfer-destination data | — | Bit | ANY_BOOL |

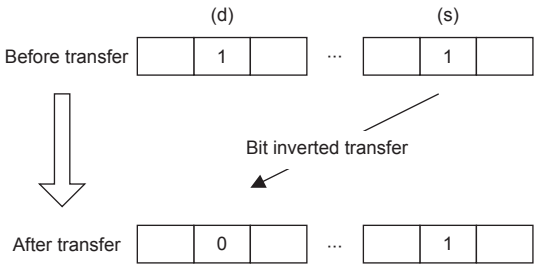
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | — |
| (d) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

These instructions invert the bit data in the device specified by (s), and transfer the result in the device specified by (d).



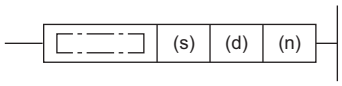

### Operation error

There is no operation error.

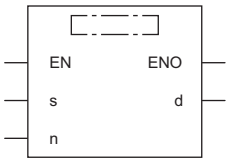
# Transferring 16-bit block data (65535 points maximum)

## BMOV(P)

These instructions block transfer the 16-bit binary data of (n) number of devices starting from the one specified by (s) to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BMOV(EN,s,n,d);<br>ENO:=BMOVP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

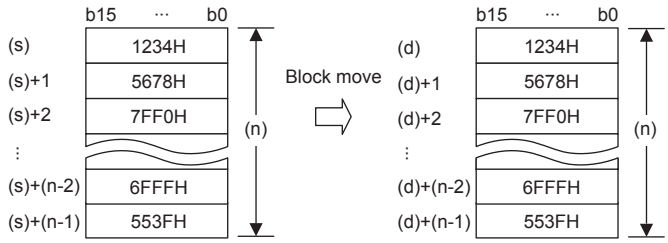
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the data to be transferred | — | 16-bit signed binary/<br>32-bit signed binary | ANY16 |
| (d) | Head number of the transfer-destination device | — | 16-bit signed binary/<br>32-bit signed binary | ANY16 |
| (n) | Number of transfers | 1 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- These instructions block transfer the 16-bit binary data of (n) number of devices starting from the one specified by (s) to the device specified by (d).

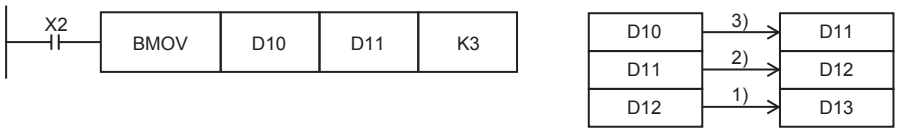



- If the device number range is exceeded, data is transferred within the possible range.
- Data can be transferred even when the device range of the transfer-source device and transfer-destination device is overlapping. To transfer data to a device having a smaller device number, transfer from (s), and to transfer data to a device having a larger device number, transfer from (s)+(n)-1.

**Ex.**

When transferring data to a device having a smaller device number


When transferring data to a device having a larger device number




## Precautions

- To perform nibble specification of bit device for both (s) and (d), be sure to set the same number of nibbles for (s) and (d).
- To use a module access device for (s) and (d), specify either (s) or (d).

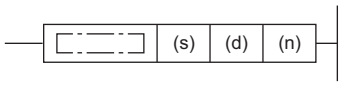
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The number of nibbles of the nibble specification of bit device of (s) and (d) is different. |
| 3420H | A module access device is specified for both (s) and (d). |

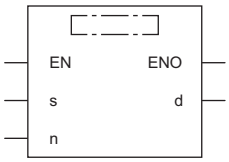
# Transferring identical 16-bit block data (65535 points maximum)

## FMOV(P)

These instructions transfer (n) point(s) of data identical to the 16-bit binary data in the device specified by (s) to the devices specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FMOV(EN,s,n,d);<br>ENO:=FMOVP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

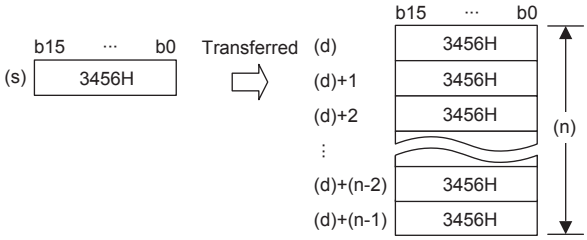
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data to be transferred or the head device for storing the data to be transferred | -32748 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device of the transfer-destination | — | 16-bit signed binary | ANY16 |
| (n) | Number of transfers | 1 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions transfer (n) point(s) of data identical to the 16-bit binary data in the device specified by (s) to the device specified by (d).




- If the number of points specified by (n) exceeds the device number range, data is transferred within the possible range.
- When a constant (K) is specified as the transfer source (s), it is automatically converted into binary.

### Precautions

When the value specified in (n) is 0, an operation error does not occur, but no processing is performed,

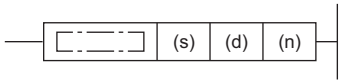
### Operation error

There is no operation error.

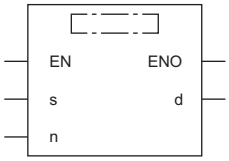
# Transferring identical 32-bit block data (65535 points maximum)

## DFMOV(P)

These instructions transfer (n) point(s) of data identical to the 32-bit binary data in the device specified by (s) to the devices specified by (d).
(65535 points maximum)

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DFMOV(EN,s,n,d);<br>ENO:=DFMOVP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

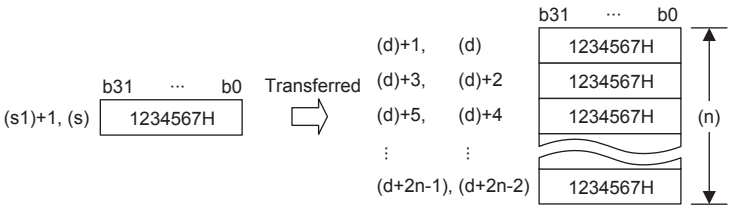
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data to be transferred or the head device for storing the data to be transferred | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device of the transfer-destination | — | 32-bit signed binary | ANY32 |
| (n) | Number of transfers | 1 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions transfer (n) point(s) of data identical to the 32-bit binary data in the device specified by (s) to the device specified by (d).




- If the number of points specified by (n) exceeds the device number range, data is transferred within the possible range.
- When a constant (K) is specified as the transfer source (s), it is automatically converted into binary.

### Precautions

When the value specified in (n) is 0, an operation error does not occur, but no processing is performed.

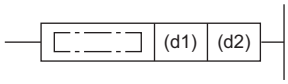
### Operation error

There is no operation error.

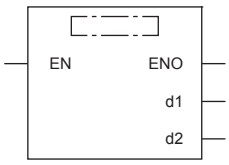
# Exchanging 16-bit data

## XCH(P)

These instructions exchange 16-bit binary data of (d1) and (d2).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=XCH(EN,d1,d2);<br>ENO:=XCHP(EN,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

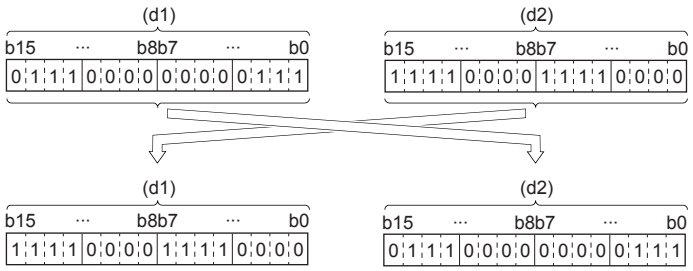
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d1) | Head device for storing the data to be exchanged | — | 16-bit signed binary | ANY16 |
| (d2) | Head device for storing the data to be exchanged | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

These instructions exchange 16-bit binary data of (d1) and (d2).



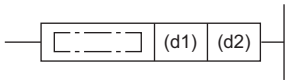

### Operation error

There is no operation error.

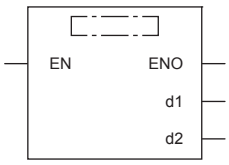
# Exchanging 32-bit data

## DXCH(P)

These instructions exchange 32-bit binary data of (d1) and (d2).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DXCH(EN,d1,d2);<br>ENO:=DXCHP(EN,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

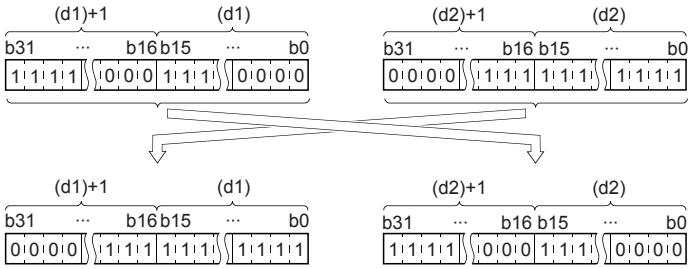
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d1) | Head device for storing the data to be exchanged | — | 32-bit signed binary | ANY32 |
| (d2) | Head device for storing the data to be exchanged | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

These instructions exchange 32-bit binary data of (d1), (d1)+1 and (d2), (d2)+1



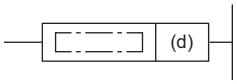

### Operation error

There is no operation error.

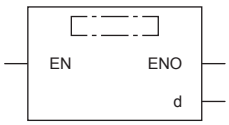
# Exchanging the upper and lower bytes of 16-bit data

## SWAP(P)

These instructions swap the value of 8 bits of the upper and lower bytes of the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SWAP(EN,d);<br>ENO:=SWAPP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

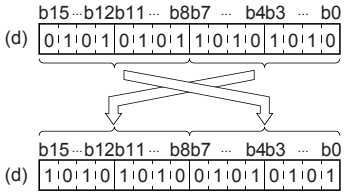
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for storing the data to be swapped | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

These instructions swap the value of 8 bits of the upper and lower bytes of the device specified by (d).




### Precautions

If a continuous operation type instruction is used, swap is done in each operation cycle.

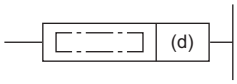
### Operation error

There is no operation error.

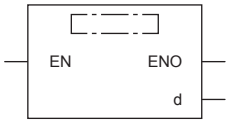
# Exchanging the upper and lower bytes of 32-bit data

## DSWAP(P)

These instructions swap the value of 8 bits of the upper and lower bytes of the word devices specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSWAP(EN,d);<br>ENO:=DSWAPP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

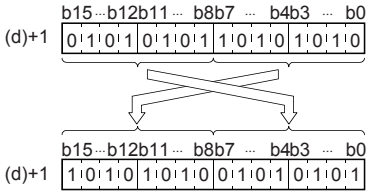
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device for storing the data to be swapped | — | 32-bit signed binary | ANY32 |

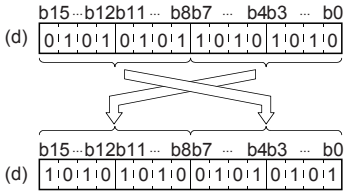
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

These instructions swap the value of each of the upper and lower 8 bits of the device specified by (d) and (d)+1.




### Precautions

If a continuous operation type instruction is used, swap is done in each operation cycle.

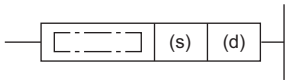
### Operation error

There is no operation error.

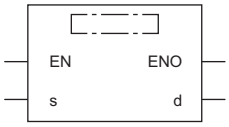
# Transferring 1-bit data

## MOVB(P)

These instructions store bit data specified by (s) to (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MOVB(EN,s,d);<br>ENO:=MOVBP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the transfer-source data | — | Bit | ANY_BOOL |
| (d) | Head device for storing the transfer-destination data | — | Bit | ANY_BOOL |

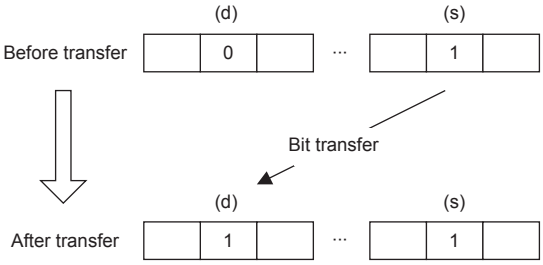
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | — |
| (d) | ○ | ○ | ○ | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions transfer bit data specified by (s) to (d).



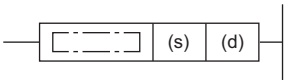

### Operation error

There is no operation error.

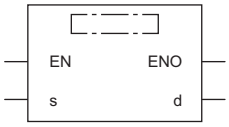
# Transferring octal bits (16-bit data)

## PRUN(P)

These instructions handle the device number of (s) and (d) with nibble specification as octal numbers, and transfer data.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PRUN(EN,s,d);<br>ENO:=PRUNP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Nibble specification*1 | — | 16-bit signed binary | ANY16 |
| (d) | Device number of transfer destination*1 | — | 16-bit signed binary | ANY16 |

*1 Make sure that the least significant digit of a specified device number is "0".

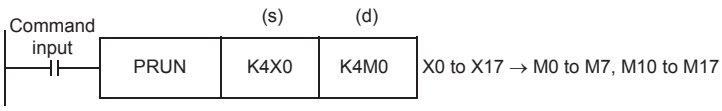
#### ■Applicable devices

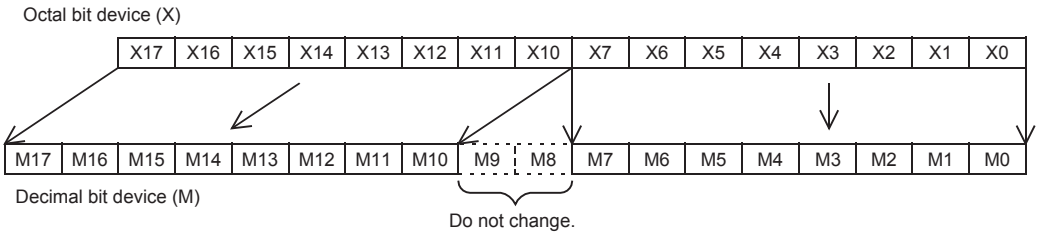
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | — | — | — | — | — | ○ | — | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | ○ | — | — | — | — |

*1 B, SB cannot be used.

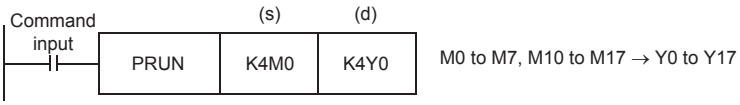
### Processing details

- Octal bit device → Decimal bit device

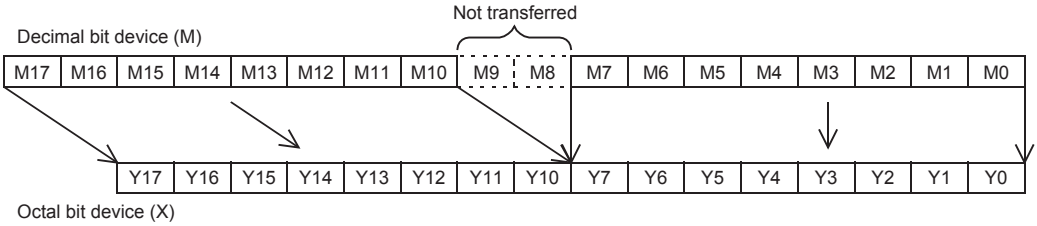


• Decimal bit device → Octal bit device




M0 to M7, M10 to M17 → Y0 to Y17



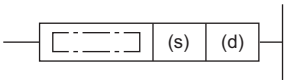

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The devices specified by (s) and (d) exceed the range of the corresponding device. |

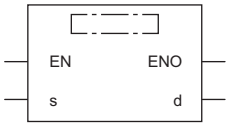
# Transferring octal bits (32-bit data)

## DPRUN(P)

These instructions handle the device number of (s) and (d) with nibble specification as octal numbers, and transfer data.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPRUN(EN,s,d);<br>ENO:=DPRUNP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Nibble specification*1 | — | 32-bit signed binary | ANY32 |
| (d) | Device number of transfer destination*1 | — | 32-bit signed binary | ANY32 |

*1 Make sure that the least significant digit of a specified device number is "0".

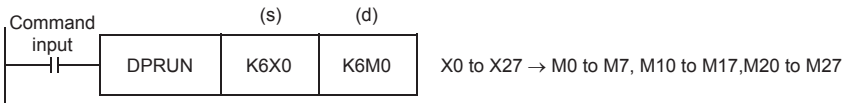
#### ■Applicable devices

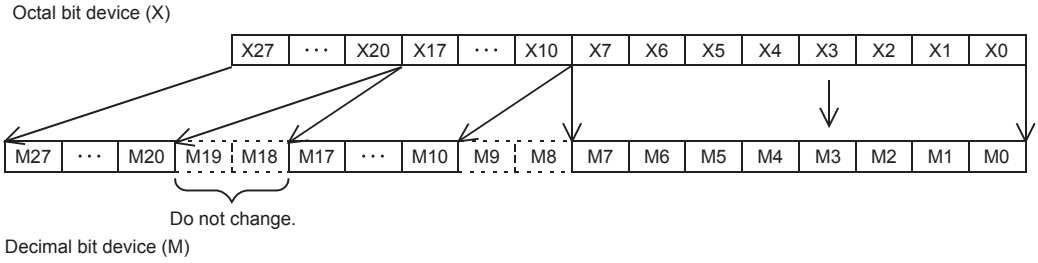
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | — | — | — | — | — | ○ | — | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | ○ | — | — | — | — |

*1 B, SB cannot be used.

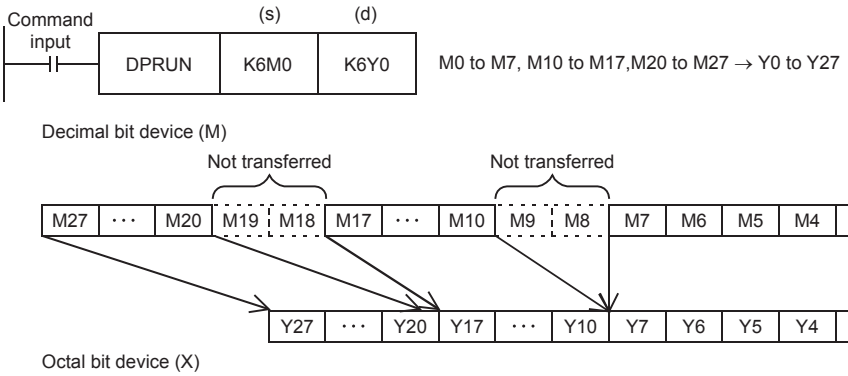
### Processing details

- Octal bit device → Decimal bit device



• Decimal bit device → Octal bit device



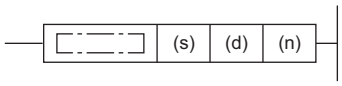

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The devices specified by (s) and (d) exceed the range of the corresponding device. |

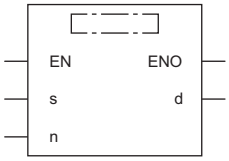
# Transferring n-bit data

## BLKMOVB(P)

These instructions block transfer the bit data of (n) point(s) from the device specified by (s) to the bit data of (n) point(s) from (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BLKMOVB(EN,s,n,d);<br>ENO:=BLKMOVBP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device for storing the transfer-source bit data | — | Bit | ANY_BOOL |
| (d) | Head device for storing the transfer-destination bit data | — | Bit | ANY_BOOL |
| (n) | Number of transfers | 0 to 65535 | 16-bit unsigned binary | ANY16 |

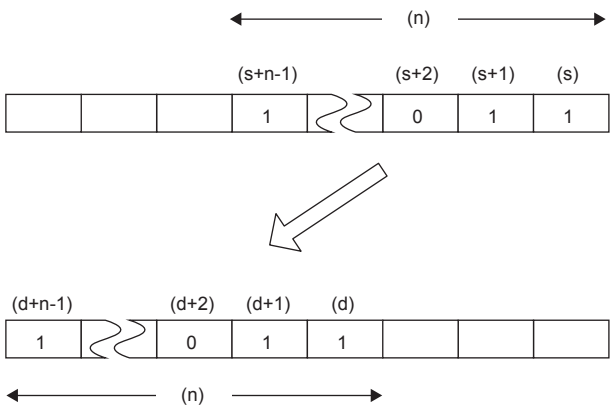
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions block transfer the bit data of (n) point(s) from the device specified by (s) to the bit data of (n) point(s) from the device specified by (d).
- Data can be transferred even when the device range of the transfer-source device and transfer-destination device is overlapping.



## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of (n) point(s) of data starting from the device specified by (s) and (d) exceed the corresponding device range. |

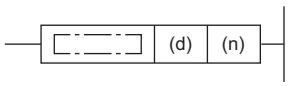
# 8 APPLICATION INSTRUCTION

## 8.1 Rotation Instruction

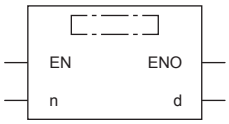
### Rotating 16-bit data to the right

#### ROR(P), RCR(P)

- ROR(P): These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s) (not including the carry flag).
- RCR(P): These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s) (including the carry flag).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=RORP(EN,n,d);<br>ENO:=RCR(EN,n,d);<br>ENO:=RCRP(EN,n,d); |

**FBD/LD*1**




*1 The ROR instruction is not supported by the ST language and the FBD/LD language. Use ROR of the standard function.
☞ Page 892 ROR(_E)

##### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the rotation target data is stored | — | 16-bit signed binary | ANY16 |
| (n) | Number of rotations | 0 to 15 | 16-bit unsigned binary | ANY16 |

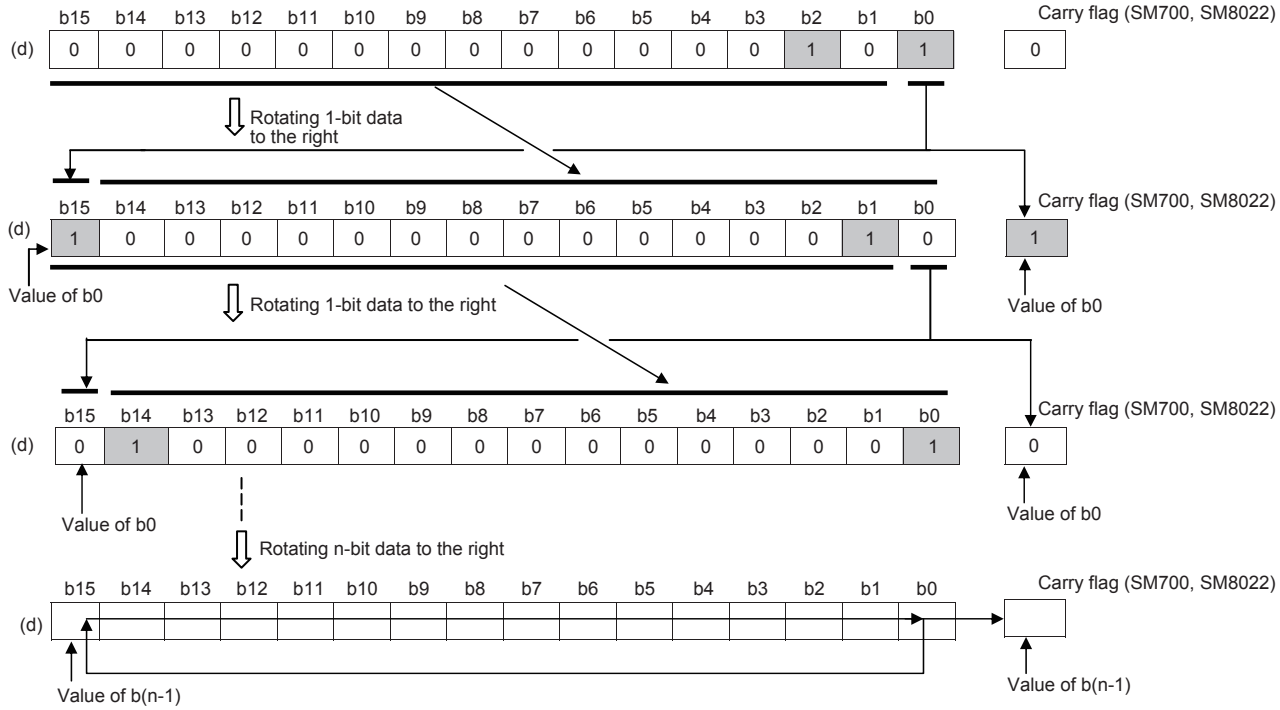
**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

### ■ROR(P)

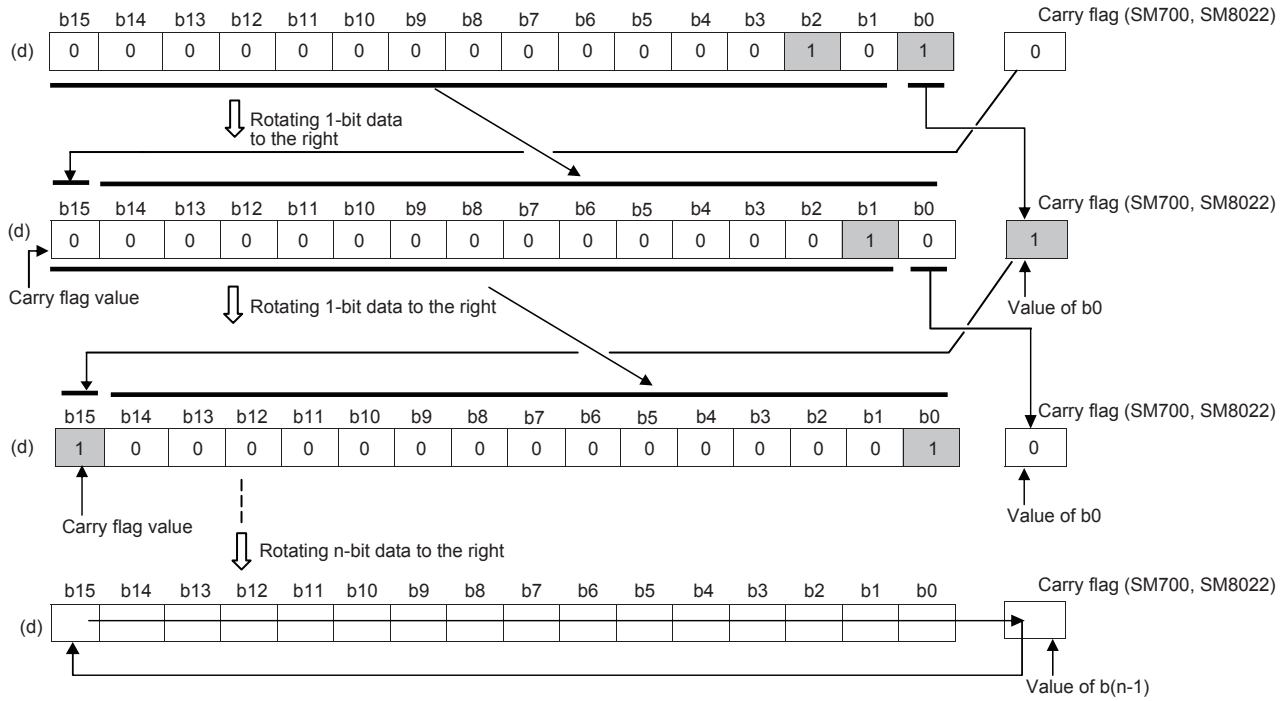
- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s) (not including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the right within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

## ■RCR(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s) (including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

### Precautions

- Do not set a negative value to the number of bits to be rotated (n).
- In the case of continuous operation type instructions (ROR and RCR), note that shift and rotation are executed in every scan time (operation cycle).

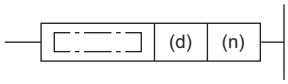
### Operation error

There is no operation error.

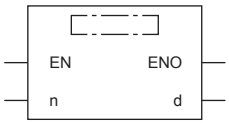
# Rotating 16-bit data to the left

## ROL(P), RCL(P)

- ROL(P): These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s) (not including the carry flag).
- RCL(P): These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s) (including the carry flag).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=ROLP(EN,n,d);<br>ENO:=RCL(EN,n,d);<br>ENO:=RCLP(EN,n,d); |

**FBD/LD*1**




*1 The ROL instruction is not supported by the ST language and the FBD/LD language. Use ROL of the standard function.
☞ Page 890 ROL(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the rotation target data is stored | — | 16-bit signed binary | ANY16 |
| (n) | Number of rotations | 0 to 15 | 16-bit unsigned binary | ANY16 |

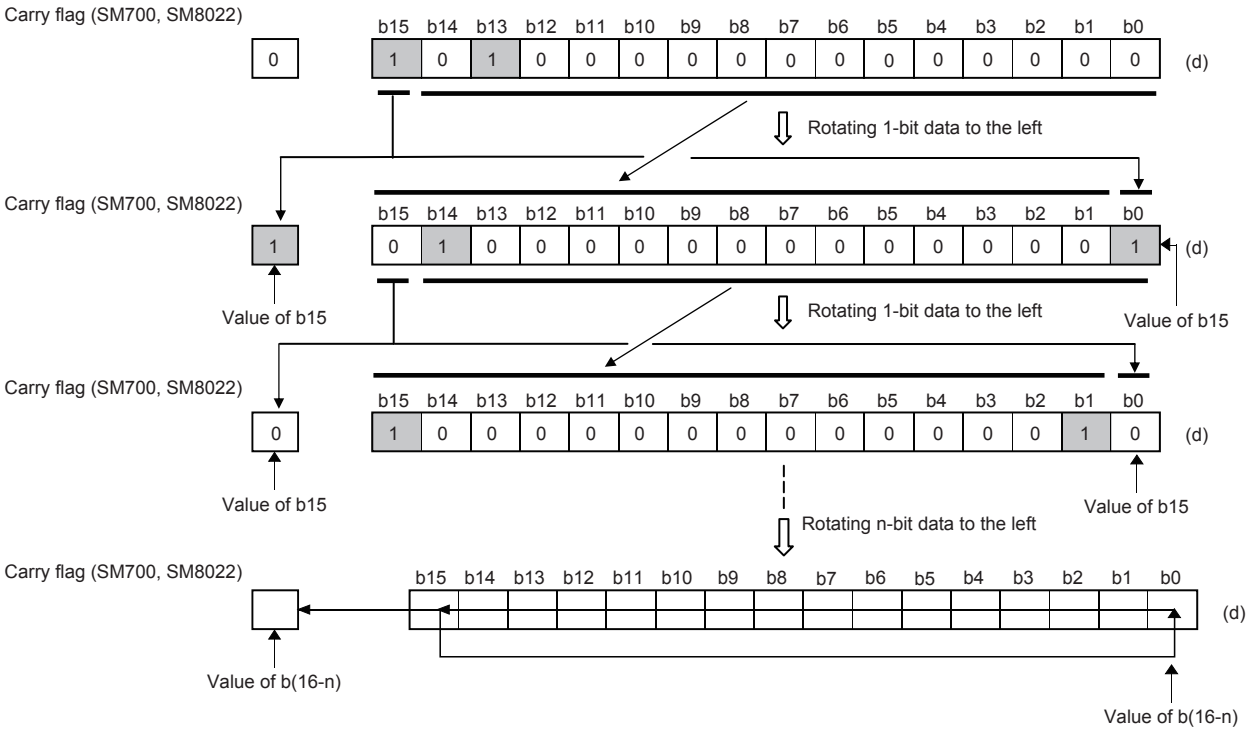
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

### ■ROL(P)

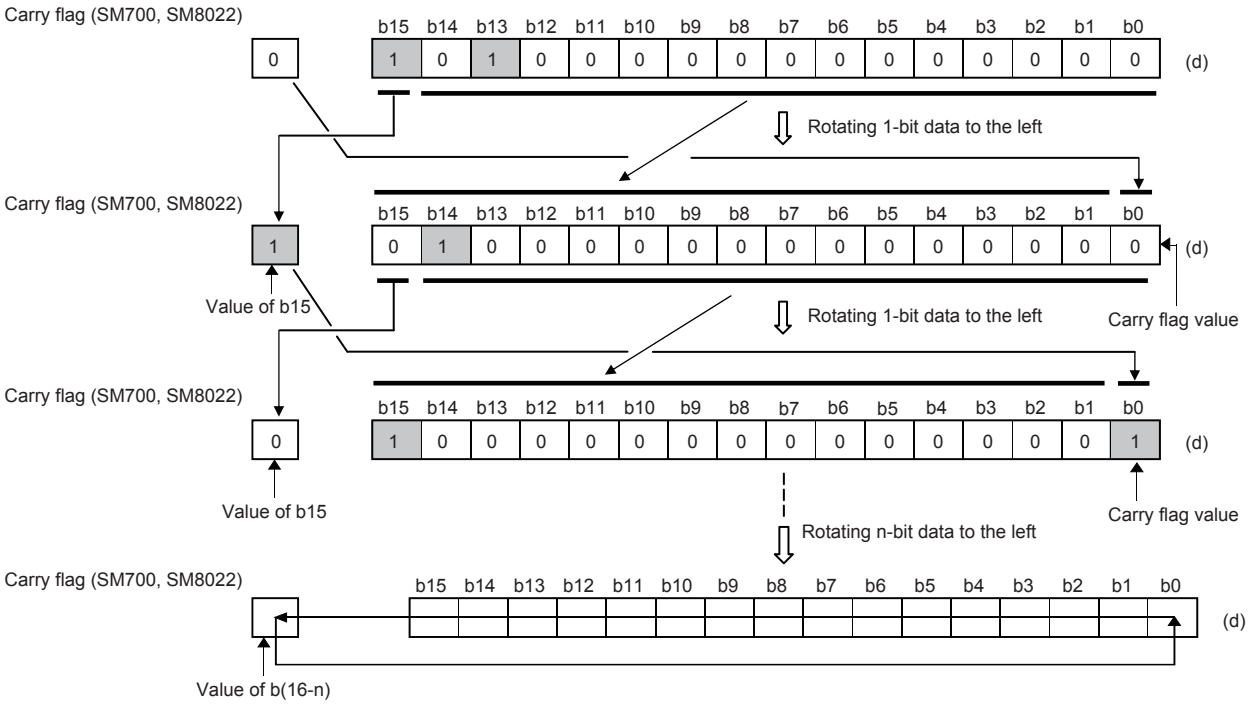
- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s) (not including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the left within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

## ■RCL(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s) (including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the left within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

### Precautions

- Do not set a negative value to the number of bits to be rotated (n).
- In the case of continuous operation type instructions (ROL and RCL), note that shift and rotation are executed in every scan time (operation cycle).

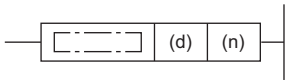
### Operation error

There is no operation error.

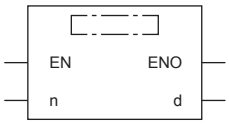
# Rotating 32-bit data to the right

## DROR(P), DRCR(P)

- DROR(P): These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s) (not including the carry flag).
- DRCR(P): These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s) (including the carry flag).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=DRORP(EN,n,d);<br>ENO:=DRCR(EN,n,d);<br>ENO:=DRCRP(EN,n,d); |

**FBD/LD*1**




*1 The DROR instruction is not supported by the ST language and the FBD/LD language. Use ROR of the standard function.
☞ Page 892 ROR(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the rotation target data is stored | — | 32-bit signed binary | ANY32 |
| (n) | Number of rotations | 0 to 31 | 16-bit unsigned binary | ANY16 |

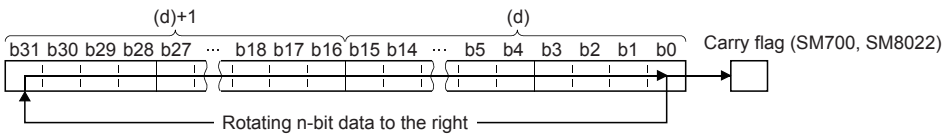
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

#### ■DROR(P)

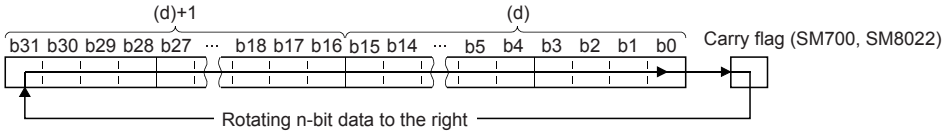
- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s) (not including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the right within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

## ■DRCR(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s) (including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the right within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

### Precautions

- Do not set a negative value to the number of bits to be rotated (n).
- In the case of continuous operation type instructions (DROR and DRCR), note that shift and rotation are executed in every scan time (operation cycle).

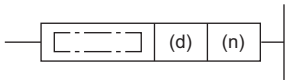
### Operation error

There is no operation error.

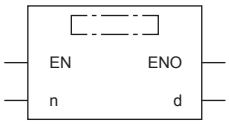
# Rotating 32-bit data to the left

## DROL(P), DRCL(P)

- DROL(P): These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s) (not including the carry flag).
- DRCL(P): These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s) (including the carry flag).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=DROLP(EN,n,d);<br>ENO:=DRCL(EN,n,d);<br>ENO:=DRCLP(EN,n,d); |

**FBD/LD*1**




*1 The DROL instruction is not supported by the ST language and the FBD/LD language. Use ROL of the standard function.
☞ Page 890 ROL(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the rotation target data is stored | — | 32-bit signed binary | ANY32 |
| (n) | Number of rotations | 0 to 31 | 16-bit unsigned binary | ANY16 |

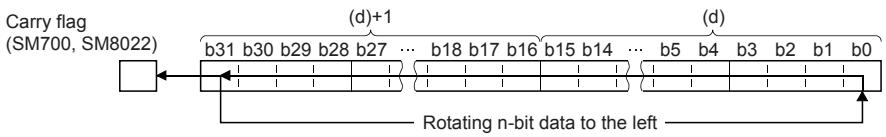
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

#### ■DROL(P)

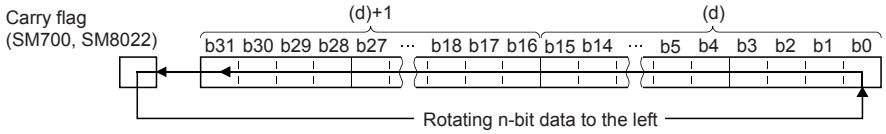
- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s) (not including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the left within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

## ■DRCL(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s) (including the carry flag). The carry flag is on or off depending on the status prior to the execution of the instruction.




- When (d) is a bit device, bits are rotated to the left within the device range specified by nibble specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

### Precautions

- Do not set a negative value to the number of bits to be rotated (n).
- In the case of continuous operation type instructions (DROL and DRCL), note that shift and rotation are executed in every scan time (operation cycle).

### Operation error

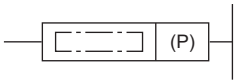
There is no operation error.

# 8.2 Program Branch Instruction

## Pointer branch

### CJ(P)

These instructions execute the program specified by the pointer number within the same program file when the jump command is on.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (P) | Pointer number of the jump destination | — | Device name | POINTER |

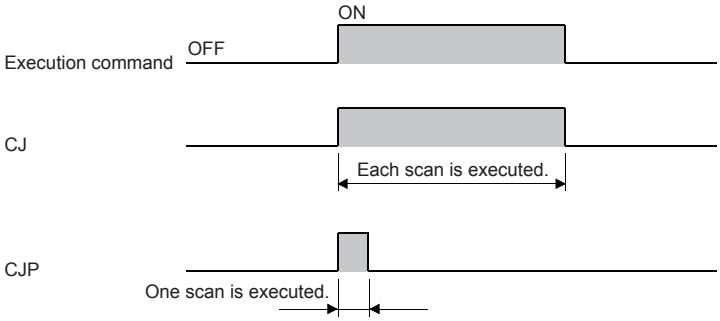
##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (P) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |

#### Processing details

##### ■CJ(P)

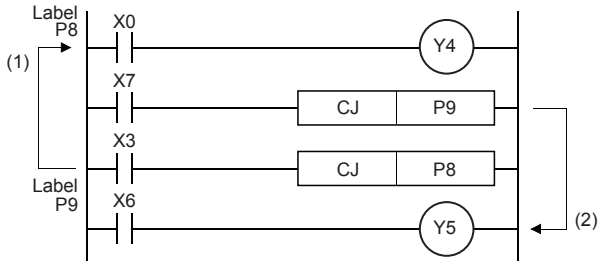
- These instructions execute the program specified by the pointer number when the execution command is on.
- When the execution command is off, the program in the next step is executed.




#### Precautions

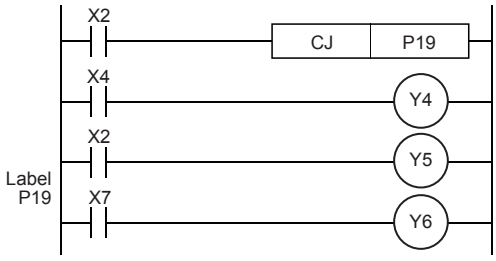
- If the timer with its coil on is skipped by these instructions, time cannot be measured correctly.
- If the OUT instruction is skipped by these instructions, the scan time will be shortened.
- If these instructions specify and jump to a later step, the scan time will be shortened.

- These instructions can specify and jump from the current step to a smaller step number. In this case, consider a method to exit a loop so that the watchdog timer does not time out.



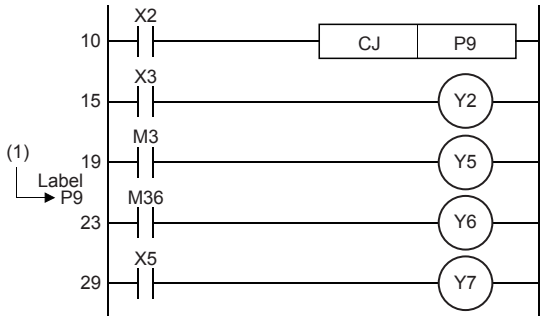

(1) While X3 is on, the loop is repeated.
(2) To exit the loop, turn on X7.

- The value in the device skipped with these instructions remains the same.



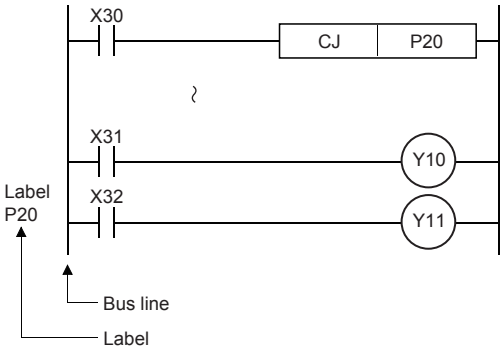

When X2 turns on, the program jumps to the label, P19.
Y4 and Y5 remain the same even if X2 and X4 turn on/off during the execution of the CJ instruction.

- A label (P□) occupies two steps.

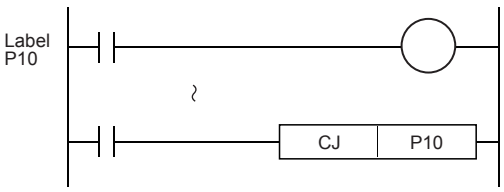



(1) A label occupies two steps.

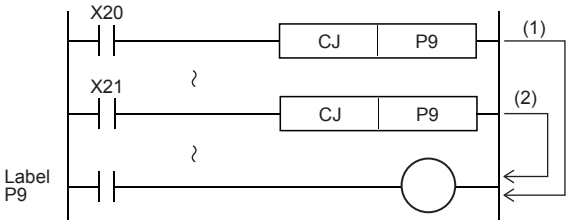
- Only the pointer numbers within the same program file can be specified.
- During skip operation, if the program jumps to the pointer number within the skip range, the programs of the jump destination pointer number and later are executed.
- The figure below shows programming of a label. When creating a circuit program, move the cursor to the left side of the bus line in the ladder diagram, and input a label (P) at the head of the circuit block.




- A label can be programmed in a smaller number step than CJ instruction. However, note that a watchdog timer error occurs when the scan time exceeds 200 ms (default setting).

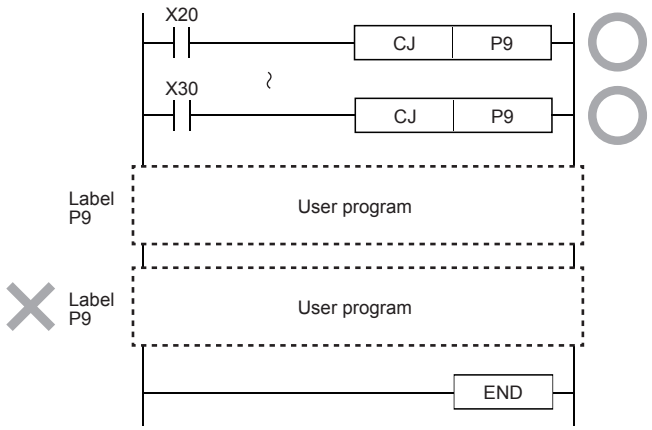


- When the pointer number in operands is same and there is one label, the following operation is caused:

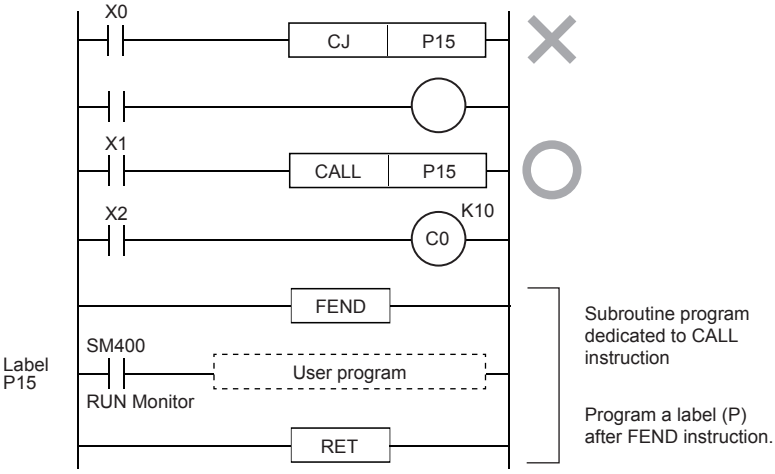



(1) When X20 turns ON, the program execution jumps from CJ instruction corresponding to X20 to the label P9.
(2) When X20 turns OFF and X21 turns ON, the program execution jumps from CJ instruction corresponding to X21 to the label P9.

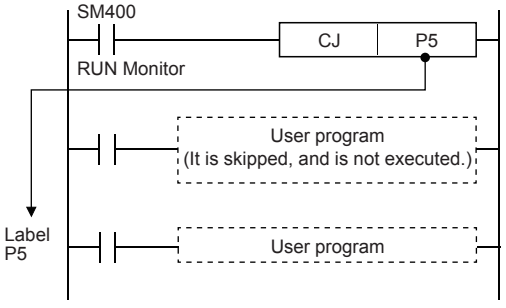
- When a label number (including labels for CALL instructions described later) is used two or more times, an error is caused.




- No label can be shared by CALL instruction and CJ instruction.




- Because SM400/SM8000 is normally ON while a PLC is operating, unconditional jump is applied when SM400 is used as shown in the following example:



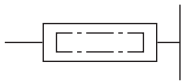

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3380H | A pointer number which is not used as a label in the same program file is specified. |

# Jump to END

## GOEND

This instruction moves the program execution to the FEND or END instruction in the same program file.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Processing details

- This instruction moves the program execution to the FEND or END instruction in the same program file.

### Precautions

- When a GOEND instruction is executed by invalid jump during interrupt program execation, it becomes the same operation as the IRET instruction.

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3340H | After the FOR instruction is executed, the GOEND instruction is executed before the NEXT instruction is executed. |
| 3381H | After the CALL(P) or XCALL instruction is executed, the GOEND instruction is executed before the RET instruction is executed. |

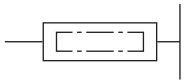
# 8.3 Program Execution Control Instruction

## Disabling/enabling interrupt programs

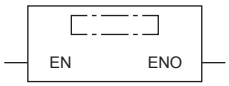
### DI, EI

Interrupts are usually disabled in CPU module. These instructions enable interrupts in CPU module (EI instruction) or disable interrupts again (DI instruction).

- DI: Disables the execution of the interrupt program.
- EI: Releases the execution disabled state of interrupt programs.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DI(EN);<br>ENO:=EI(EN); |

**FBD/LD**



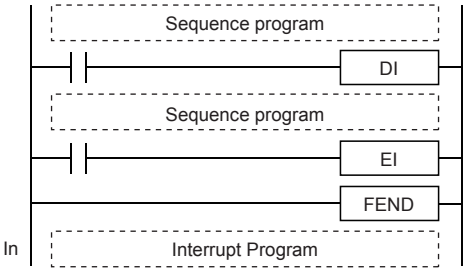

#### Processing details

**■DI**

- This instruction disables the execution of the interrupt program until the EI instruction is executed, even if the interrupt cause occurs.
- When the power is turned on or the CPU module is reset, the state in which the DI instruction is executed is applied.
- For the operation of the DI instruction (DI instruction without an argument) when using the interrupt disable instruction with a specified priority or lower (DI instruction with an argument), refer to ☞ Page 368 Disabling the interrupt program with specified priority or lower.

**■EI**

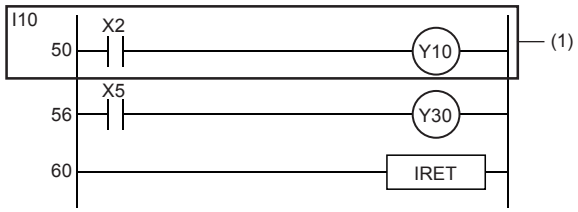
- This instruction releases the execution disabled state of interrupt programs when the DI instruction is executed, and enables the execution of the interrupt program with the interrupt pointer number enabled by the IMASK instruction.
- For the operation of the EI instruction when using the interrupt disable instruction with a specified priority or lower (DI instruction with an argument), refer to ☞ Page 368 Disabling the interrupt program with specified priority or lower.




Even though an interrupt occurs between the DI and EI instructions, the execution of the interrupt is held until the processing between the instructions ends.

**Point**

- An interrupt pointer occupies two steps. (In (1) below, I10 is the step 50, X2 is the step 52, and Y10 is the step 54.)




- If the master control contains the EI or DI instruction, such an instruction is executed regardless of the execution of the MC instruction.

## Precautions

Interrupts (requests) that are generated after the DI instruction execution, are processed after the EI instruction is executed.

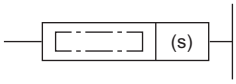
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3362H | Nesting of the DI instruction exceeds 16 levels. |

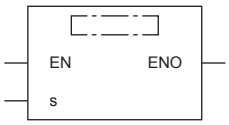
# Disabling the interrupt program with specified priority or lower

## DI

This instruction disables the execution of the interrupt program with a priority specified by (s) or lower until the EI instruction is executed, even if the interrupt cause occurs.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DI_1(EN,s); |

**FBD/LD**




("DI_1" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Priority for disabling interrupts | 1 to 3 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

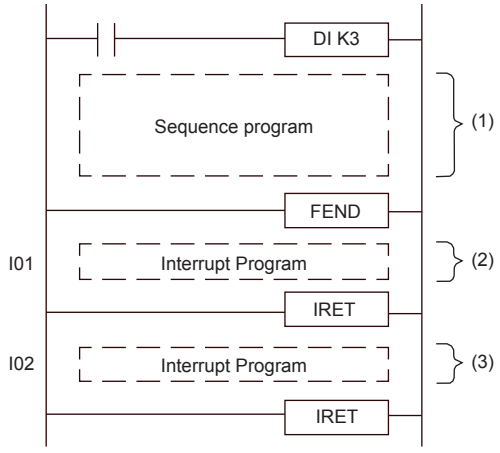
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction disables the execution of the interrupt program of the interrupt pointer number with an interrupt priority specified by (s) or lower.

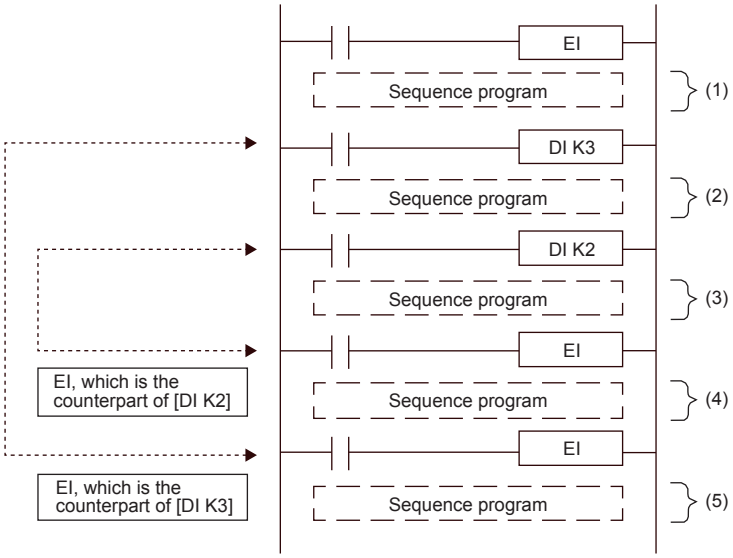
Interrupt priority setting

| I No. | Priority |
|---|---|
| I01 | 2 |
| I02 | 3 |



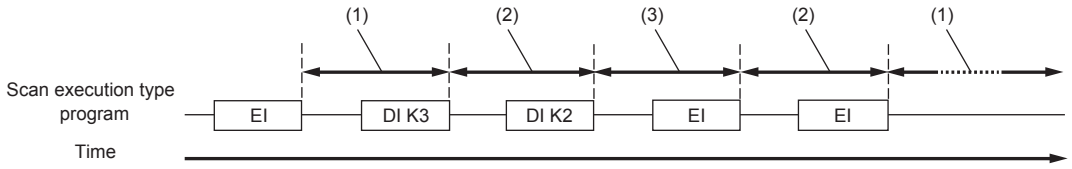

(1) Interrupt-disabled section for the priority 3 or lower (Interrupt-enabled section for the priority 2 or higher)
(2) Can be executed because of the priority 2.
(3) Cannot be executed because of the priority 3.

- By executing the EI instruction, the interrupt with the priority disabled by the counterpart DI instruction is enabled. However, when interrupts are disabled only with the DI instruction without an argument, interrupts with all the priorities are enabled by executing the EI instruction once.



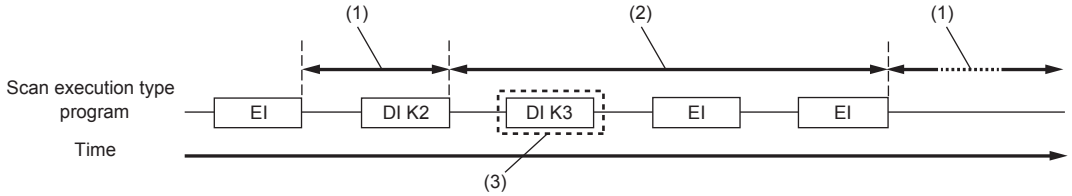

(1) Interrupt-enabled section for all priority
(2) Interrupt-disabled section for the priority 3 or lower (Interrupt-enabled section for the priority 2 or higher)
(3) Interrupt-disabled section for the priority 2 or lower (Interrupt-enabled section for the priority 1 or higher)
(4) Interrupt-disabled section for the priority 3 or lower (Interrupt-enabled section for the priority 2 or higher)
(5) Interrupt-enabled section for all priority

- Interrupts (requests) that are generated after the DI instruction are processed after the EI instruction is executed.
- When multiple DI instructions are executed and the argument has a priority higher than the currently disabled priority, interrupts with a priority lower than that of the argument are disabled.
- When multiple DI instructions are executed and the argument has a priority lower than the currently disabled priority, the interrupt disabled state is not changed.
- The DI instruction can be nested in up to 16 levels.
- The interrupt priority of the interrupt pointer can be set with parameters. (MELSEC iQ-F FX5 User's Manual (Application))
- The interrupt-disabled priority can be checked with SD758 (interrupt-disabling priority setting value).
- The following shows the interrupt-disabled section when the DI or EI instruction is executed.
  - When multiple DI instructions are executed (when interrupts with a priority higher than the currently disabled priority are specified and disabled)



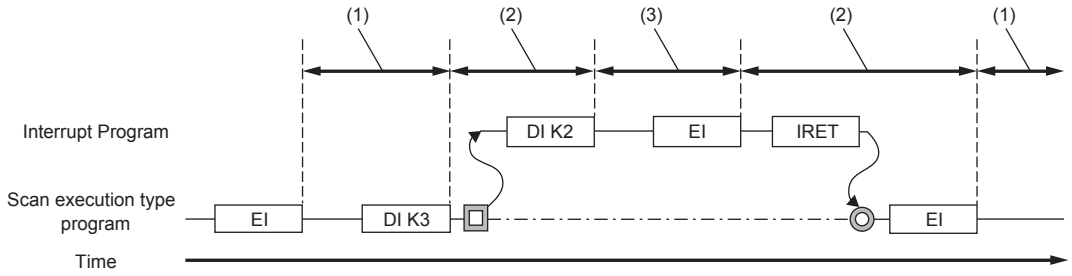

(1) Interrupt-enabled section for all priority
(2) Interrupt-disabled section for the priority 3 or lower (interrupt-enabled section for the priority 2 or higher)
(3) Interrupt-disabled section for the priority 2 or lower (interrupt-enabled section for the priority 1 or higher)

  - When multiple DI instructions are executed (when interrupts with a priority lower than the currently disabled priority are specified and disabled)




(1) Interrupt-enabled section for all priority
(2) Interrupt-disabled section for the priority 2 or lower (interrupt-enabled section for the priority 1 or higher)
(3) Because interrupts with the priority 2 or lower are already disabled, the interrupt-disabling priority is not changed.

• When the DI instruction is executed in an interrupt program

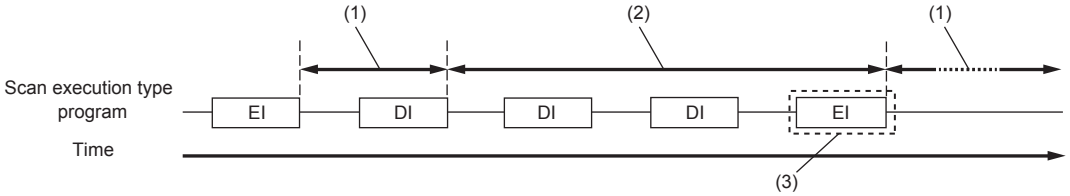



(1) Interrupt-enabled section for all priority

(2) Interrupt-disabled section for the priority 3 or lower (interrupt-enabled section for the priority 2 or higher)

(3) Interrupt-disabled section for the priority 2 or lower (interrupt-enabled section for the priority 1 or higher)

• When the DI instruction without an argument is executed

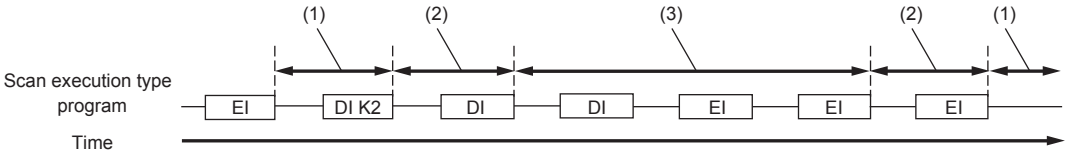



(1) Interrupt-enabled section for all priority

(2) Interrupt-disabled section for the priority 1 or lower (where all the interrupts are disabled)

(3) Because interrupts are disabled with the DI instruction without an argument, interrupts with all the priorities are enabled by executing the EI instruction once.

• When the DI instructions with and without an argument are executed (Execution order is DI instruction with an argument → DI instruction without an argument)

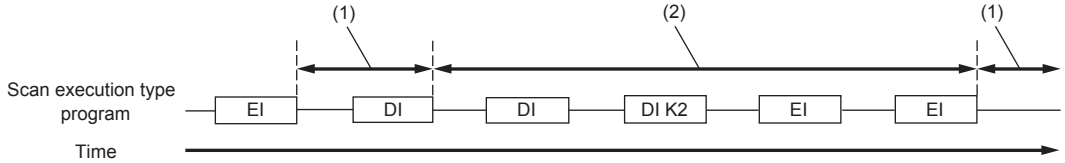



(1) Interrupt-enabled section for all priority

(2) Interrupt-disabled section for the priority 2 or lower (interrupt-enabled section for the priority 1 or higher)

(3) Interrupt-disabled section for the priority 1 or lower (where all the interrupts are disabled)

• When the DI instructions with and without an argument are executed (Execution order is DI instruction without an argument → DI instruction with an argument)




(1) Interrupt-enabled section for all priority

(2) Interrupt-disabled section for the priority 1 or lower (where all the interrupts are disabled)

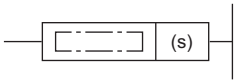
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The value specified by (s) is other than the following.<br>1 to 3 |
| 3362H | Nesting of the DI instruction exceeds 16 levels. |

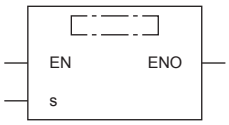
# Interrupt program mask

## IMASK

This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number according to the 16-point bit pattern starting from the device specified in (s).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IMASK(EN,s); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the interrupt mask data is stored<br>The device specified in (s) and following 15 devices are used. | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 16) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

## Processing details

- This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number according to the 16-point bit pattern starting from the device specified in (s).
  - 1 (ON): The execution of interrupt programs is enabled.
  - 0 (OFF): The execution of interrupt programs is disabled.
- The following shows the assignment of the interrupt pointer numbers to each bit.

| | b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (s) | I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| (s)+1 | I31 | I30 | I29 | I28 | - | - | - | - | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| (s)+2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| (s)+3 | I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | - | - |
| (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 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | I177 | I176 |
| (s)+12 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| (s)+13 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| (s)+14 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| (s)+15 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

- When the power is turned on or the CPU module is reset, execution status of the interrupt programs of I0 to I177 is applied.
- The states of the device (s) to (s)+15 are stored in SD1400 to SD1415 (IMASK instruction mask pattern).

**Point**

The IMASK instruction can enable or disable the interrupt pointers I0 to I177 in a batch.

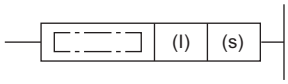
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The 16-point range starting from the device specified by (s) exceeds the corresponding device range. |

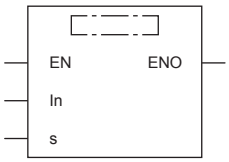
# Disabling/enabling the specified interrupt pointer

## SIMASK

This instruction enables or disables the interrupt pointer number specified by (I) according to the value of (s).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SIMASK(EN,In,s); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (I)*1 | Interrupt pointer number for which interrupts are enabled or disabled | I0 to I177 | Device name | POINTER |
| (s) | Enabled or disabled state of the specified interrupt pointer number | 0: Disabled<br>1: Enabled | 16-bit signed binary | ANY16 |

*1 In the case of the ST language and the FBD/LD language, I displays as In.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (I) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction enables or disables the execution of the interrupt program with the interrupt pointer number specified by (I) according to the data specified by (s)
- When 1 is set in (s): The execution of the interrupt program is enabled.
- When 0 is set in (s): The execution of the interrupt program is disabled.
- When the power is turned on or the CPU module is reset, the execution status of the interrupt programs of I0 to I177 is applied.
- The execution-enabled/disabled states of interrupt pointers are stored in SD1400 to 1415 (IMASK instruction mask pattern).

**Point**

Indexing is available for (I). By using the SIMASK instruction with indexing, the execution of the interrupt pointers I0 to I177 can be enabled or disabled.

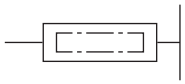
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The interrupt pointer number specified by (I) exceeds the range of the interrupt pointer number (I0 to I177). |
| | The value in (s) is other than the interrupt disabled (0) or interrupt enabled (1). |

# Returning from the interrupt program

## IRET

This instruction indicates an end of the processing of an interrupt program.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Processing details

When an interrupt (input or timer) is generated while the main program is executing, the program execution jumps to an interrupt (I) routine. The IRET instruction returns the program execution to the main routine.
The table below shows two types of jump to an interrupt routine.

| Function | Interrupt No. | Description |
|---|---|---|
| Interrupt from inputs (including counter) | I0 to I23 | Interrupt pointer used for the CPU built-in functions (such as input interrupt, high-speed comparison match interrupt) |
| Internal timer interrupt | I28 to I31 | Interrupt pointer used for fixed-cycle interrupts of the internal timer |

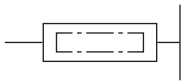
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 33E6H | The IRET instruction is executed in the main program. |

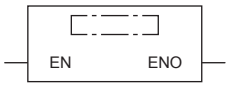
# Resetting the watchdog timer

## WDT(P)

These instructions reset the watchdog timer in a program.

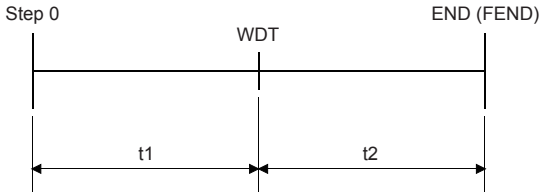
| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WDT(EN);<br>ENO:=WDTP(EN); |

**FBD/LD**




### Processing details

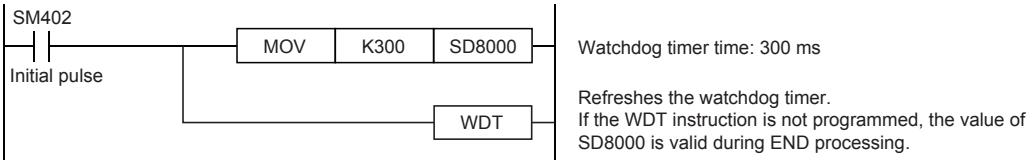
- These instructions reset the watchdog timer in a program.
- These instructions are used when the scan time exceeds the value set for the watchdog timer depending on the condition. If the scan time exceeds the value set for the watchdog timer every scan, change the setting of the watchdog timer in the parameter setting of the engineering tool.
- Design a program so that t1 from the step 0 to the WDT(P) instruction and t2 from the WDT(P) instruction to the END (FEND) instruction do not exceed the setting value of the watchdog timer.




- The WDT(P) instruction can be used more than once in one scan. However, note that turning off the output takes some time if an error occurs.

### Precautions

- The time of the watchdog timer can be changed in the [RAS] tab of [CPU Parameter]. The default value is 200 ms.
- By overwriting the contents of SD8000 (watchdog timer time), the watchdog timer detection time can be changed using a program. When the program shown below is input, the sequence program will be monitored with the new watchdog timer time.




### Operation error

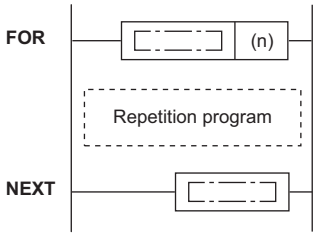
There is no operation error.

# 8.4 Structuring Instruction

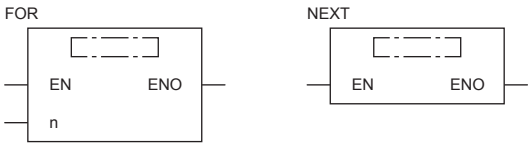
## FOR to NEXT

### FOR, NEXT

When the processing between the FOR and NEXT instructions is executed (n) times without any condition, the processing of the step following the NEXT instruction is executed.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (n) | Number of repetitions of the loop between FOR and NEXT instructions | 1 to 32767 | 16-bit signed binary | ANY16 |

**■Applicable devices**

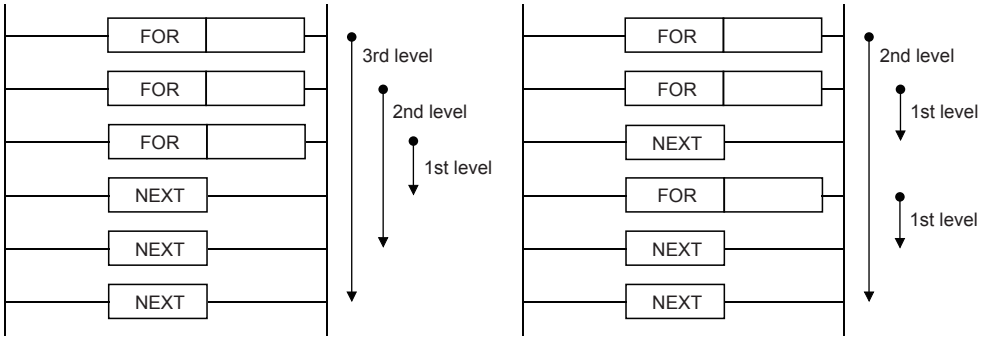
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

#### Processing details

- When the processing between the FOR and NEXT instructions is executed (n) times without any condition, the processing of the step following the NEXT instruction is executed.
- In (n), any of 1 to 32767 can be specified. If any of -32768 to 0 is specified, the processing of (n)=1 is applied.
- To skip the processing between the FOR and NEXT instructions, jump the program execution with the CJ instruction.
- Up to 16 FOR instructions can be nested.

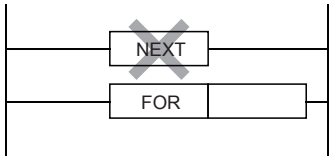
## Precautions

- The FOR-NEXT loop can be nested up to 16 levels.

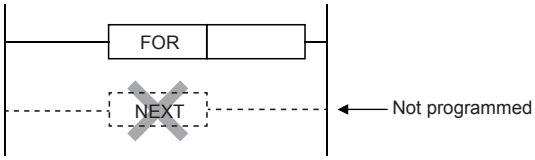



- The FOR-NEXT loop cannot be interrupted by the I, IRET, SRET, RET, FEND, or END instruction.
- When FOR-NEXT loop is repeated many times, the operation cycle is too long, and a watchdog timer error may occur. In such a case, change the watchdog timer time or reset the watchdog timer.
- The following programs are regarded as errors.

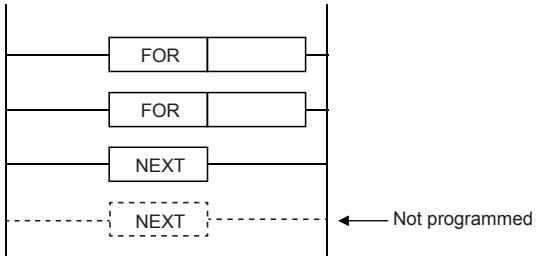
When the NEXT instruction is located before FOR



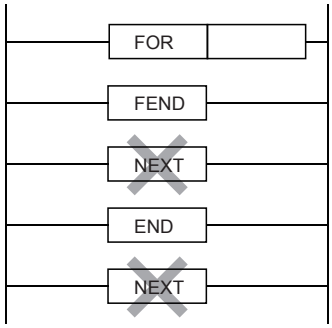

No NEXT instruction




When the number of FOR instructions is not equivalent to the number of NEXT instructions




When the NEXT instruction is located after the FEND or END instruction




## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3340H | After the FOR instruction is executed, the END or GOEND instruction is executed before the NEXT instruction is executed. |
| 3361H | When the FOR instruction is nested, the 17th level is executed. |

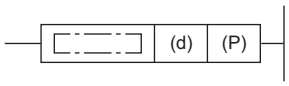
**Point**

- To terminate the FOR to NEXT instruction loop halfway, use the BREAK instruction. (☞ Page 378 Forcibly terminating the FOR to NEXT instruction loop)

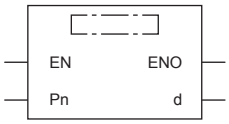
# Forcibly terminating the FOR to NEXT instruction loop

## BREAK(P)

This instruction forcibly terminates the FOR to NEXT instruction loop and shifts the program execution to the pointer specified by (P).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device number storing the number of remaining loops | — | 16-bit signed binary | ANY16 |
| (P)*1 | Pointer number of the branch destination when the loop is forcibly terminated | — | Device name | POINTER |

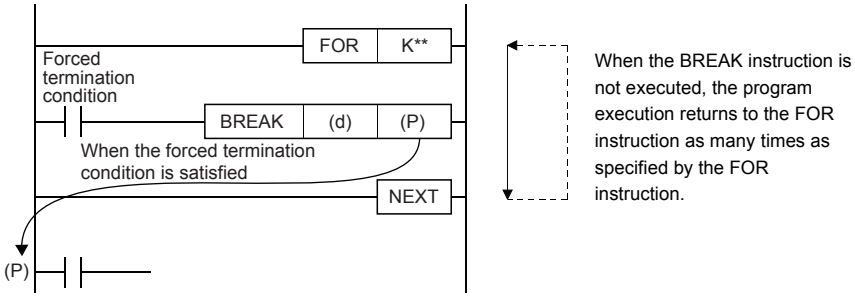
*1 In the case of the FBD/LD language, P displays as Pn.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (P) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |

### Processing details

- This instruction forcibly terminates the FOR to NEXT instruction loop and shifts the program execution to the pointer specified by (P). Only the pointer numbers within the same program file can be specified in (P). If a pointer in another program is specified in (P), an operation error occurs.




- In (d), the number of remaining FOR to NEXT instruction loops at the forced termination is stored. Note that the number includes the loop when the BREAK(P) instructions are executed.
- The BREAK(P) instructions can be used only between the FOR and NEXT instructions.
- The BREAK(P) instructions can be used for only one nesting level. To forcibly terminate multiple nesting levels, execute as many BREAK(P) instructions as the number of nesting levels.

## Precautions

- If the branch pointer number of the BREAK instruction outside two nesting levels or more is specified, an operation error occurs and the program execution stops when the BREAK instruction is executed.

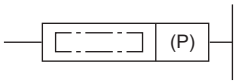
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3340H | The branch pointer number outside two nesting levels or more is specified. |
| 3342H | The BREAK(P) instructions are used other than between the FOR and NEXT instructions. |
| 3380H | The destination pointer specified by (P) does not exist. |
| | A pointer in other program file is specified in (P). |

# Calling a subroutine program

## CALL(P)

This instruction executes the subroutine program specified by (P).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

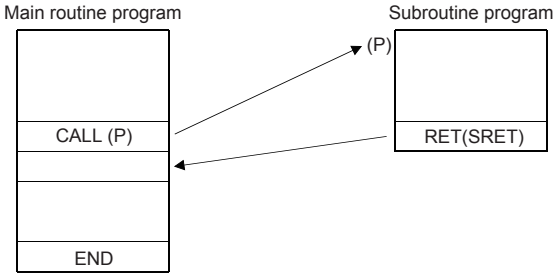
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (P) | Start pointer number of the subroutine program | — | Device name | POINTER |

#### ■Applicable devices

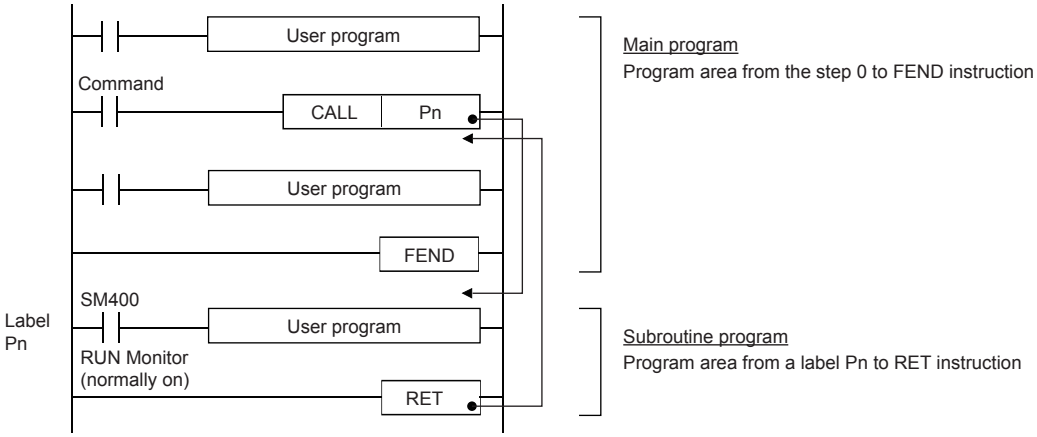
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (P) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |

## Processing details

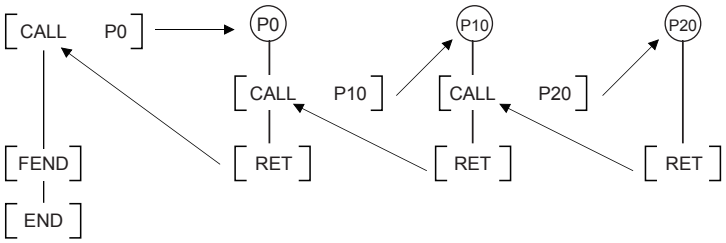
- When the CALL(P) instructions are executed, the subroutine program specified by the pointer (P) is executed. The CALL(P) instructions can execute a subroutine program specified by a pointer in the same program file or by a common pointer.




- While the command input is ON, the CALL instruction is executed and the program execution jumps to a step with a label (Pn). Then, a subroutine program with the label (Pn) is executed. When the RET (SRET) instruction is executed, the program execution returns to the step following the CALL instruction. At the end of the main program, put FEND instruction. Put a label (Pn) for the CALL instruction after the FEND instruction.

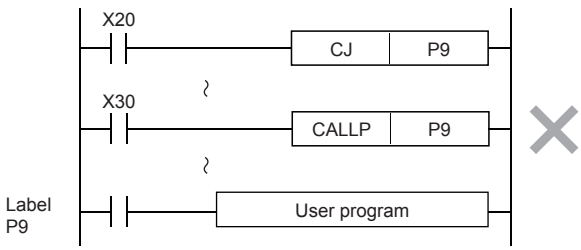



- The CALL(P) instructions can be nested up to 16 levels. However, the 16 levels are the total of the CALL(P) and XCALL instructions.




## Precautions

- In the CALL instruction, the same number can be used two or more times in operands (P). However, do not use a label (P) and number used in another instruction (CJ instruction).




- In a subroutine (or interrupt routine), use timers for routine programs. These timers count when a coil instruction or END instruction is executed. After a timer reaches the set value, the output contact is activated when the coil instruction or END instruction is executed. Because general timers count only when the coil instruction is executed, they do not count if they

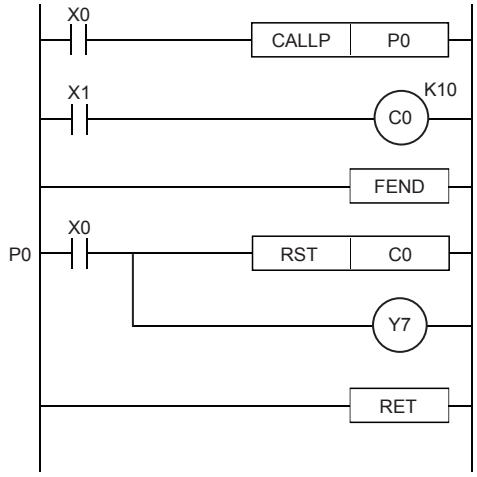
are used in subroutines in which the coil instruction is executed only under some conditions.

- If a retentive type 1 ms timer is used in a subroutine (interrupt routine), note that the output contact is activated when the first coil instruction (or subroutine) is executed after the timer reaches its set value.
- Devices which were set to ON in a subroutine (or interrupt routine) are latched in the ON status even after the subroutine is finished. (Refer to the program example shown below). When the RST instruction for a timer or counter is executed, the reset status of the timer or counter is latched also. For turning OFF such a device latched in the ON status or for canceling such a timer or counter latched in the reset status, reset such a device in the main program after the subroutine is finished, or program a sequence for resetting such a device or for deactivating the RST instruction in the subroutine. (Refer to the program example shown below).

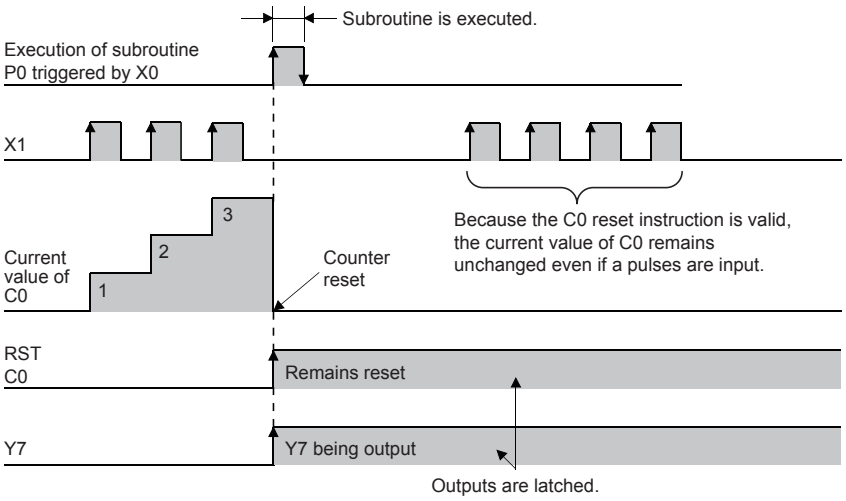
Example in which outputs are latched

In the following program example, the counter C0 is provided to count X1. When X0 is input, the subroutine P0 is executed only in one scan, and then the counter is reset and Y7 is output.

[Program example]

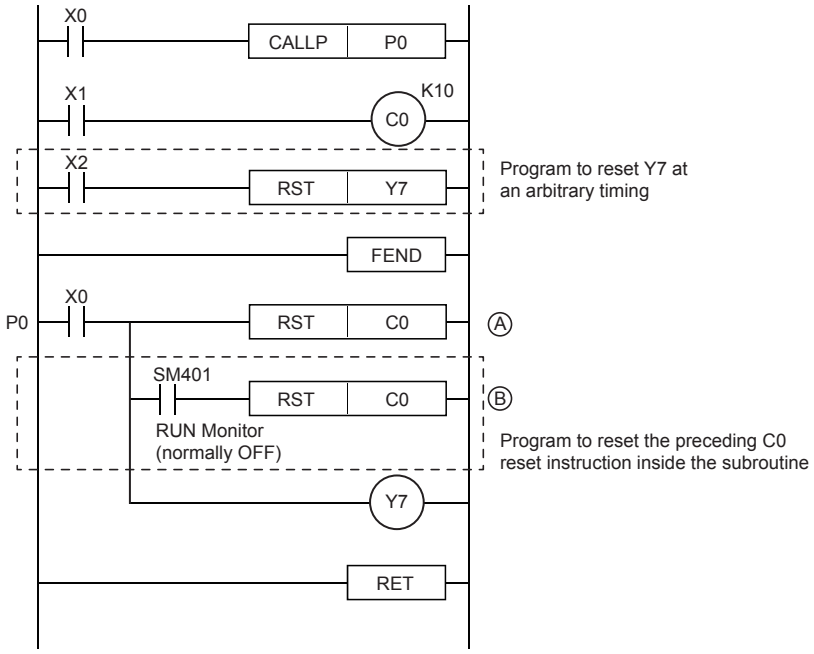



[Timing chart]

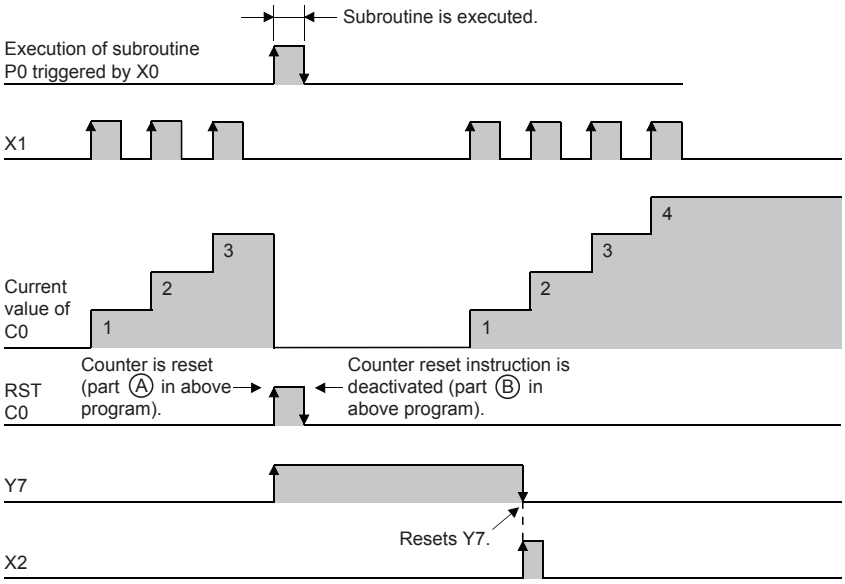



Example in which latched outputs are reset (countermeasures)

[Program example]




[Timing chart]




## Operation error

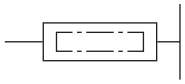
| Error code (SD0/SD8067) | Description |
|---|---|
| 3360H | The 17th level of the nesting is executed. |
| 3380H | The subroutine program specified by the pointer in the CALL(P) instructions do not exist. |
| 3381H | After the CALL(P) instructions are executed, the END, FEND, GOEND, or STOP instruction is executed before the RET (SRET) instruction is executed. |
| 3382H | The RET (SRET) instruction is executed before the CALL(P) instructions are executed. |

# Returning from the subroutine program

## RET/SRET

These instructions indicate an end of a subroutine program.

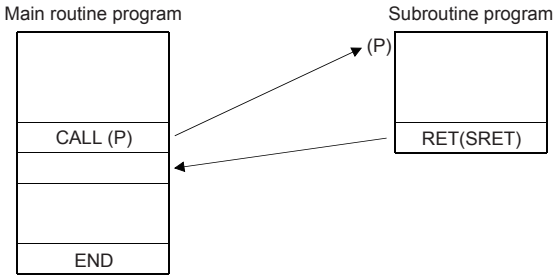
The RET instruction can be used as SRET.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Processing details

- These instructions indicate an end of a subroutine program.
- When the RET instruction is executed, the program execution returns to the step following the CALL(P) or XCALL instruction that called the subroutine program.




### Precautions

- If the RET (SRET) instruction is executed in a user interrupt program (I-IRET), a compiling error occurs.

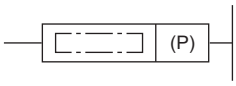
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3381H | The END, FEND, GOEND, or STOP instruction is executed before the RET instruction is executed. |
| 3382H | While the number of nesting levels is decreased by the return instruction, the result becomes negative.<br>(The number of RET (SRET) instructions is larger than that of the CALL instructions.) |

# Calling a subroutine program

## XCALL

This instruction executes CALL for (turns on and executes) the subroutine program specified by (P) when the execution condition is established. When the condition is turned off, this instruction executes turns off and terminates for the subroutine program.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (P) | Start pointer number of the subroutine program | — | Device name | POINTER |

#### ■Applicable devices

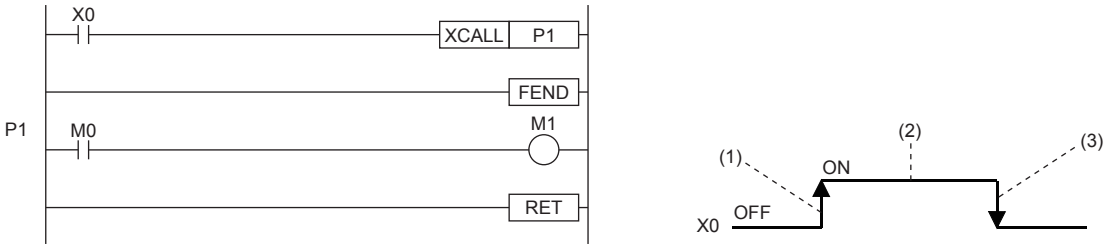
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (P) | — | — | — | — | — | — | — | — | — | — | — | — | ○ |

### Processing details

- The XCALL instruction controls the execution and non-execution processing of subroutine programs.
  - In the execution of subroutine programs, each coil instruction is operated according to the ON/OFF status of the condition contact.
  - In the non-execution processing of subroutine programs, each coil instruction is operated with the OFF status of the condition contact applied.
- The following table lists the operation result of each coil instruction after the non-execution processing. Regardless of the status of the condition contact, the following result is applied.

| Device used for operation | Operation result (device status) |
|---|---|
| 1 ms timer, 10 ms timer, 100 ms timer | 0 |
| 1 ms retentive timer, 10 ms retentive timer, 100 ms retentive timer, counter | The current status is held. |
| Device in the OUT instruction | Forcibly turned off. |
| Device in the SET, RST, or SFT(P) instruction or basic/applied instruction | The current status is held. |
| PLS instruction, pulse instruction (□P) | Same as when the condition contact is off |

- The following shows the operation of the XCALL instruction.

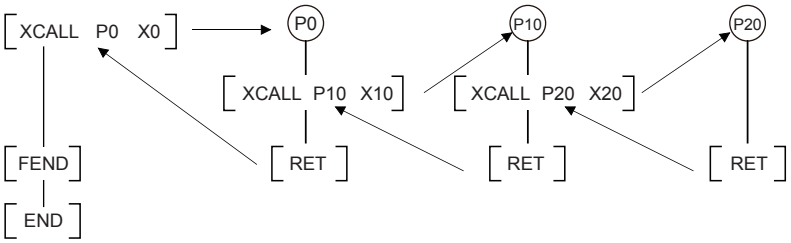



(1) Rising edge of X0 (OFF → ON): The subroutine program of P1 is executed.

(2) While X0 is on: The subroutine program of P1 is executed. (The rising edge of X0 is not included.)

(3) Falling edge of X0 (ON → OFF): The non-execution processing of the subroutine program of P1 is executed.

- The XCALL instruction can be nested up to 16 levels. However, the 16 levels are the total of the CALL(P) and XCALL instructions.




## Operation error

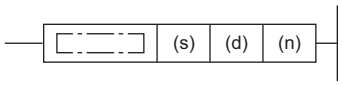
| Error code (SD0/SD8067) | Description |
|---|---|
| 3360H | The 17th level of the nesting is executed. |
| 3380H | The subroutine program specified by the pointer in the XCALL instruction does not exist. |
| 3381H | After the XCALL instruction is executed, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction is executed. |

# 8.5 Data Table Operation Instruction

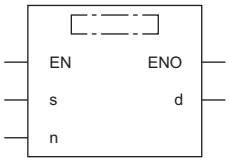
## Reading the oldest data from the data table

### SFRD(P)

These instructions read data for first-in first-out control.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFRD(EN,s,n,d);<br>ENO:=SFRDP(EN,s,n,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

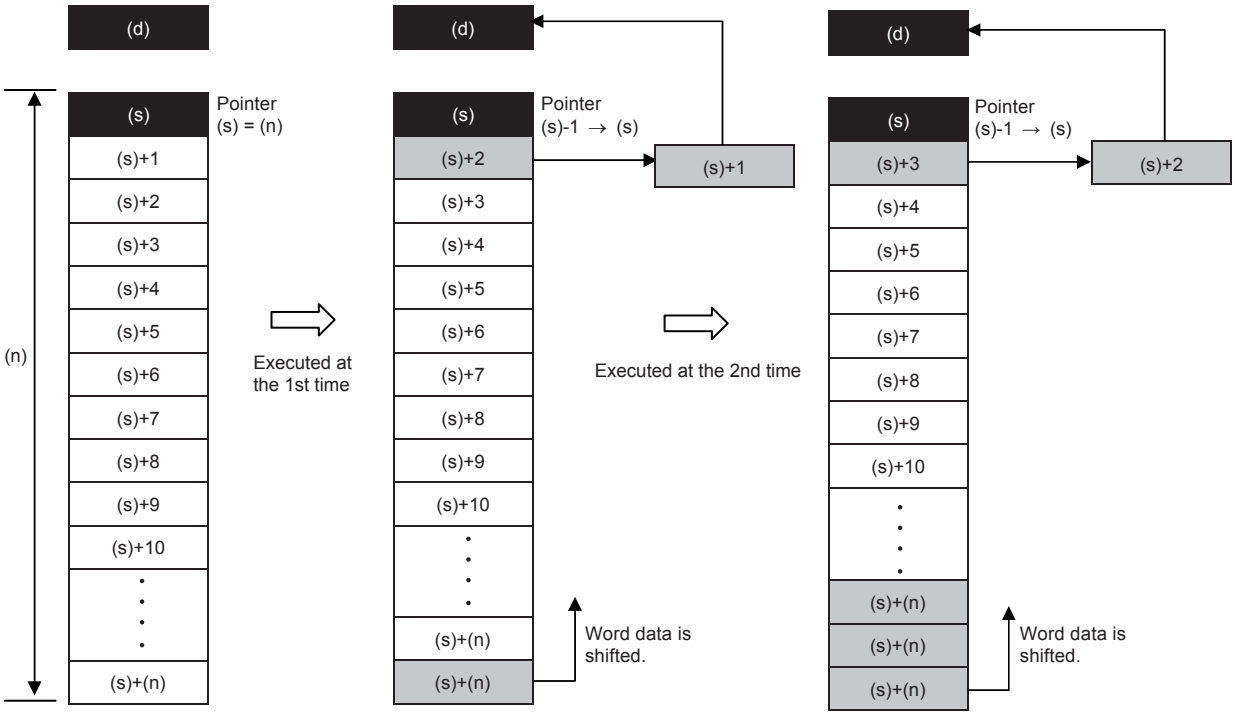
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Start number of the word device storing the data<br>(The start is a pointer. The data is stored starting from (s)+1.) | — | 16-bit signed binary | ANY16 |
| (d) | Word device number storing data taken out first | — | 16-bit signed binary | ANY16 |
| (n) | Number of stored points plus "1". "+1" is required for the pointer. | 2 to 32768 | 16-bit unsigned binary | ANY16_U |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- These instructions transfer (read) (s)+1, which was sequentially written by the SFWR instruction, to (d), and shift the word data of (n)-1 points starting from (s)+1 upward by 1 word. Then, these instructions decrease the number of data points stored in (s) by 1.




- The data of (s)+1 is transferred (read) to (d). Accompanied by this transfer, the contents of the pointer (s) decrease, and the data is shifted upward by 1 word. (When the continuous operation type SFRD instruction is used, the contents are stored in turn in each operation cycle. Use the pulse operation type SFRDP instruction in programming.)

## Precautions

- The contents of (s)+(n) do not change by reading.
- When the continuous operation type (SFRD) instruction is used, data is read in turn in each scan time (operation cycle), but the contents of (s)+(n) do not change.
- When 0 is set in the pointer (s), no processing is executed and the contents of (d) do not change.

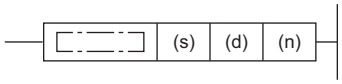
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of device points (n) from (s) exceed the device range. |
| 3405H | The value set in (n) is other than the following.<br>2≤(n)≤32768 |
| | In (s), a negative value is specified. |

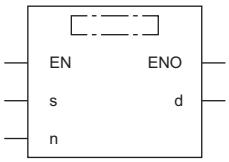
# Reading the newest data from the data table

## POP(P)

These instructions read the latest data written by a shift write (SFWR) instruction for FIFO/FILO control.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=POP(EN,s,n,d);<br>ENO:=POPP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing the first-in data (including pointer data) (start number of the word device storing the data) | — | 16-bit signed binary | ANY16 |
| (d) | Device number storing last-out data | — | 16-bit signed binary | ANY16 |
| (n) | Length of data array<br>(Add "1" because pointer data is also included.) | 2 to 32768 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

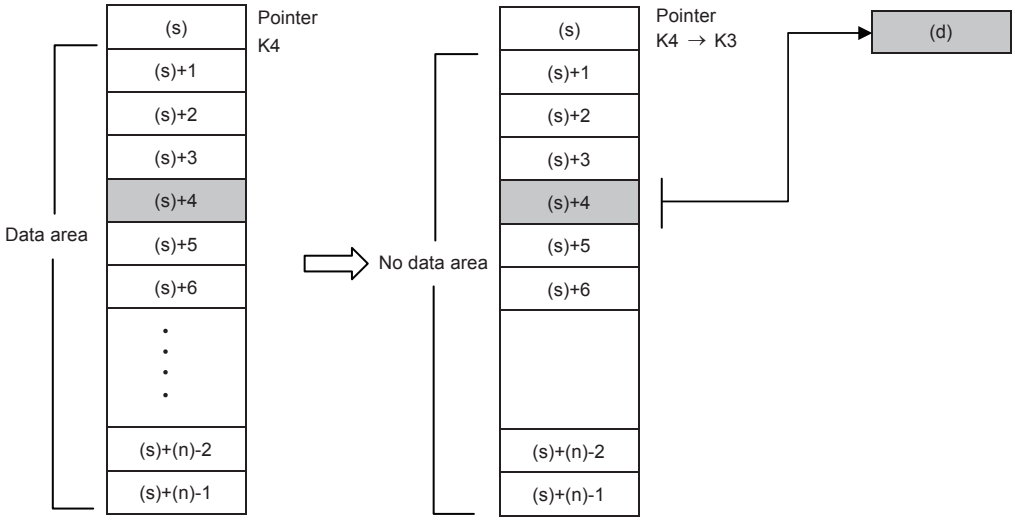
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- Every time the instruction is executed for the word devices (s) to (s)+(n)-1, a device "(s) + Pointer data (s)" is read to (d). (The last data entry written by the shift write (SFWR) instruction for first-in first-out control is read to (d).) Specify any value between 2 and 32767 for (n).
- Subtract "1" from the value of the pointer data (s).

Data for FILO control

| | Description |
|---|---|
| (s) | Pointer data (amount of data stored) |
| (s)+1 | Data area<br>(First-in data written by shift write (SFWR) instruction) |
| (s)+2 | |
| (s)+3 | |
| ⋮ | |
| (s)+(n)-3 | |
| (s)+(n)-2 | |
| (s)+(n)-1 | |




## Precautions

- If programed in the continuous operation type, the POP(P) instructions are executed in every operation cycle. As a result, expected operation may not be achieved. Usually, program the POP(P) instructions in the "pulse operation type", or let them be executed by a "pulsed command contact".
- When the current value of the pointer (s) is "0", the zero flag SM8020 turns ON and the POP(P) instructions are not executed.
- When the current value of the pointer (s) is "1", "0" is written to (s) and the zero flag SM8020 turns ON.

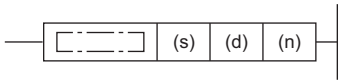
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range (s)+(n)-1 exceeds the device. |
| 3405H | (s) is larger than (n)-1. |
| | (s) is smaller than 0. |
| | The value set in (n) is other than the following.<br>2≤(n)≤32768 |

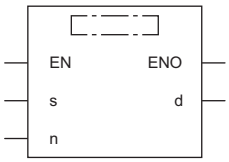
# Writing data to the data table

## SFWR(P)

These instructions write data for first-in first-out (FIFO) and last-in first-out (LIFO) control.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SFWR(EN,s,n,d);<br>ENO:=SFWRP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

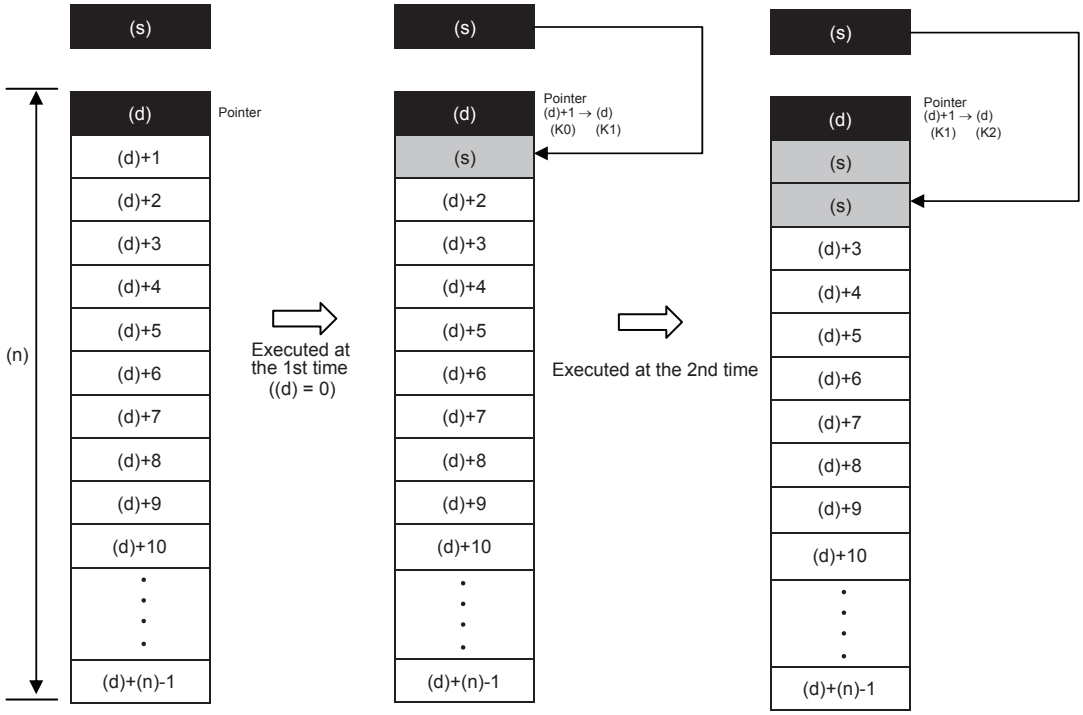
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Word device number storing data to be put in first | — | 16-bit signed binary | ANY16 |
| (d) | Start word device number storing and shifting data<br>(The start is a pointer. The data is stored starting from (d)+1.) | — | 16-bit signed binary | ANY16 |
| (n) | Number of stored points plus "1". | 2 to 32768 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- The contents of (s) are written to "(n)-1" devices from (d)+1, and "1" is added to the number of data stored in (d). For example, for (d)=0, the contents are written to (d)+1, and for (d)=1, to (d)+2.




- At the first execution, the contents of (s) are stored in (d)+1.
- When the contents of (s) are changed and then the instruction is executed again, the new contents of (s) are stored to (d)+2. So the contents of +2 become equivalent to (s). (When the continuous operation type SFWR instruction is used, the contents are stored in each operation cycle. Use the pulse operation type SFWRP instruction in programming.) Data is stored from the right end in the same way, and the number of stored data is specified by the contents of the pointer (d).

## Precautions

- In the case of the continuous operation type instruction (SFWR), note that data is stored (overwritten) in every scan time (operation cycle).

## Operation error

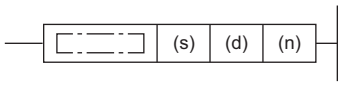
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of device points (n) from (d) exceeds the device range. |
| 3405H | The value set in (n) is other than the following.<br>2≤(n)≤32768 |
| | In (d), a negative value is specified. |

# Inserting data to the data table

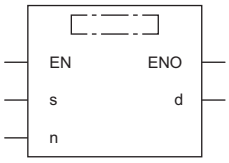
## FINS(P)

These instructions insert 16-bit data specified by (s) to the data table specified by (d) as the (n)th data.

After these instructions are executed, the data after the (n)th data in the data table is moved down by one data point.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FINS(EN,s,n,d);<br>ENO:=FINSP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

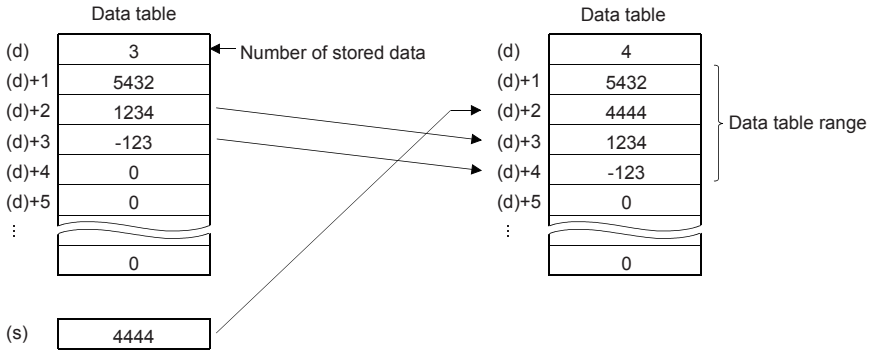
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the insertion-target data is stored | — | 16-bit signed binary | ANY16 |
| (d) | Start number of the table | — | Word | ANY16 |
| (n) | Data insertion position in the table | 1 to 32767 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions insert 16-bit binary data specified by (s) to the data table specified by (d) as the (n)th data. After these instructions are executed, the data after the (n)th data in the data table is moved down by one data point.




When 2 is set in (n), the data is inserted in (d)+2.

### Precautions

- The device range used in a data table should be controlled by the user.
- The data table has (d) number of stored data starting from ((d)+1).

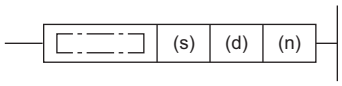
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | When the FINS(P) instructions are executed, the data table range exceeds the corresponding device range. |
| 3405H | When the FINS(P) instructions are executed, the value (n) exceeds the corresponding device range of the table (d). |
| | When the FINS(P) instructions are executed, the table position (n) where the data is inserted exceeds "the number of stored data points + 1". |
| | The value set in (n) is other than the following.<br>2≤(n)≤32767 |

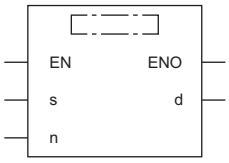
# Deleting data from the data table

## FDEL(P)

These instructions remove the (n)th data in the data table specified by (d) and store the data in the device specified by (s). After these instructions are executed, the data after the (n)+1th data in the data table is moved up by one data point.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FDEL(EN,s,n,d);<br>ENO:=FDELP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

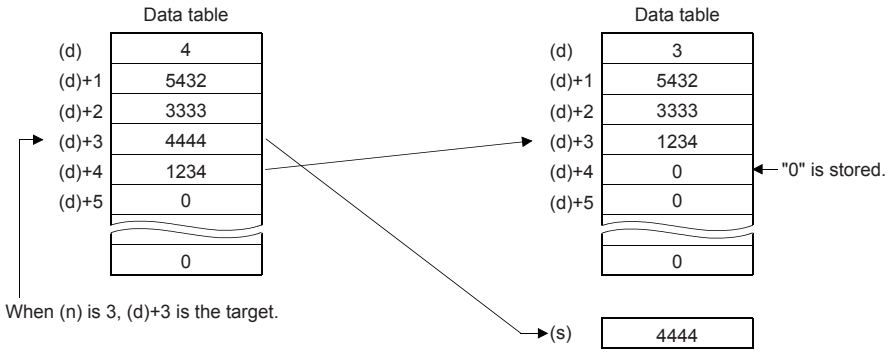
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number for storing the data to be deleted | — | 16-bit signed binary | ANY16 |
| (d) | Start number of the table | — | Word | ANY16 |
| (n) | Position of the data to be deleted in the table | 1 to 32767 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions remove the (n)th data in the data table specified by (d) and store the data in the device specified by (s). After these instructions are executed, the data after the (n)+1th data in the data table is moved up by one data point.




### Precautions

- The device range used in a data table should be controlled by the user.
- The data table has (d) number of stored data starting from ((d)+1).

## Operation error

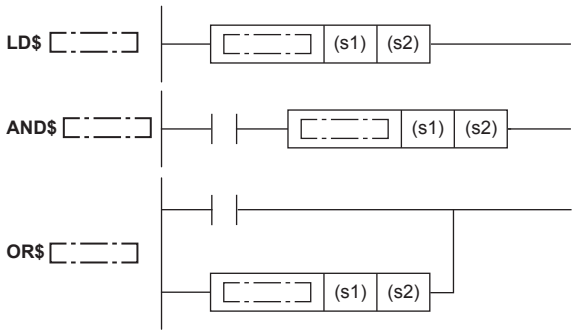
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | When the FDEL(P) instructions are executed, the data table range exceeds the corresponding device range. |
| 3405H | When the FDEL(P) instructions are executed, the value (n) exceeds the corresponding device range of the table (d). |
| | When 0 is set in (d), and the FDEL(P) instructions are executed. |
| | When the FDEL(P) instructions are executed, the table position (n) where the data to be deleted is stored exceeds the number of stored data points. |
| | The value set in (n) is other than the following.<br>2≤(n)≤32767 |

# 8.6 Character String Operation Instruction

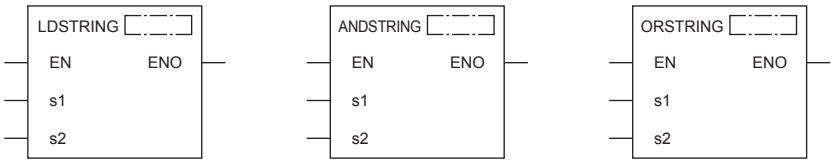
## Comparing character strings

### LD$□, AND$□, OR$□

These instructions perform a comparison operation between the character string data in the device specified by (s1) and later and the character string data in the device specified by (s2) and later. (Devices are used as a normally open contact.)

| Ladder diagram | Structured text |
|---|---|
|  ("=", "<>", ">", "<=", "<", ">=" enters □.) | Not supported |

**FBD/LD**




("_EQ", "_NE", "_GT", "_LE", "_LT", "_GE" enters □.)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Comparison data or head device number where the comparison data is stored | — | Character string | ANYSTRING_SINGLE |
| (s2) | Comparison data or head device number where the comparison data is stored | — | Character string | ANYSTRING_SINGLE |

##### ■Applicable devices

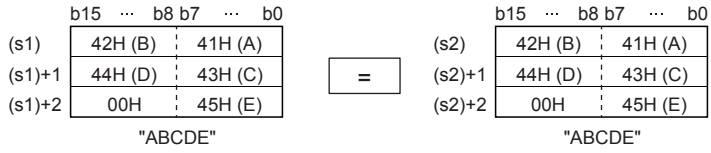
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (s2) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |

*1 T, ST, C cannot be used.

#### Processing details

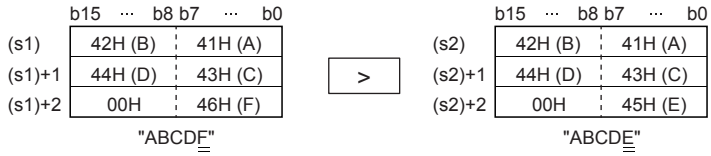
- These instructions perform a comparison operation between the character string data specified by (s1) and the character string data specified by (s2). (Devices are used as a normally open contact.)
- In the comparison operation, the ASCII codes of the character strings are compared one by one from the start of the strings.

- Character strings in the devices specified by (s1) and (s2) to a device that stores 00H are compared.
  - When all the character strings match, the comparison is considered as matched.



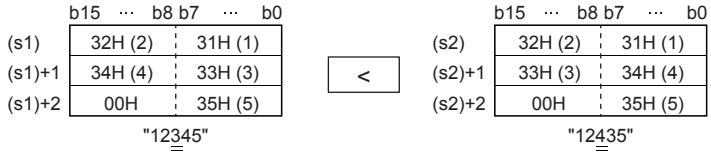

| Instruction symbol in ☐ | Result | Instruction symbol in ☐ | Result |
|---|---|---|---|
| $= | Conductive state | $<= | Conductive state |
| $<> | Non-conductive state | $< | Non-conductive state |
| $> | Non-conductive state | $>= | Conductive state |

  - When the character strings are different, the string with a large character code is considered as the large one.



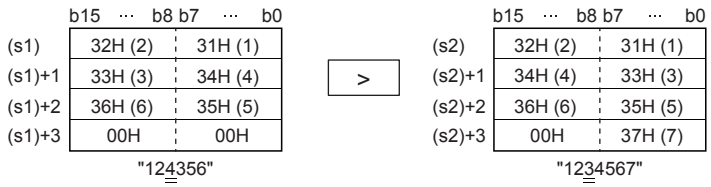

| Instruction symbol in ☐ | Result | Instruction symbol in ☐ | Result |
|---|---|---|---|
| $= | Non-conductive state | $<= | Non-conductive state |
| $<> | Conductive state | $< | Non-conductive state |
| $> | Conductive state | $>= | Conductive state |

  - When the character strings are different, the magnitude relation between them is determined based on the size of the first different character code.




| Instruction symbol in ☐ | Result | Instruction symbol in ☐ | Result |
|---|---|---|---|
| $= | Non-conductive state | $<= | Conductive state |
| $<> | Conductive state | $< | Conductive state |
| $> | Non-conductive state | $>= | Non-conductive state |

  - When the length of the character string data differs for (s1) and (s2), the relative sizes of the character strings are determined based on the size of the first different character code.



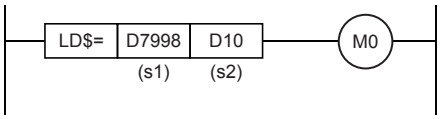

| Instruction symbol in ☐ | Result | Instruction symbol in ☐ | Result |
|---|---|---|---|
| $= | Non-conductive state | $<= | Non-conductive state |
| $<> | Conductive state | $< | Non-conductive state |
| $> | Conductive state | $>= | Conductive state |

- If the character string specified by (s1) or (s2) has more than 16383 characters, the operation result is the non-conductive state.

## Precautions

- In character string comparison operation, if the target device range does not have "00H", the values until the last number of the device are retrieved. Thus, even if the target device range does not have "00H", a comparison operation result is output when a mismatch between the acquired character strings is detected.

[Example]




Data of (s1)

| | | |
|---|---|---|
| D7998 | 42H(B) | 41H(A) |
| D7999 | 44H(D) | 43H(C) |

Data of (s2)

| | | |
|---|---|---|
| D10 | 5A(Z) | 41H(A) |
| D11 | 00H | 43H(C) |

- For the data specified by (s1) and (s2) as shown above, the second character is different between them. Thus, the operation result is non-conductive.

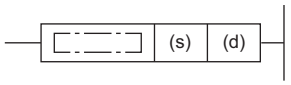
## Operation error

There is no operation error.

# Concatenating character strings

## $+(P) [For 2 operands]

These instructions concatenate the character string data stored in the device specified by (s) and later to the end of the character string data stored in the device specified by (d) and later, and store the concatenated string in the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing data to be concatenated or data, or directly specified character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number storing data to which another data is concatenated | — | Character string | ANYSTRING_SINGLE |

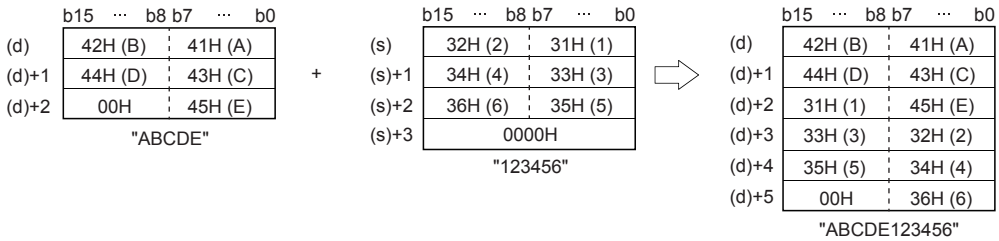
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions concatenate the character string data stored in the device specified by (s) and later to the end of the character string data stored in the device specified by (d) and later, and store the concatenated string in the device specified by (d) and later.



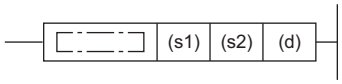

- Character strings in the devices specified by (s) and (d) up to a device that stores 00H are concatenated.
- When character strings are concatenated, 00H indicating an end of the character string specified by (d) is ignored and the character string specified by (s) is concatenated to the last character of (d).

## Operation error

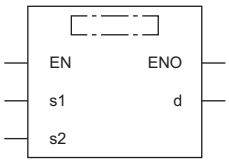
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range after the device specified by (s), "00H" does not exist. |
| | In the corresponding device range after the device specified by (d), "00H" does not exist. |
| 3406H | The whole concatenated character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |
| | The number of characters of the character string in the device specified by (s)+(d) exceeds 16383.. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| | The character string specified by (d) has more than 16383 characters. |

## $+(P) [For 3 operands]

These instructions concatenate the character string data stored in the device specified by (s2) and later to the end of the character string data stored in the device specified by (s1) and later, and store the concatenated string in the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("STRINGPLUS", "STRINGPLUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number storing data to which another data is concatenated or data, or directly specified character string | — | Character string | ANYSTRING_SINGLE |
| (s2) | Head device number storing data to be concatenated or data, or directly specified character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number for storing the concatenated data | — | Character string | ANYSTRING_SINGLE |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (s2) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions concatenate the character string data stored in the device specified by (s2) and later to the end of the character string data stored in the device specified by (s1) and later, and store the concatenated string in the device specified by (d) and later.
- Character strings in the devices specified by (s1) and (s2) up to a device that stores 00H are concatenated.

| | b15 ··· b8 | b7 ··· b0 |
|---|---|---|
| (s1) | 46H (F) | 48H (H) |
| (s1)+1 | 2DH (-) | 41H (A) |
| (s1)+2 | 00H | |

+

| | b15 ··· b8 | b7 ··· b0 |
|---|---|---|
| (s2) | 35H (5) | 31H (1) |
| (s2)+1 | 39H (9) | 33H (3) |
| (s2)+2 | 00H | 41H (A) |

⇨

| | b15 ··· b8 | b7 ··· b0 |
|---|---|---|
| (d) | 46H (F) | 48H (H) |
| (d)+1 | 2DH (-) | 41H (A) |
| (d)+2 | 35H (5) | 31H (1) |
| (d)+3 | 39H (9) | 33H (3) |
| (d)+4 | 00H | 41H (A) |

- When character strings are concatenated, 00H indicating an end of the character string specified by (s1) is ignored and the character string specified by (s2) is concatenated to the last character of (s1).
- After two character strings are connected, "00H" is automatically added at the end. When the number of characters after the concatenation is odd, 00H is stored in the upper byte of the device storing the last character. When the number is even, 0000H is stored in the device after the last character.

## Precautions

- For direct specification, up to 32 characters can be specified (input). When word devices are specified in (s1) or (s2), this restriction (up to 32 characters) is not applicable.
- When the values in both (s1) and (s2) start from "00H" (that is, when the number of characters is "0"), "0000H" is stored in (d).

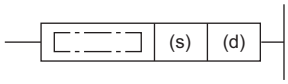
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range after the device specified by (s1), "00H" does not exist. |
| | In the corresponding device range after the device specified by (s2), "00H" does not exist. |
| 2821H | The numbers of the character string-storing devices specified by (s1), (s2), and (d) overlap. |
| 3405H | The character string specified by (s1) has more than 16383 characters. |
| | The character string specified by (s2) has more than 16383 characters. |
| 3406H | The character string specified by (d) has more than 16383 characters. |
| | The whole concatenated character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

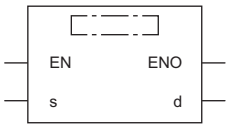
# Transferring character strings

## $MOV(P)

These instructions transfer the character string data specified by (s) to the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("STRINGMOV", "STRINGMOVP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Character string to be transferred (up to 255 characters) or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number storing transferred character string | — | Character string | ANYSTRING_SINGLE |

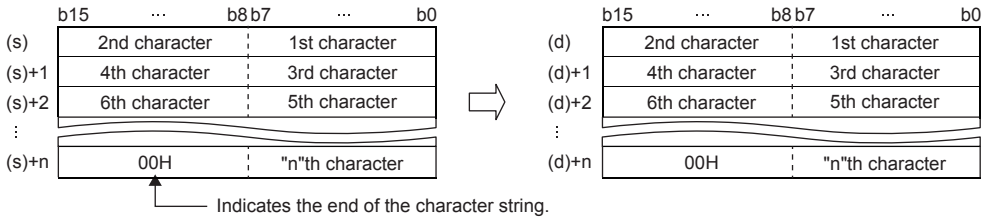
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

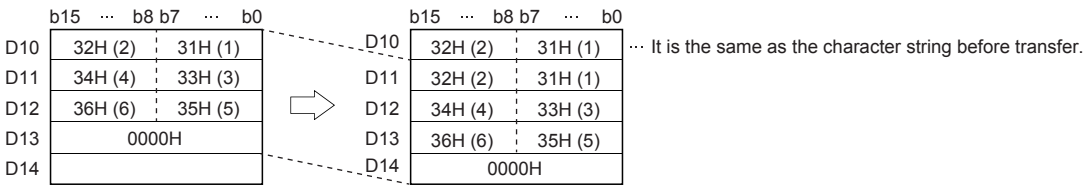
*1 T, ST, C cannot be used.

### Processing details

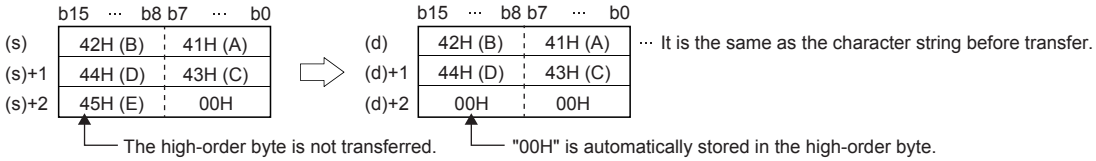
- These instructions transfer the character string data specified by (s) to the device specified by (d) and later. A character string enclosed with double quotation marks and specified by (s) or stored in the devices from the device specified by (s) to the device storing 00H is transferred in a batch.




- Even though the device range of the data to be transferred (s) to (s)+n and the device range for storing the transferred data (d) to (d)+n overlap, the processing is performed normally. For example, when a character string stored in D10 to D13 is transferred to D11 to D14, the transfer is executed as shown below:



• When "00H" is stored in the lower byte of (s)+n, "00H" is stored to both the upper byte and lower byte of (d)+n.




## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

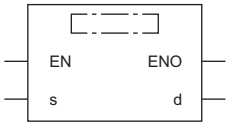
# Converting 16-bit binary data to decimal ASCII

## BINDA(P)(_U)

These instructions convert 16-bit binary data specified by (s) into decimal ASCII codes, and store the converted data in the device specified by (d) and later.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=BINDA(EN,s,d);<br>ENO:=BINDAP(EN,s,d) | ENO:=BINDA_U(EN,s,d);<br>ENO:=BINDAP_U(EN,s,d) |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | BINDA(P) | Binary data to be converted into ASCII codes | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BINDA(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | | Head device number storing conversion result | — | Character string | ANYSTRING_SINGLE |

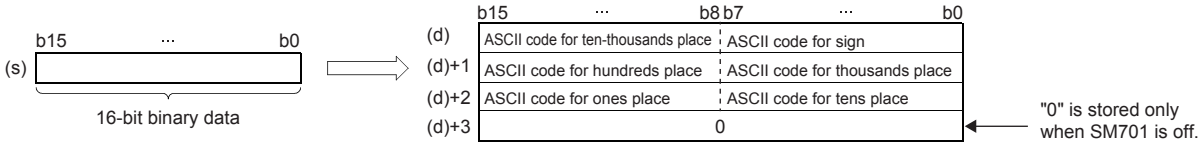
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

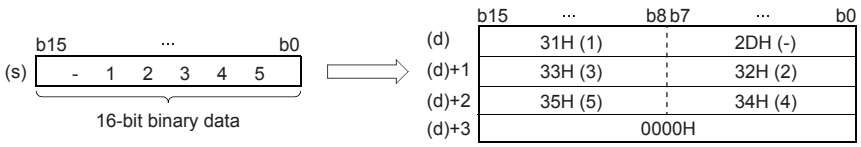
### Processing details

- These instructions convert 16-bit binary data specified by (s) into decimal ASCII codes, and store the converted data in the device specified by (d) and later.




**Ex.**

When -12345 is specified in (s) (when signed data is specified)




- The following shows the operation result to be stored in (d).
  - As sign data, "20H" is stored if the 16-bit binary data is positive, and "2DH" is stored if the data is negative.
  - "20H" is stored for "0" on the left side of the valid digits (zero suppression). For "00325", 20H is stored for "00", and the number of digits is 3 based on "325".
  - In the device specified by (d)+3, 0 is stored when SM701 (output character number selector signal) is off, and the original data remains when SM701 is on.

## Precautions

- The number of occupied points of (d) is 3 when SM701 is on, and 4 when SM701 is off.

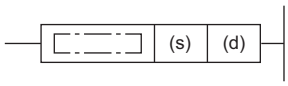
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (d) exceeds the corresponding device range. |

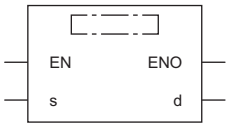
# Converting 32-bit binary data to decimal ASCII

## DBINDA(P)(_U)

These instructions convert 32-bit binary data specified by (s) into decimal ASCII codes, and store the converted data in the device specified by (d) and later.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DBINDA(EN,s,d);<br>ENO:=DBINDAP(EN,s,d); | ENO:=DBINDA_U(EN,s,d);<br>ENO:=DBINDAP_U(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DBINDA(P) | Binary data to be converted into ASCII codes | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBINDA(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | | Head device number storing conversion result | — | Character string | ANYSTRING_SINGLE |

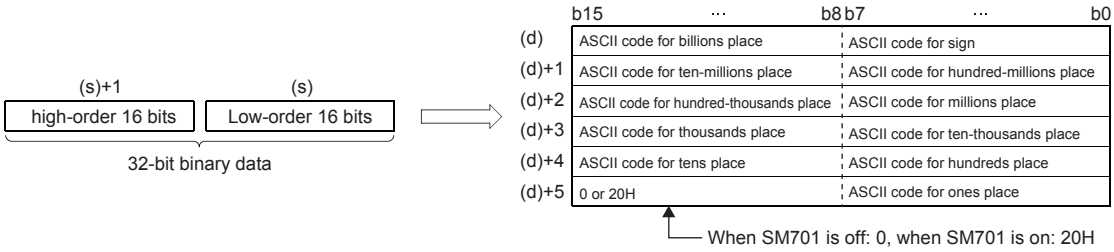
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

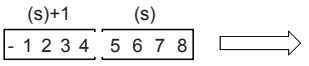
### Processing details

- These instructions convert 32-bit binary data specified by (s) into decimal ASCII codes, and store the converted data in the device specified by (d) and later.



**Ex.**

When -12345678 is specified in (s) (when signed data is specified)




| | b15 ··· b8 | b7 ··· b0 |
|---|---|---|
| (d) | 20H (space) | 2DH (-) |
| (d)+1 | 31H (1) | 20H (space) |
| (d)+2 | 33H (3) | 32H (2) |
| (d)+3 | 35H (5) | 34H (4) |
| (d)+4 | 37H (7) | 36H (6) |
| (d)+5 | 00H or 20H | 38H (8) |

- The following shows the operation result to be stored in (d).
  - As sign data, "20H" is stored if the 16-bit binary data is positive, and "2DH" is stored if the data is negative.
  - "20H" is stored for "0" on the left side of the valid digits (zero suppression). For "0012034560", 20H is stored for "00", and the number of digits is 8 based on "12034560".
  - In the upper 8 bits of the device specified by (d)+5, 0 is stored when SM701 (output character number selector signal) is off, and 20H is stored when SM701 is on.

## Precautions

- (d) occupies six points.

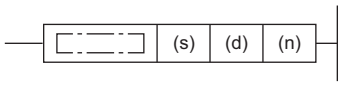
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (d) exceeds the corresponding device range. |

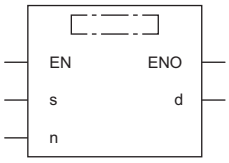
# Converting HEX code data to ASCII

## ASCI(P)

These instructions convert the (n) characters (digits) within the hexadecimal code data specified by (s) to ASCII, and store the converted data in the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ASCI(EN,s,n,d);<br>ENO:=ASCIP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing hexadecimal code to be converted | — | 16-bit signed binary | ANY16 |
| (d) | Head device number storing converted ASCII code | — | Character string | ANYSTRING_SINGLE |
| (n) | Number of characters (digits) of hexadecimal code to be converted | 1 to 32767 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

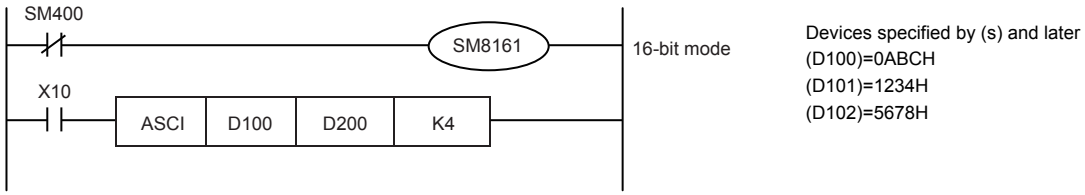
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

- These instructions convert the (n) characters (digits) within the hexadecimal code data specified by (s) to ASCII, and store the converted data in the device specified by (d) and later.
- The 16-bit mode and 8-bit mode options are available for the ASCI(P) instructions. For the operation in each mode, refer to the proceeding pages.
  - 16-bit conversion mode (while SM8161 is OFF)

    Each digit of hexadecimal data stored in the device specified by (s) and later is converted into ASCII code, and transferred to the upper 8 bits and lower 8 bits of each device specified by (d) and later. SM8161 must always be off in the 16-bit conversion mode.
    In the following program, conversion is executed as follows:




■Number of specified digits (characters) and conversion result

| (n) / (d) | K1 | K2 | K3 | K4 | K5 | K6 | K7 | K8 | K9 |
|---|---|---|---|---|---|---|---|---|---|
| D200 lowest-order byte | "C" | "B" | "A" | "0" | "4" | "3" | "2" | "1" | "8" |
| D200 highest-order byte | | "C" | "B" | "A" | "0" | "4" | "3" | "2" | "1" |
| D201 lowest-order byte | | | "C" | "B" | "A" | "0" | "4" | "3" | "2" |
| D201 highest-order byte | | | | "C" | "B" | "A" | "0" | "4" | "3" |
| D202 lowest-order byte | | | | | "C" | "B" | "A" | "0" | "4" |
| D202 highest-order byte | | | | | | "C" | "B" | "A" | "0" |
| D203 lowest-order byte | | | Do not change | | | | "C" | "B" | "A" |
| D203 highest-order byte | | | | | | | | "C" | "B" |
| D204 lowest-order byte | | | | | | | | | "C" |

■Bit configuration when (n) is K4

D100 = 0ABCH

| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | A | | | | B | | | | C | | | |

D200

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| "A" → 41H | | | | | | | | "0" → 30H | | | | | | | |

D201

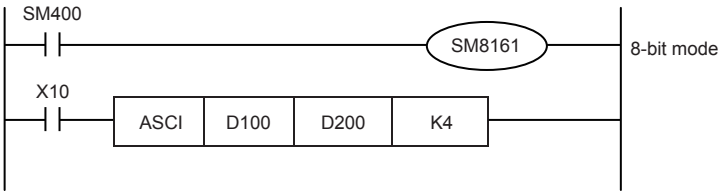
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| "C" → 43H | | | | | | | | "B" → 42H | | | | | | | |

ASCII code

| | | |
|---|---|---|
| "0" = 30H | "1" = 31H | "5" = 35H |
| "A" = 41H | "2" = 32H | "6" = 36H |
| "B" = 42H | "3" = 33H | "7" = 37H |
| "C" = 43H | "4" = 34H | "8" = 38H |

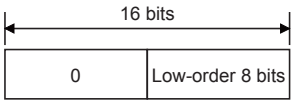
• 8-bit conversion mode (while SM8161 is ON)

Each digit of hexadecimal data stored in the device specified by (s) and later is converted into ASCII code, and transferred to the lower 8 bits of each device specified by (d) and later. SM8161 must always be on in the 8-bit conversion mode.
In the following program, conversion is executed as follows:




Devices specified by (s) and later
(D100)=0ABCH
(D101)=1234H
(D102)=5678H

When SM8161 is set to on, the 8-bit mode is selected.
The conversion processing is executed as follows.




Destination

■Number of specified digits (characters) and conversion result

| (n) / (d) | K1 | K2 | K3 | K4 | K5 | K6 | K7 | K8 | K9 |
|---|---|---|---|---|---|---|---|---|---|
| D200 | "C" | "B" | "A" | "0" | "4" | "3" | "2" | "1" | "8" |
| D201 | | "C" | "B" | "A" | "0" | "4" | "3" | "2" | "1" |
| D202 | | | "C" | "B" | "A" | "0" | "4" | "3" | "2" |
| D203 | | | | "C" | "B" | "A" | "0" | "4" | "3" |
| D204 | | | | | "C" | "B" | "A" | "0" | "4" |
| D205 | | | | | | "C" | "B" | "A" | "0" |
| D206 | | | Do not change | | | | "C" | "B" | "A" |
| D207 | | | | | | | | "C" | "B" |
| D208 | | | | | | | | | "C" |

■Bit configuration when (n) is K2

D100 = 0ABCH

| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | A | | | | B | | | | C | | | |

D200 = ASCII code of B = 42H

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 4 | | | | 2 | | | |

D201 = ASCII code of C = 34H

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 4 | | | | 3 | | | |

ASCII code

| | | |
|---|---|---|
| "0" = 30H | "1" = 31H | "5" = 35H |
| "A" = 41H | "2" = 32H | "6" = 36H |
| "B" = 42H | "3" = 33H | "7" = 37H |
| "C" = 43H | "4" = 34H | "8" = 38H |

## Precautions

- When outputting data in the BCD format for a printer, for example, it is necessary to convert binary data into BCD data before executing the ASCI(P) instructions.
- Whether NULL (00H) is stored after the last character or not depends on the ON/OFF status of the output character number selector signal SM701. When SM701 is off, NULL (00H) is stored. When SM701 is on, the original data remains.
- Depending on the ON/OFF status of SM701 and SM8161, the number of devices occupied by (d) differs.

| SM701 | SM8161 | Number of devices occupied by (d) |
|---|---|---|
| ON | ON | Number of letters |
| ON | OFF | Number of letters ÷ 2 |
| OFF | ON | Number of letters ÷ 1 |
| OFF | OFF | (Number of letters ÷ 2) + 1 |

- When RS2, HEX, or CCD is used, the extension flag SM8161 is common to other instructions. When using an instruction described above and the ASCI(P) instructions in the same program, make sure to set SM8161 to ON or OFF just before each instruction so that SM8161 does not apply to another instruction.

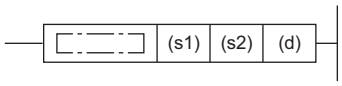
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (s) or (d) exceeds the corresponding device range. |
| 3405H | The value specified by (s) is other than any of 1 to 32767. |

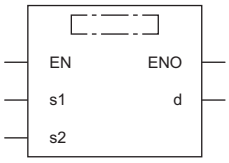
# Converting 16-bit binary data to character string

## STR(P)(_U)

These instructions add a decimal point to the 16-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=STR(EN,s1,s2,d);<br>ENO:=STRP(EN,s1,s2,d); | ENO:=STR_U(EN,s1,s2,d);<br>ENO:=STRP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | STR(P) | Head device number where the number of digits of the conversion target data is stored | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | STR(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |
| (s2) | STR(P) | Conversion target data | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | STR(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | | Head device number for storing the converted data | — | Character string | ANYSTRING_SINGLE |

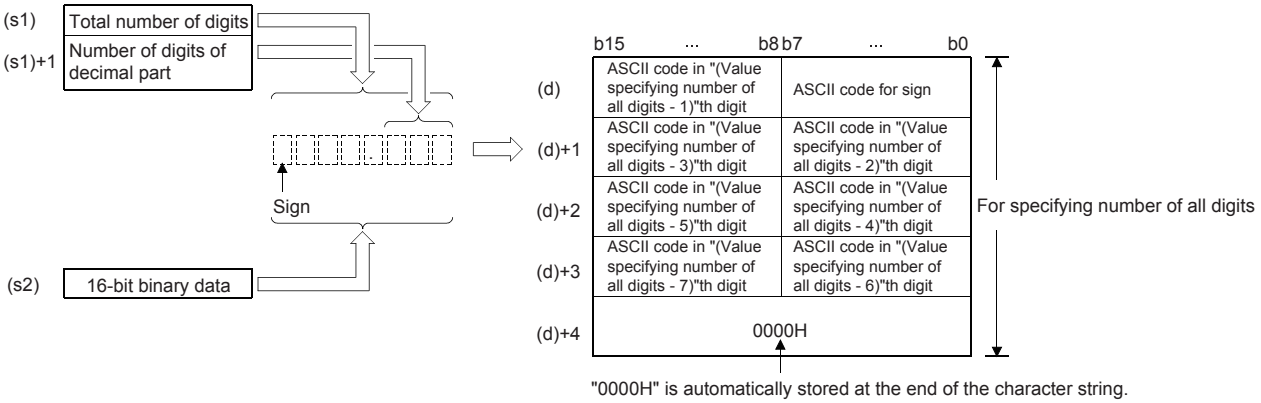
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

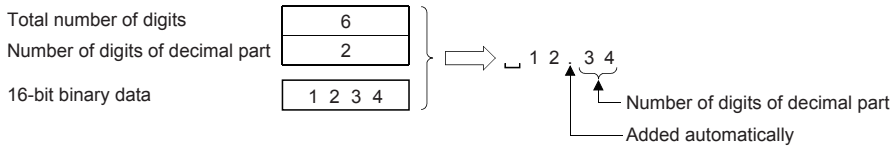
### Processing details

- These instructions add a decimal point to the 16-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.

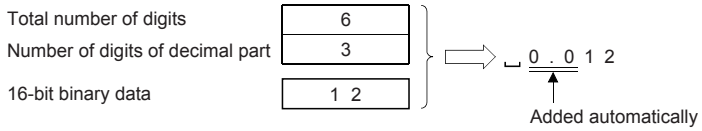



- The total number of digits that can be specified by (s1) is 2 to 8.

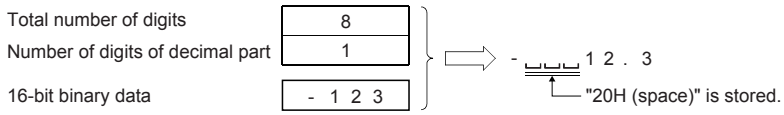
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 5. Note that the number of digits in the decimal part must be smaller than or equal to the total number of digits minus 3.
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the 16-bit binary data is positive, and "2DH" (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.




  - If the specified number of digits in the decimal part is greater than the number of digits of the 16-bit binary data, 0(s) is automatically added and the data is regarded as "0.□□□□".




  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 16-bit binary data, "20H" (space) is stored between the sign and the numeric value. If the number of digits of the 16-bit binary data is greater, an error occurs.




  - The value "00H" is automatically stored at the end of the converted character string.
  - When the number of all digits is even, "0000H" is stored in the device after the last character. When the number of all digits is odd, "00H" is stored in the upper byte (8 bits) of the device storing the final character.

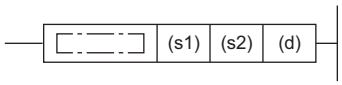
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 16-bit binary data in the device specified by (s2). (The additional 2 digits indicate the sign (+/-) and the decimal point.) |
| | The total number of digits specified by (s1) is out of the valid range (2 to 8). |
| | The number of digits in the decimal part specified by (s1)+1 is out of the valid range (0 to 5). |
| | The relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following.<br>(Total number of digits)-3 ≥ Number of digits in the decimal part |
| 3406H | The device areas storing the character string specified by (d) exceed the corresponding device range. |
| 2820H | The device range specified by (s1) exceeds the corresponding device range. |

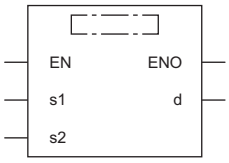
# Converting 32-bit binary data to character string

## DSTR(P)(_U)

These instructions add a decimal point to the 32-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DSTR(EN,s1,s2,d);<br>ENO:=DSTRP(EN,s1,s2,d); | ENO:=DSTR_U(EN,s1,s2,d);<br>ENO:=DSTRP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DSTR(P) | Head device number where the number of digits of the conversion target data is stored | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 2) |
| | DSTR(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 2) |
| (s2) | DSTR(P) | Conversion target data | -2147483648 to +2147483647 | 16-bit signed binary | ANY32_S |
| | DSTR(P)_U | | 0 to 4294967295 | 16-bit unsigned binary | ANY32_U |
| (d) | | Head device number for storing the converted data | — | Character string | ANYSTRING_SINGLE |

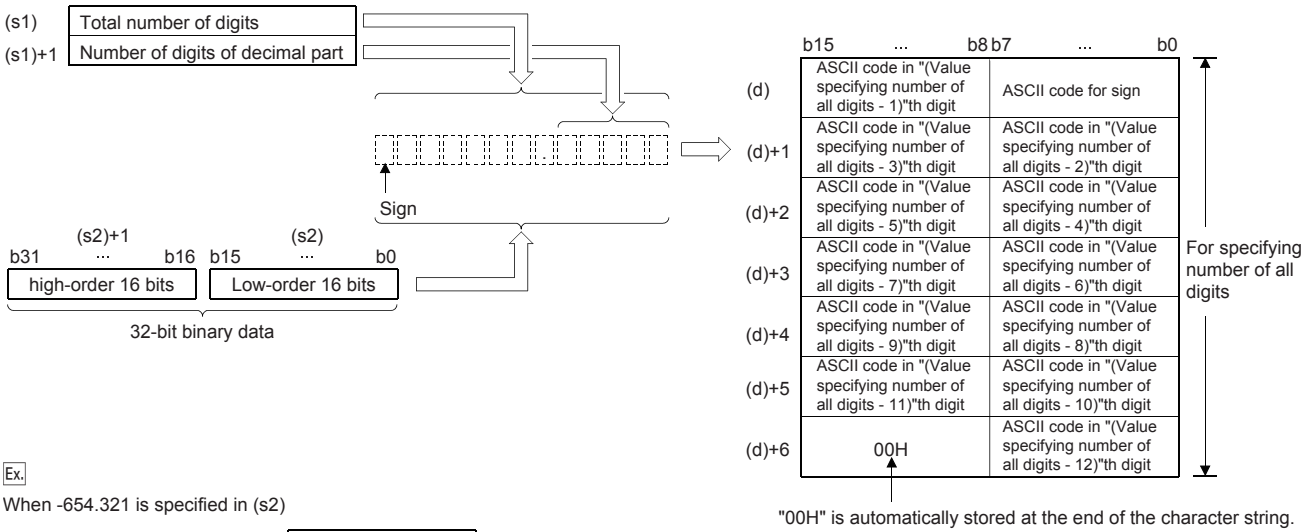
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

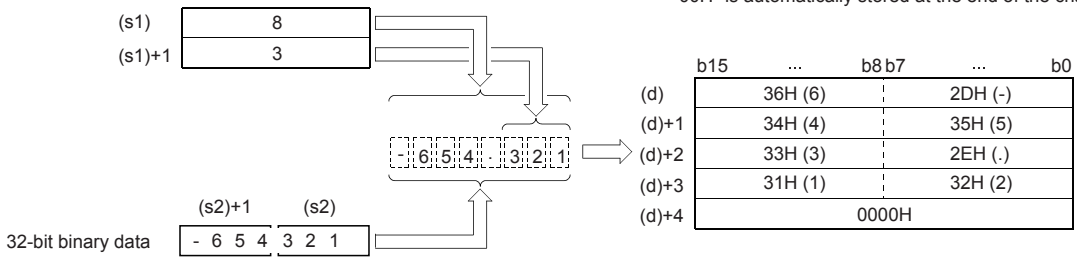
## Processing details

- These instructions add a decimal point to the 32-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.

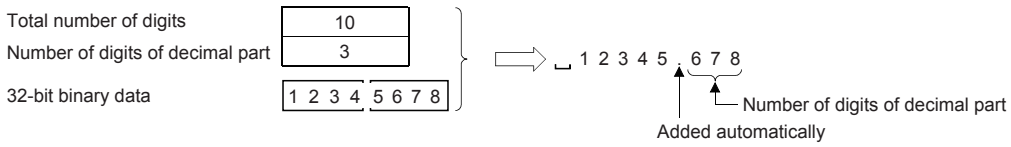



Ex.

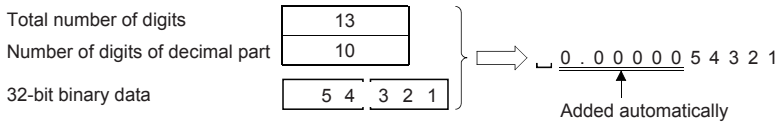
When -654.321 is specified in (s2)



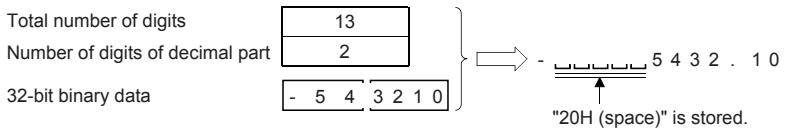

- The total number of digits that can be specified by (s1) is 2 to 13.
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 10. Note that the number of digits in the decimal part must be smaller than or equal to the total number of digits minus 3.
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the 32-bit binary data is positive, and "2DH" (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.




  - If the specified number of digits in the decimal part is greater than the number of digits of the 32-bit binary data, 0(s) is automatically added and the data is regarded as "0.□□□□".




  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 32-bit binary data, "20H" (space) is stored between the sign and the numeric value. If the number of digits of the 32-bit binary data is greater, an error occurs.




  - The value "00H" is automatically stored at the end of the converted character string.
  - When the number of all digits is even, "0000H" is stored in the device after the last character. When the number of all digits is odd, "00H" is stored in the upper byte (8 bits) of the device storing the final character.

## Operation error

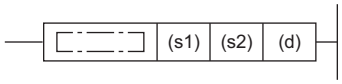
| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 16-bit binary data in the device specified by (s2). (The additional 2 digits indicate the sign (+/-) and the decimal point.) |
| | The total number of digits specified by (s1) is out of the valid range (2 to 13). |
| | The number of digits in the decimal part specified by (s1)+1 is out of the valid range (0 to 10). |
| | The relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following.<br>(Total number of digits)-3 ≥ Number of digits in the decimal part |
| 3406H | The device areas storing the character string specified by (d) exceed the corresponding device range. |
| 2820H | The device range specified by (s1) exceeds the corresponding device range. |

# Converting single-precision real number to character string

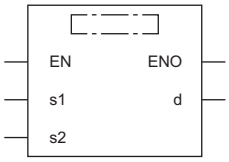
## ESTR(P)/DESTR(P)

These instructions convert the single-precision real number data stored in the device specified by (s1) into a character string according to the display specification stored in the device specified by (s2) and later, and store the string in the device specified by (d) and later.

The ESTR(P) instructions can also be used as DESTR(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ESTR(EN,s1,s2,d);<br>ENO:=ESTRP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Single-precision real number data to be converted or the start number of the device where data is stored | 0, $2^{-126}$<\|(s1)\|<$2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Head device number storing the display specification of a numeric value to be converted<br>The device specified in (s2) and following 2 devices are used. | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 3) |
| (d) | Head device number for storing the converted data | — | Character string | ANYSTRING_SINGLE |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

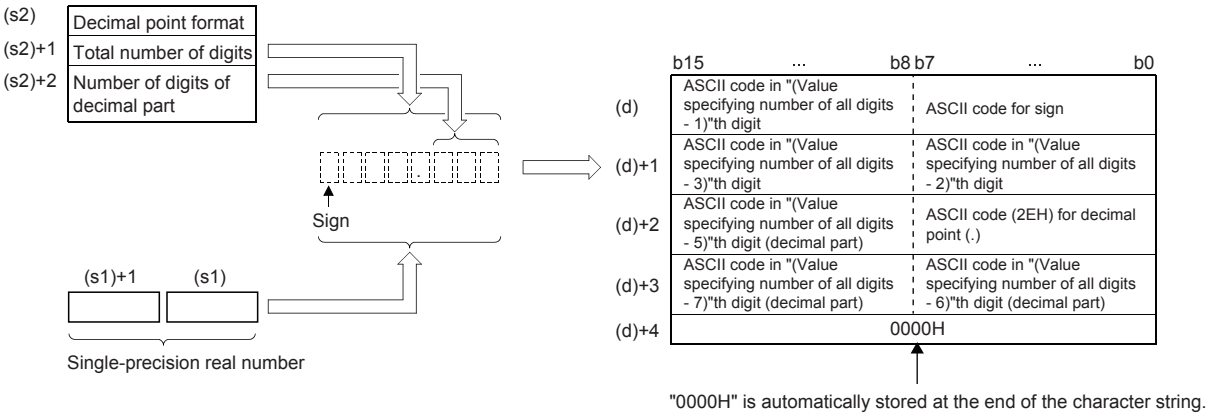
- These instructions convert the single-precision real number data stored in the device specified by (s1) into a character string according to the display specification stored in the device specified by (s2) and later, and store the string in the device specified by (d) and later. A real number can be directly specified as (s1).
- The data after conversion varies depending on the display specification stored in (s2).

| | |
|---|---|
| (s2) | 0: Decimal part format<br>1: Exponent format |
| (s2)+1 | Total number of digits |
| (s2)+2 | Number of digits of decimal part |

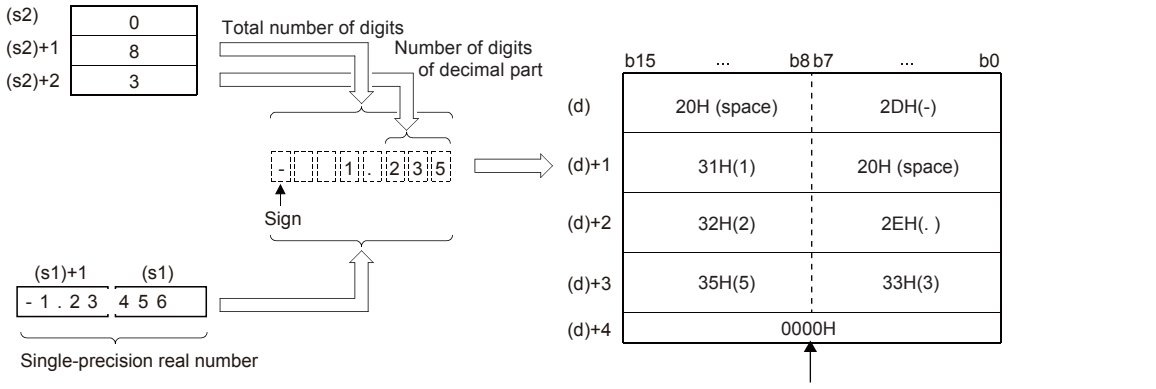
··· 2 to 24 can be set.

## ■Decimal point format

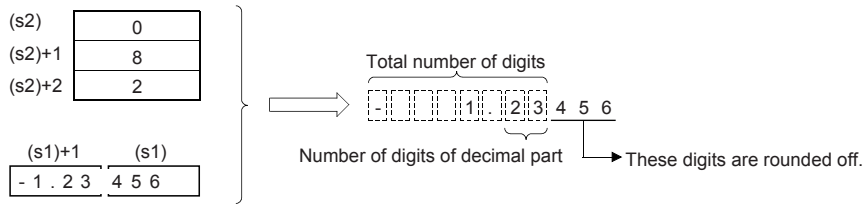
- When 0 is specified in (s2), the decimal point format is applied.



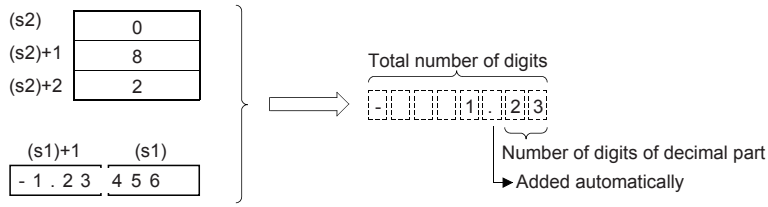

- When the number of decimal part digits is 0, the number of digits that can be specified by (s2)+1 is "the number of digits (24 at a maximum) ≥ 2". For other than 0, the number of digits that can be specified by (s2)+1 is "the number of digits (24 at a maximum) ≥ (the number of decimal point digits + 3)".
- The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.
- For example, when the total number of digits is "8", the number of digits of the decimal part is "3", and "-1.23456" is specified, data is stored in (d) and later as shown below:



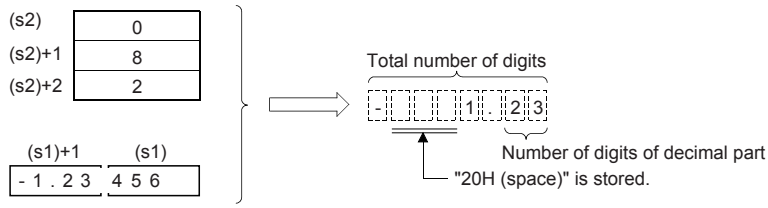
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the single-precision real number is positive, and "2DH" (-) is stored if the data is negative.
  - If the decimal part of the single-precision real number data cannot be accommodated in the number of digits of the decimal part, lower digits of the decimal part are rounded off.




  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.



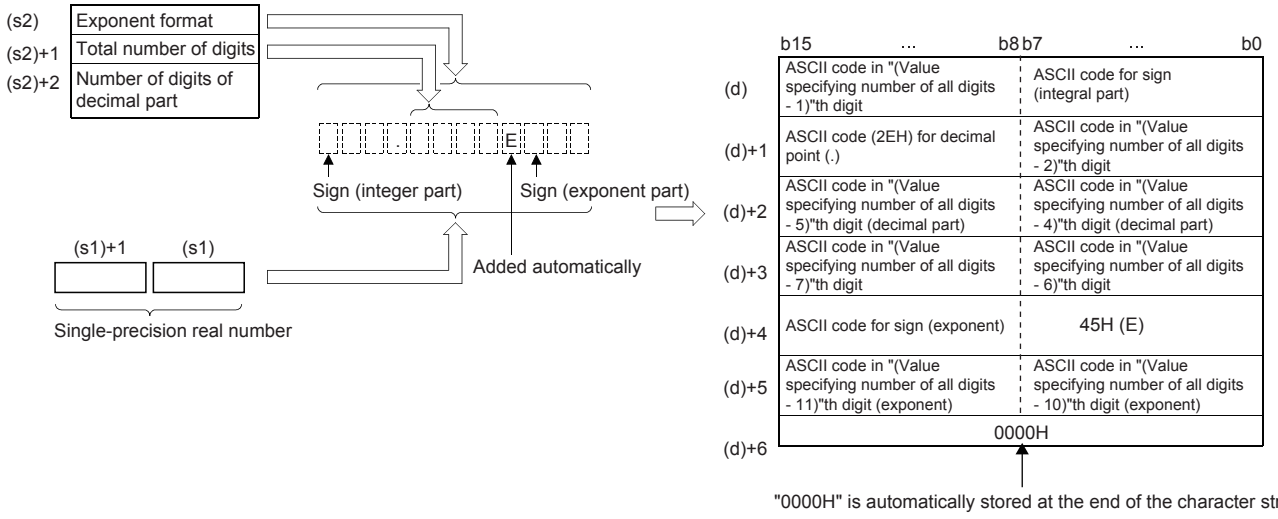

  - When the total number of digits subtracted by the digits for sign, decimal point, and decimal part is larger than the integer part of the single-precision real number data, "20H (space)" is stored between the sign and the integer part.




  - The value "00H" is automatically stored at the end of the converted character string.

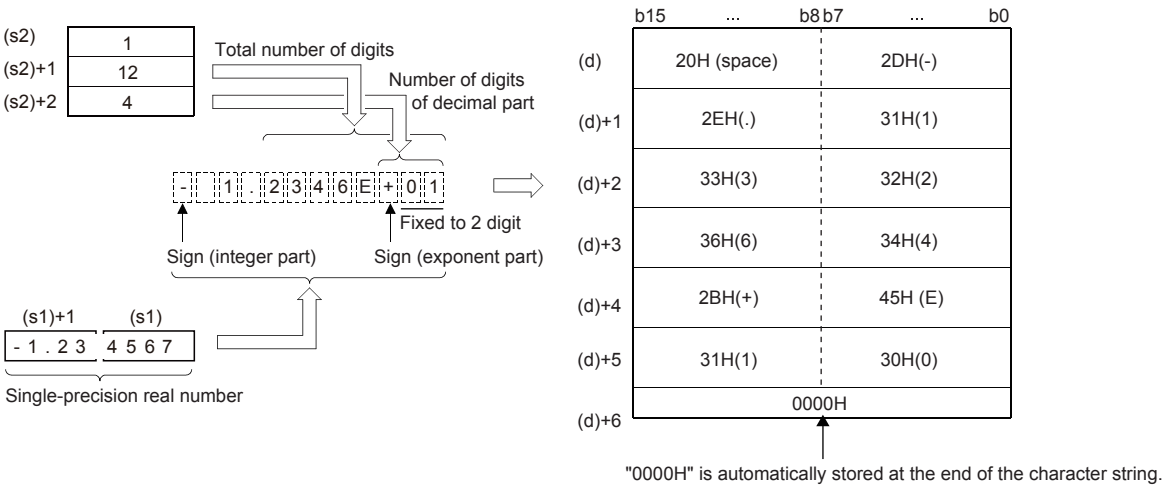
## ■Exponent format

- When 1 is specified in (s2), the exponent format is applied.




- When the number of decimal part digits is 0, the number of digits that can be specified by (s2)+1 is "the number of digits (24 at a maximum) ≥ 6". For other than 0, the number of digits that can be specified by (s2)+1 is "the number of digits (24 at a maximum) > (the number of decimal point digits + 7)".
- The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be equal to or smaller than the total number of digits minus 7.

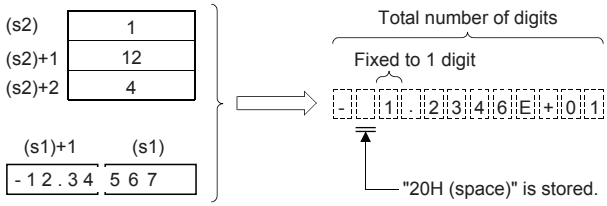
• For example, when the total number of digits is "12", the number of digits of the decimal part is "4", and "-12.34567" is specified, data is stored in (d) and later as shown below:



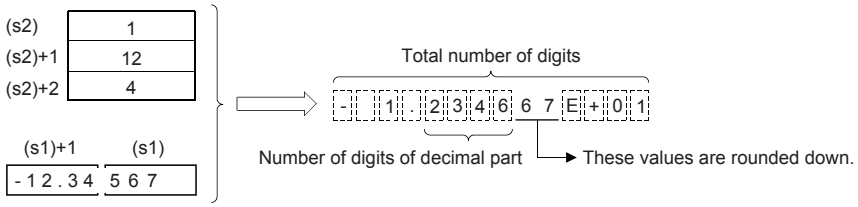

• The converted character string data are stored in the device areas specified by (d) and later as shown below.

  • As sign data of the integral part, "20H" (space) is stored if the single-precision real number is positive, and "2DH" (-) is stored if the data is negative.

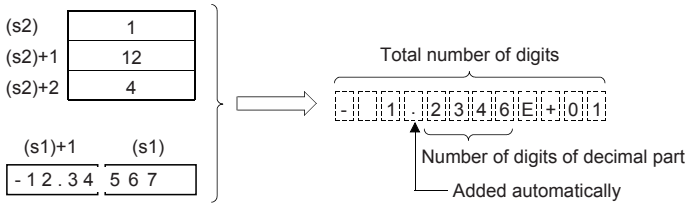
  • The integer part is fixed to 1 digit. "20H (space)" is stored between the integer part and the sign.




  • If the decimal part of the single-precision real number data cannot be accommodated in the number of digits of the decimal part, lower digits of the decimal part are rounded.

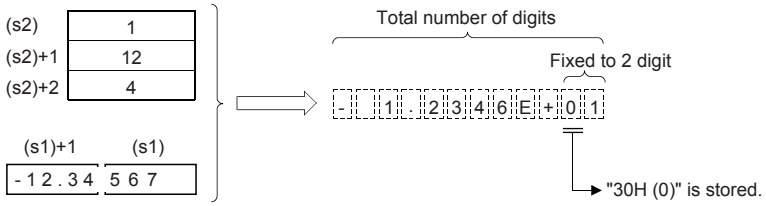



  • If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.




  • For the sign of the exponent part, "2BH (+)" is stored when the exponent is positive, and "2DH (-)" is stored when the exponent is negative.

  • The exponent part is fixed to 2 digits. When the exponent part is 1 digit, "30H (0)" is stored after the sign of the exponent part.




  • The value "00H" is automatically stored at the end of the converted character string.

## Operation error

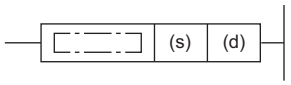
| Error code (SD0/SD3067) | Description |
|---|---|
| 2820H | The device specified by (s2) exceeds the corresponding device range. |
| 3401H | The number of total digits specified by (s1)+1 exceeds 24. |
| | The format specified by (s2) is any value other than "0" or "1". |
| | The total number of digits specified by (s2)+1 is not within the following range in the decimal point format.<br>When the number of digits of the decimal part is "0": Total number of digits ≥ 2<br>When the number of digits of the decimal part is any value other than "0": Total number of digits ≥ (Number of digits of decimal part + 3) |
| | The total number of digits specified by (s2)+1 is not within the following range in the exponent format.<br>When the number of digits of the decimal part is "0": Total number of digits ≥ 6<br>When the number of digits of the decimal part is any value other than "0": Total number of digits ≥ (Number of digits of decimal part + +7) |
| | The number of digits of the decimal part specified by (s2)+2 is not within the following range.<br>In the decimal part format ≤ (Total number of digits - 3)<br>In the exponent format ≤ (Total number of digits - 7) |
| | When the conversion result exceeds the specified total number of digits |
| 3402H | (s1) is not within the following range<br>0, $\pm 2^{-126} \leq (s1) < \pm 2^{128}$ |
| | The specified device value is denormalized number, NaN (not a number), or $\pm\infty$. |
| 3405H | The number of digits of the decimal part specified by (s2)+2 is not within the following range.<br>0 to 7 |
| 3406H | The device areas that store the character string specified by (d) exceed the corresponding device range. |

# Detecting a character string length

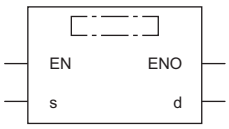
## LEN(P)

These instructions detect the length of the character string specified by (s), and store the length in the device specified by (d) and later.

These instructions handle data stored in the device specified by (s) to the device storing 00H as a character string.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=LENP(EN,s,d); |

**FBD/LD*1**




*1 The LEN instruction is not supported by the ST language and the FBD/LD language. Use LEN of the standard function.
☞ Page 909 LEN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Character string or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Device number storing the detected character string length | — | 16-bit signed binary | ANY16 |

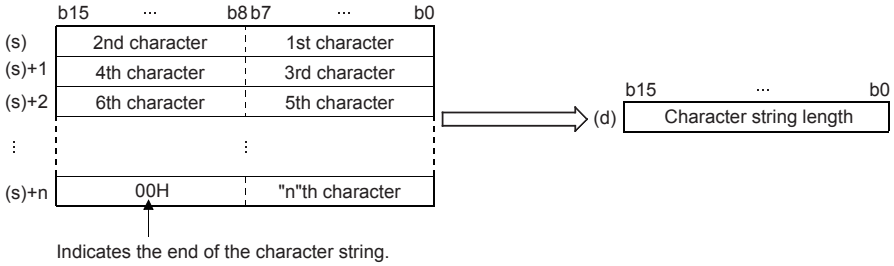
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

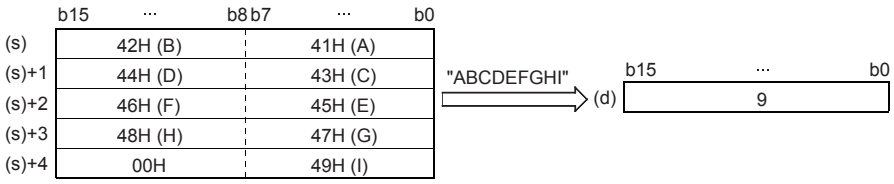
### Processing details

- These instructions detect the length of the character string specified by (s), and store the length in the device specified by (d) and later.
- These instructions handle data stored in the device specified by (s) to the device storing 00H as a character string.



Ex.

When "ABCDEFGHI" is stored in (s) and later




## Precautions

The LEN(P) instructions can handle character codes other than ASCII codes, but the character string length is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, the length of 1 character is detected as "2".

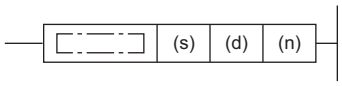
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |

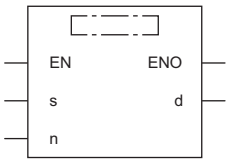
# Extracting character string data from the right

## RIGHT(P)

These instructions extract "n" characters of the character string data stored in the device specified by (s) and later from the right end (from the end), and store the extracted characters in the device specified by (d) and later.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=RIGHTP(EN,s,n,d); |

**FBD/LD*1**




*1 The RIGHT instruction is not supported by the ST language and the FBD/LD language. Use RIGHT of the standard function.
☞ Page 910 LEFT(_E), RIGHT(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Character string or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number for storing "n" characters extracted from the right of the device specified by (s) | — | Character string | ANYSTRING_SINGLE |
| (n) | Number of characters to be extracted | 1 to 16383 | 16-bit signed binary | ANY16 |

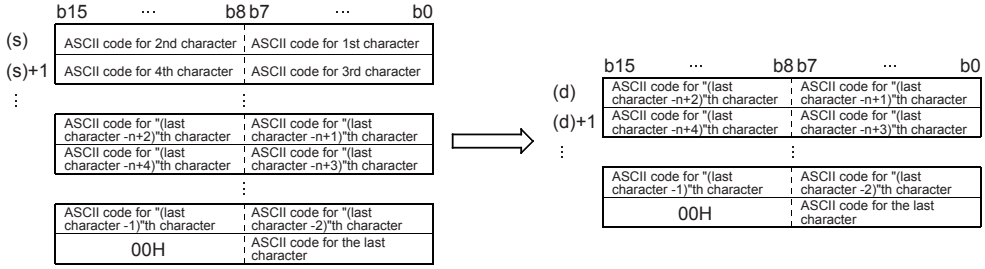
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

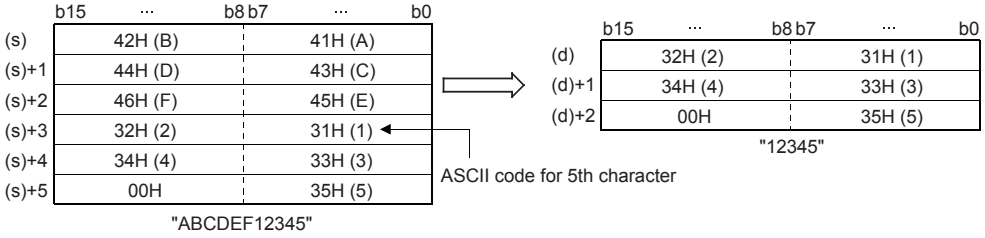
- These instructions extract "n" characters of the character string data stored in the device specified by (s) and later from the right end (from the end), and store the extracted characters in the device specified by (d) and later.




- A character string stored in (s) indicates data stored in devices from the specified device until "00H" is first detected in units of 1 byte.

Ex.

When 5 is specified in (n)




- A NULL code (00H), which indicates an end of a character string, is automatically added at the end of the character string data.
- When the number of extracted characters is odd, "00H" is stored in the upper byte of a device storing the last character. When the number of extracted characters is even, "0000H" is stored in the device after the last character.
- When the number of characters specified by (n) is 0, a NULL code (00H) is stored in (d).

## Precautions

When handling character codes other than ASCII codes, note the following points:

- The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, 1 character is detected as "2".
- When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character. Note that the expected character code is not retrieved if only 1 byte is extracted out of a 2-byte character code.

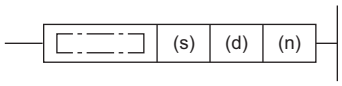
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | (n) is not within the following range<br>0 to 16383 |
| | The character string specified by (s) has more than 16383 characters. |
| | "n" exceeds the number of characters specified by (s) |
| 3406H | The (n) points of data in the device starting from the one specified by (d) exceed the corresponding device range. |

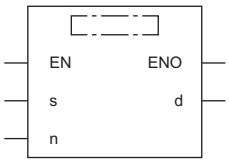
# Extracting character string data from the left

## LEFT(P)

These instructions extract "n" characters of the character string data stored in the device specified by (s) and later from the left end (from the start), and store the extracted characters in the device specified by (d) and later.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=LEFTP(EN,s,n,d); |

**FBD/LD*1**




*1 The LEFT instruction is not supported by the ST language and the FBD/LD language. Use LEFT of the standard function.
☞ Page 910 LEFT(_E), RIGHT(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Character string or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number for storing "n" characters extracted from the left of the device specified by (s) | — | Character string | ANYSTRING_SINGLE |
| (n) | Number of characters to be extracted | 1 to 16383 | 16-bit signed binary | ANY16 |

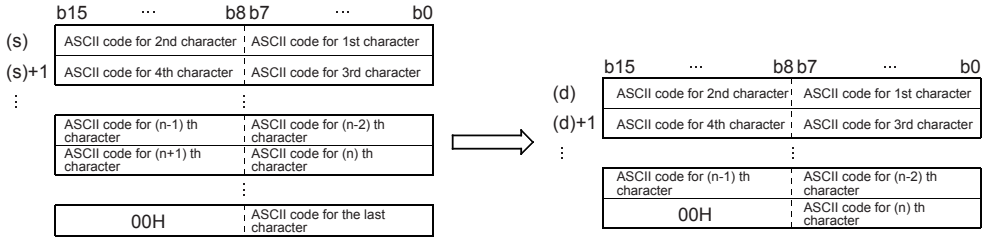
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

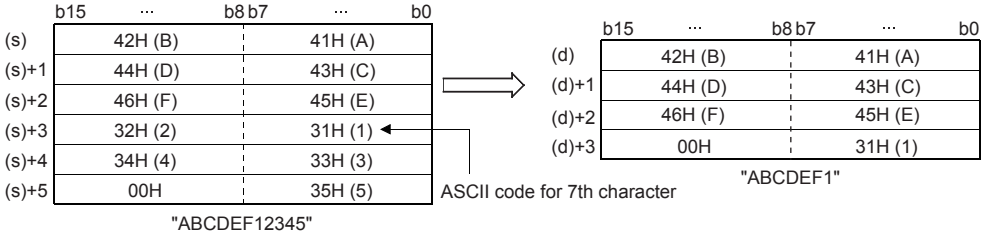
- These instructions extract "n" characters of the character string data stored in the device specified by (s) and later from the left end (from the start), and store the extracted characters in the device specified by (d) and later.




- A character string stored in (s) indicates data stored in devices from the specified device until "00H" is first detected in units of 1 byte.

Ex.

When 7 is specified in (n)




- A NULL code (00H), which indicates an end of a character string, is automatically added at the end of the character string data.
- When the number of extracted characters is odd, "00H" is stored in the upper byte of a device storing the last character. When the number of extracted characters is even, "0000H" is stored in the device after the last character.
- When the number of characters specified by (n) is 0, a NULL code (00H) is stored in (d).

## Precautions

When handling character codes other than ASCII codes, note the following points:

- The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, 1 character is detected as "2".
- When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character. Note that the expected character code is not retrieved if only 1 byte is extracted out of a 2-byte character code.

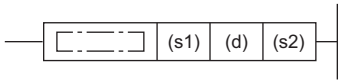
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | (n) is not within the following range<br>0 to 16383 |
| | The character string specified by (s) has more than 16383 characters. |
| | "n" exceeds the number of characters specified by (s) |
| 3406H | The (n) points of data in the device starting from the one specified by (d) exceed the corresponding device range. |

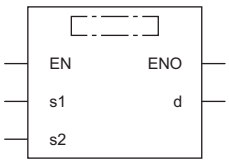
# Storing the specified number of character strings

## MIDR(P)

These instructions extract the number of characters specified by (s2)+1 of the character string data stored in the device specified by (s1) and later from the position specified by (s2), and store the extracted characters in the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MIDR(EN,s1,s2,d);<br>ENO:=MIDRP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Character string or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number for storing the character string data of the operation result | — | Character string | ANYSTRING_SINGLE |
| (s2) | Head device number for storing the number of characters and position of the start character<br>(s2)+0: Position of the start character, (s2)+1: Number of characters | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 2) |

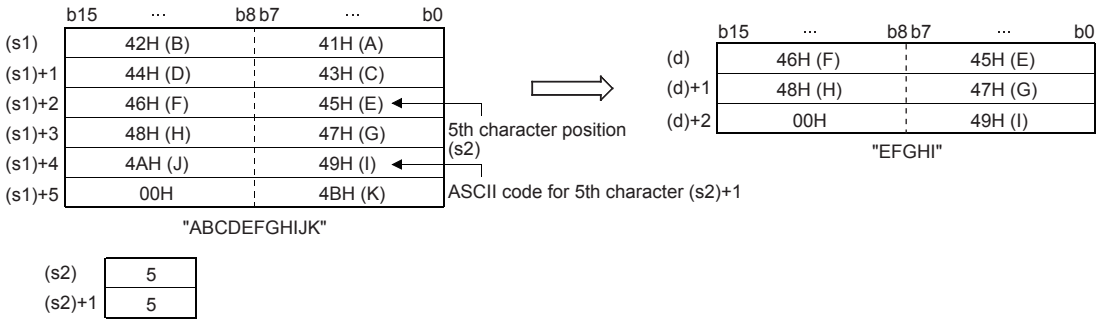
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

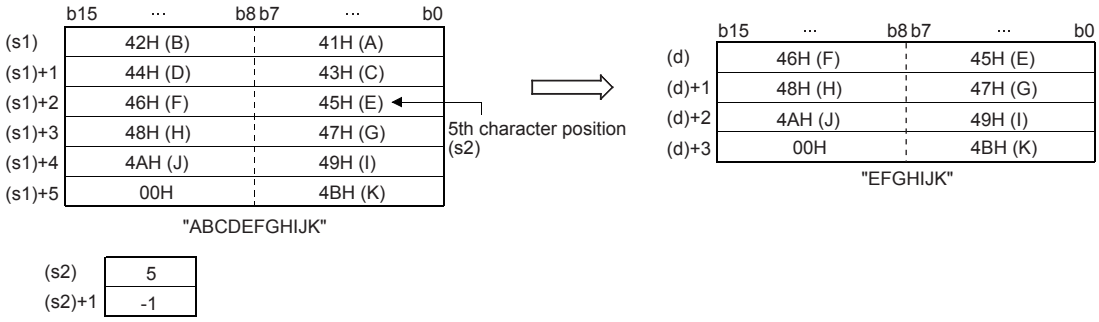
- These instructions extract the number of characters specified by (s2)+1 of the character string data stored in the device specified by (s1) and later from the position specified by (s2), and store the extracted characters in the device specified by (d) and later.




| | |
|---|---|
| (s2) | 5 |
| (s2)+1 | 5 |

- A character string stored in (s1) indicates data stored in devices from the specified device until "00H" is first detected in units of byte.

- A NULL code (00H), which indicates an end of a character string, is automatically added at the end of the character string data.
- When the number of extracted characters "(s2)+1" is odd, "00H" is stored in the upper byte of a device storing the last character. When the number of extracted characters "(s2)+1" is even, "0000H" is stored in the device after the last character.
- If the number of characters specified by (s2)+1 is 0, no processing is performed.
- When (s2)+1 (the number of characters to be extracted) is "-1", the entire character string stored in (s1) and later is stored to (d) and later.




## Precautions

When handling character codes other than ASCII codes, note the following points:

- The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, 1 character is detected as "2".
- When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character. Note that the expected character code is not retrieved if only 1 byte is extracted out of a 2-byte character code.

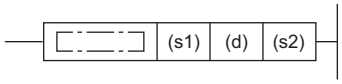
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) and later, "00H" does not exist. |
| 3405H | The value stored in a device specified in (s2)+1 is -2 or lower. |
| | The value stored in a device specified in (s2) exceeds the number of characters of (s1). |
| | A negative value is specified in (s2). |
| | The value stored in a device specified in (s2)+1 exceeds the number of characters of (s1). |
| | The character string specified by (s1) has more than 16383 characters. |
| | The total of the values stored in devices specified in (s2) and (s2)+1 exceeds the number of characters of (s1). |
| 3406H | The number of characters from the position specified by (d) to (s2)+1 exceeds the corresponding device range. |

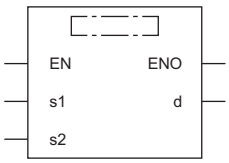
# Replacing the specified number of character strings

## MIDW(P)

These instructions extract the number of characters specified by (s2)+1 from the character string data stored in the device specified by (s1) and later, and store the extracted data in the position specified by (s2) and later of the character string data stored in the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MIDW(EN,s1,s2,d);<br>ENO:=MIDWP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Character string or head device number storing a character string | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number for storing the character string data of the operation result | — | Character string | ANYSTRING_SINGLE |
| (s2) | Head device number for storing the number of characters and position of the start character<br>(s2)+0: Position of the start character, (s2)+1: Number of characters | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 2) |

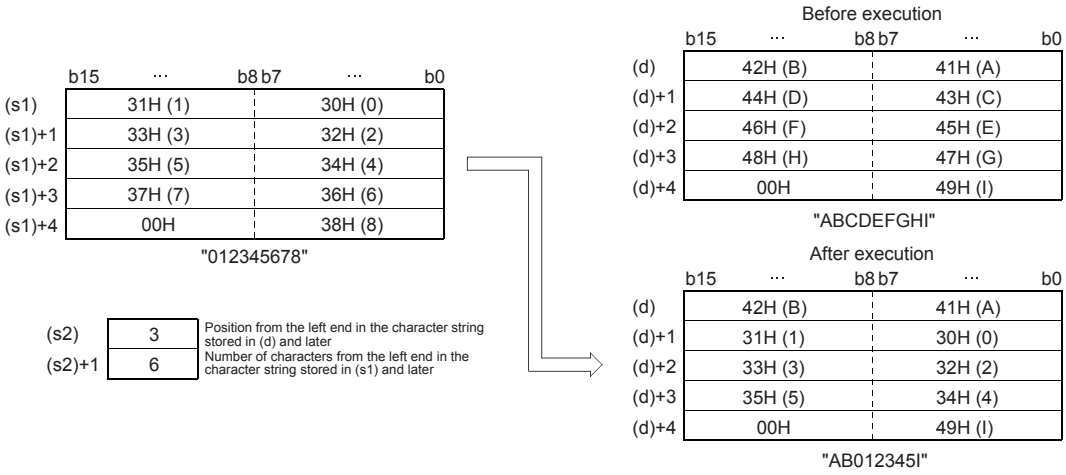
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

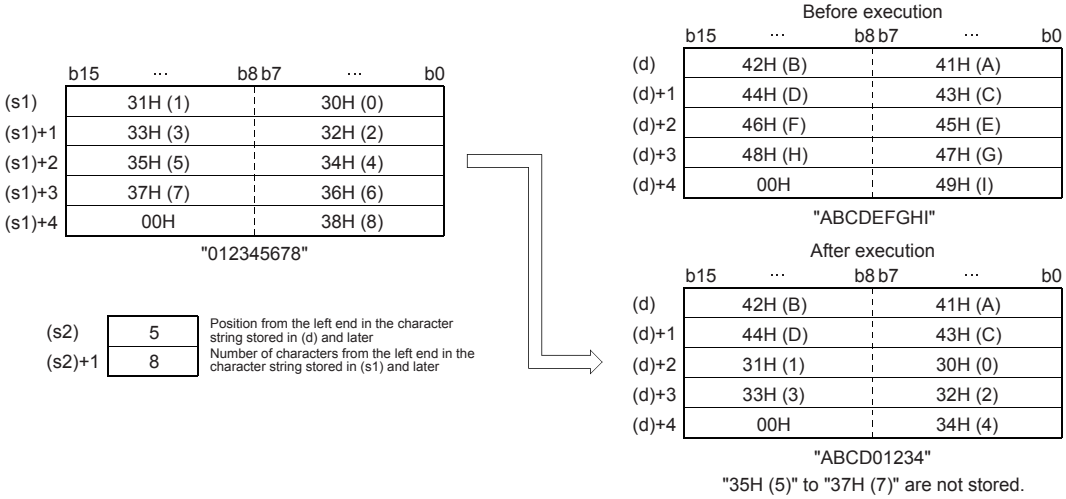
*1 T, ST, C cannot be used.

## Processing details

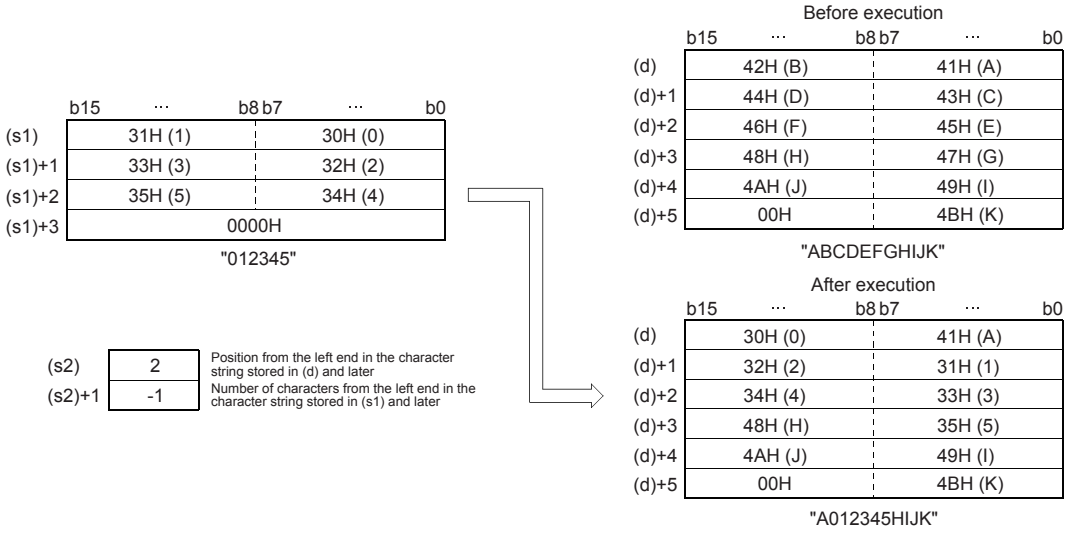
- These instructions extract the number of characters specified by (s2)+1 from the character string data stored in the device specified by (s1) and later, and store the extracted data in the position specified by (s2) and later of the character string data stored in the device specified by (d) and later.




- A character string stored in (s1) or (d) indicates data stored in devices from the specified device until "00H" is first detected in units of 1 byte.
- A NULL code (00H), which indicates an end of a character string, is automatically added at the end of the character string data.
- If the number of characters specified by (s2)+1 is 0, no processing is performed.
- When the number of characters specified by (s2)+1 exceeds the last character of the character string specified by (d), data is stored up to the last character of (d).



- When (s2)+1 (the number of characters to be extracted) is "-1", the entire character string stored in (s1) and later is stored to (d) and later.




## Precautions

When handling character codes other than ASCII codes, note the following points:

- The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, 1 character is detected as "2".
- When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character. Note that the expected character code is not retrieved if only 1 byte is extracted out of a 2-byte character code.

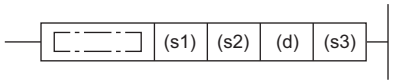
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) and later, "00H" does not exist. |
| | The device specified by (d) exceeds the corresponding device range. |
| 3405H | The value stored in a device specified in (s2)+1 is -2 or lower. |
| | The value stored in a device specified in (s2) exceeds the number of characters of (d). |
| | The value stored in a device specified in (s2)+1 exceeds the number of characters of (s1). |
| | The character string specified by (s1) has more than 16383 characters. |
| | The character string specified by (d) has more than 16383 characters. |

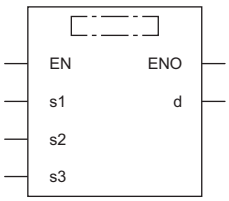
# Searching character string

## INSTR(P)

These instructions search the character string data stored in the device specified by (s2) and later starting from the (s3)th character from the left, for the character string data stored in the device specified by (s1) and later and store the search result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=INSTR(EN,s1,s2,s3,d);<br>ENO:=INSTRP(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Character string to be searched for or head device number storing a character string to be searched for | — | Character string | ANYSTRING_SINGLE |
| (s2) | Character string to be searched or head device number storing a character string to be searched | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number storing search result | — | 16-bit signed binary | ANY16 |
| (s3) | Search start position | 1 to 16383 | 16-bit signed binary | ANY16 |

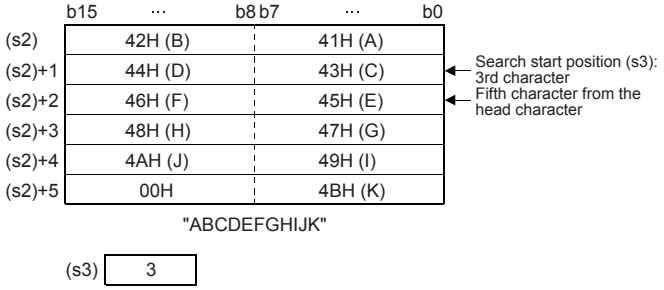
#### ■Applicable devices

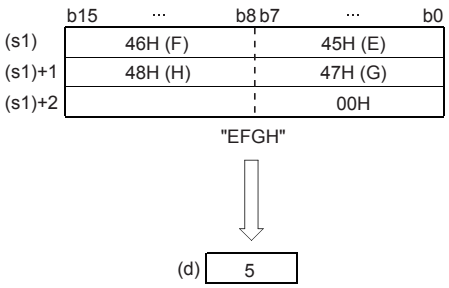
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (s2) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

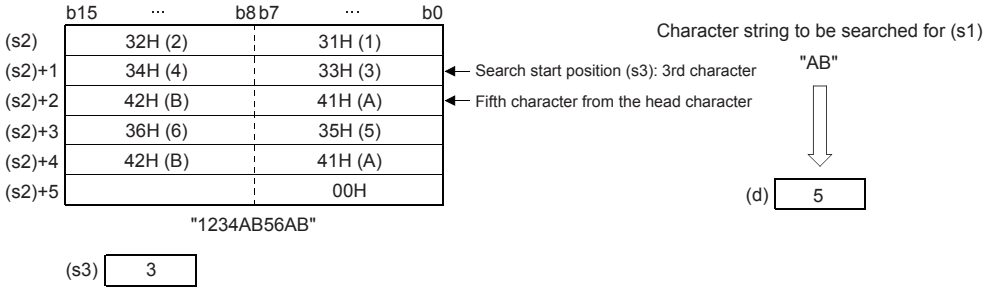
### Processing details

- These instructions search the character string data stored in the device specified by (s2) and later starting from the (s3)th character from the left, for the character string data stored in the device specified by (s1) and later and store the search result in the device specified by (d). The search result stores the position where the first detected character is located from the start character in the character string data stored in (s2).



- If no matched character string data is found, 0 is stored in (d).
- When the search start position "s3" is a negative number or "0", search processing is not executed.
- A character string can be directly specified in the character string (s1).



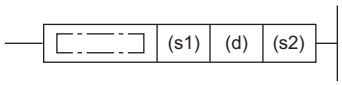

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | No NULL code (00H) exists in the corresponding device range of the device specified by (s1) and later. |
| | No NULL code (00H) exists in the corresponding device range of the device specified by (s2) and later. |
| 3405H | The value stored in a device specified in (s3) exceeds the number of characters of (s2). |
| | The character string specified by (s1) has more than 16383 characters. |
| | The character string specified by (s2) has more than 16383 characters. |

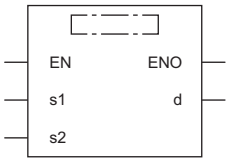
# Inserting character string

## STRINS(P)

These instructions insert the character string specified by (s1) at the (s2)th character from the start of the character string specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=STRINS(EN,s1,s2,d);<br>ENO:=STRINSP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Character string to be inserted or head device number storing the character string to be inserted | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number storing a character string to which another character string is inserted | — | Character string | ANYSTRING_SINGLE |
| (s2) | Insertion position (in units of bytes) | 1 to 16383 | 16-bit signed binary | ANY16 |

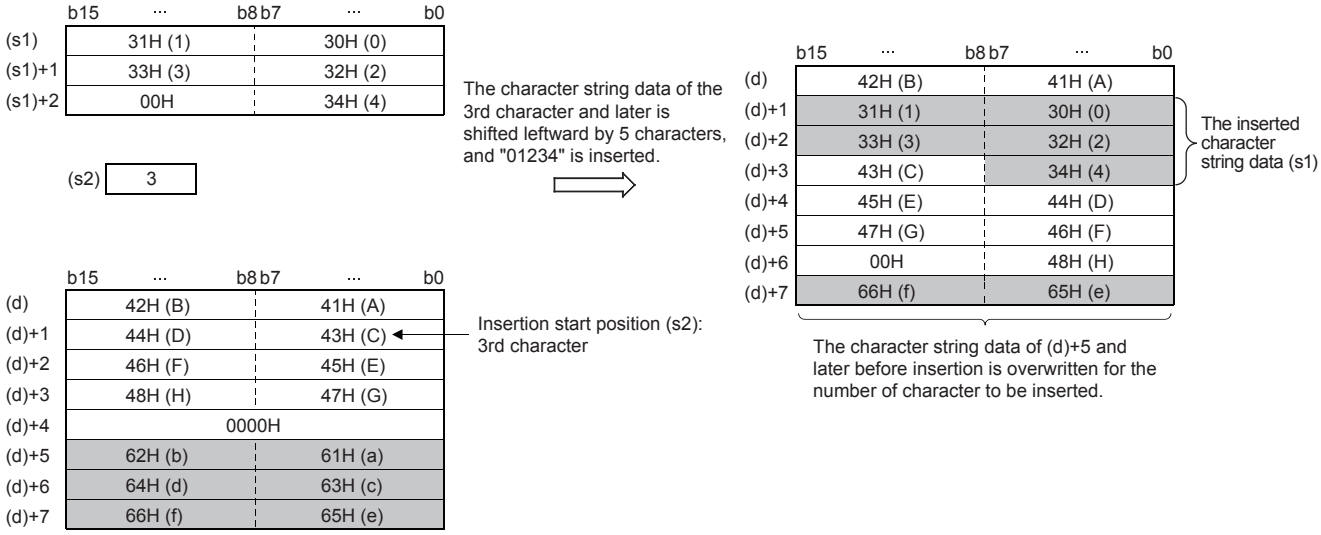
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

- These instructions insert the character string specified by (s1) at the (s2)th character from the start of the character string specified by (d).




- When the number of characters after insertion, (s1)+(d), is even, a NULL code (00H) is stored in the device (1 word) after the last device storing the character string.
- When the number of characters after insertion, (s1)+(d), is odd, a NULL code (00H) is stored in the last device (upper 8 bits) of the character string.
- If the number of characters exceeding (d) by one character is specified in (s2), the character string in (s1) is added to the end of the character string in (d).

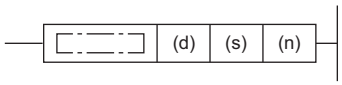
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | No NULL code (00H) exists in the corresponding device range of the device specified by (s1) and later. |
| | No NULL code (00H) exists in the corresponding device range of the device specified by (d) and later. |
| 2821H | A device of the character strings (s1) and (d) overlaps. |
| | The device storing the character string after insertion, (s1)+(d), overlaps with the character string-storing device of (s1). |
| 3405H | The character string specified by (s1) has more than 16383 characters. |
| | The character string specified by (d) has more than 16383 characters. |
| | (s2) is not within the range (1≤(s2)≤16383) |
| | The value specified by (s2) exceeds "the number of characters of the character string (d) + 1". |
| 3406H | The character string after insertion, (s1)+(d), has more than 16383 characters. |
| | The character string after insertion, (s1)+(d), exceeds the corresponding device range. |

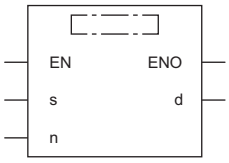
# Deleting character string

## STRDEL(P)

These instructions delete (n) characters starting from the (s)th character (deletion start position) from the start of the character string data specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=STRDEL(EN,s,n,d);<br>ENO:=STRDELP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number storing a character string having characters to be deleted | — | Character string | ANYSTRING_SINGLE |
| (s) | Deletion start position | 1 to 16383 | 16-bit signed binary | ANY16 |
| (n) | Number of characters to be deleted | 0 to 16384 | 16-bit signed binary | ANY16 |

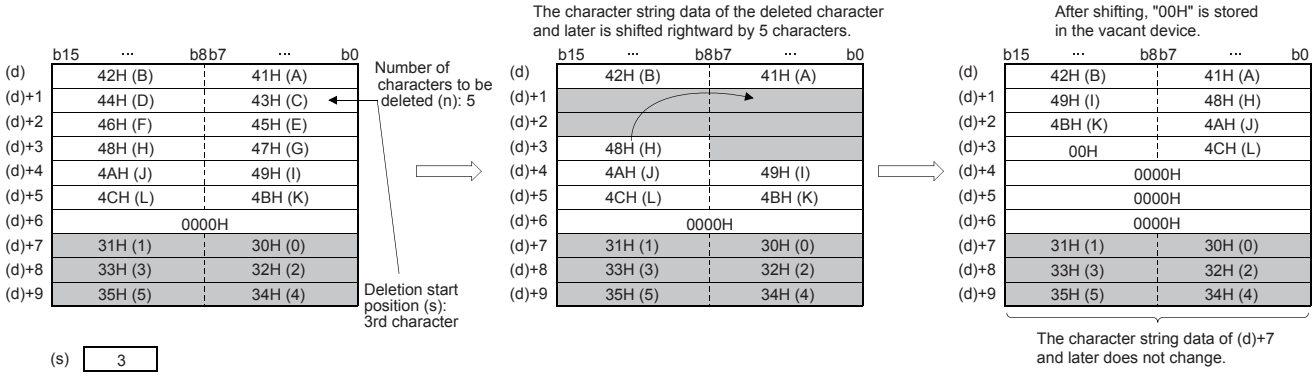
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

- These instructions delete (n) characters starting from the (s)th character (deletion start position) from the start of the character string data specified by (d).




- When the number of characters after deletion, (d), is even, a NULL code (00H) is stored in the device after the last device storing the character string.
- When the number of characters after deletion, (d), is odd, a NULL code (00H) is stored in the last device (upper 8 bits) of the character string.
- The character string after the deleted character string is shifted by (n) characters, a NULL code (00H) is stored in vacant devices.

## Operation error

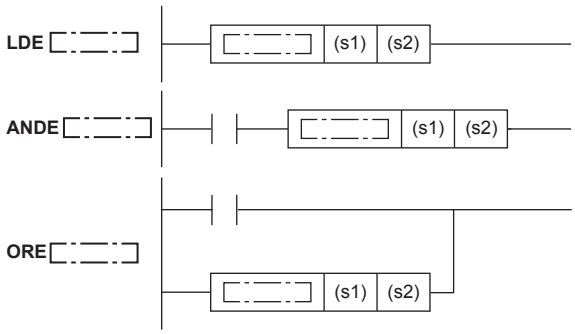
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | No NULL code (00H) exists in the corresponding device range of the device specified by (d) and later. |
| 3405H | The character string specified by (d) has more than 16383 characters. |
| | (s) is not within the range (1≤(s)≤16383) |
| | The value specified by (s) exceeds the number of characters of the character string (d). |
| | The value specified by (n) exceeds the number of characters from (s) to the last of the character string (d). |

# 8.7 Real Number Instruction

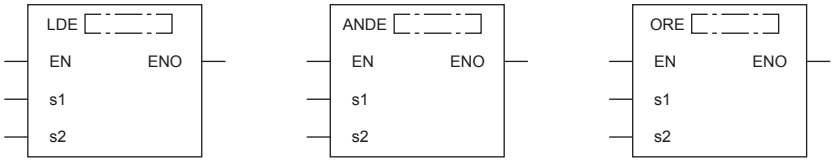
## Comparing single-precision real numbers

### LDE□, ANDE□, ORE□

These instructions perform a comparison operation between the single-precision real number in the device specified by (s1) and the single-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)

| Ladder diagram | Structured text |
|---|---|
|  ("=", "<>", ">", "<=", "<", ">=" enters □.) | Not supported |

**FBD/LD**




("_EQ", "_NE", "_GT", "_LE", "_LT", "_GE" enters □.)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Comparison data or the head device number where the comparison data is stored | 0, $2^{-126}$<\|(s1)\|<$2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Comparison data or the head device number where the comparison data is stored | 0, $2^{-126}$<\|(s2)\|<$2^{128}$ | Single-precision real number | ANYREAL_32 |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |

8

## Processing details

- These instructions perform a comparison operation between the single-precision real number in the device specified by (s1) and the single-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)
- The following table lists the comparison operation results of each instruction.

| Instruction symbol | Condition | Result | Instruction symbol | Condition | Result |
|---|---|---|---|---|---|
| E= | (s1)=(s2) | Conductive state | E= | (s1)≠(s2) | Non-conductive state |
| E<> | (s1)≠(s2) | | E<> | (s1)=(s2) | |
| E> | (s1)>(s2) | | E> | (s1)<(s2) | |
| E<= | (s1)<(s2) | | E<= | (s1)>(s2) | |
| E< | (s1)<(s2) | | E< | (s1)>(s2) | |
| E>= | (s1)>(s2) | | E>= | (s1)<(s2) | |

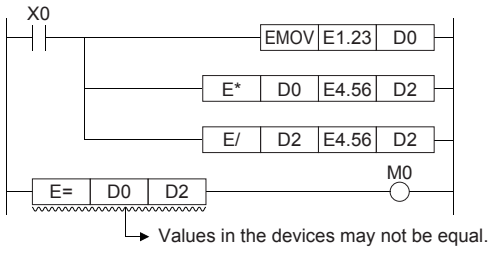
- When an input value is set from the engineering tool, a rounding error may occur.

## Operation error

There is no operation error.

**Point**

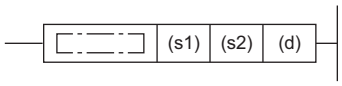
When the E= instruction is used, note that values in the devices may not be equal.



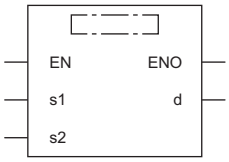
# Single-precision real number comparison

## DECMP(P)

These instructions compare two data values (single-precision real number), and output the result (larger, same or smaller) to three consecutive bit devices.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DECMP(EN,s1,s2,d);<br>ENO:=DECMPP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

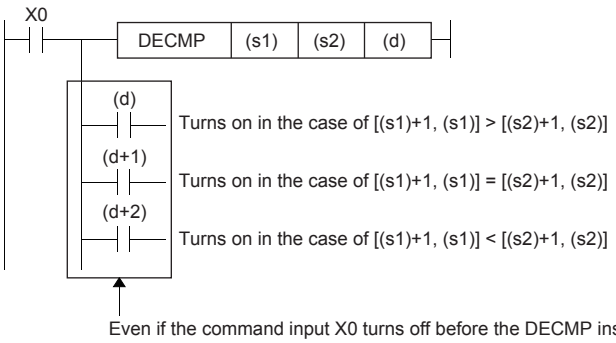
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Comparison data or the number of the device where the comparison data is stored | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Comparison data or the number of the device where the comparison data is stored | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Start bit device number to which comparison result is output (Three devices are occupied). | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

### Processing details

- These instructions compare the comparison value (s1) with the comparison source (s2) as floating point data, and one of the bits among (d), (d)+1, and (d)+2 turns on according to the result (smaller, same or larger).




- When the constant (K or H) is specified the device specified by (s1) and (s2), these instructions convert the binary value into single-precision real number automatically.

## Precautions

- Three devices ((d), (d)+1, and (d)+2) specified by (d) are occupied. Note that these devices are not used for any other purpose.

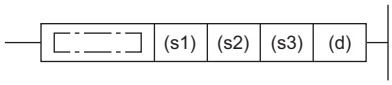
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |

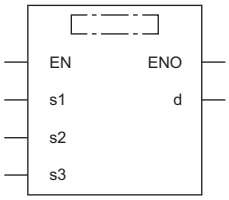
# Single-precision real number data band comparison

## DEZCP(P)

These instructions compare the comparison range of two points, upper and lower, with the binary floating point, and output the result to three consecutive bit devices in accordance with the larger, smaller, and band.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:= DEZCP (EN, s1, s2, s3, d);<br>ENO:= DEZCPP(EN, s1, s2, s3, d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

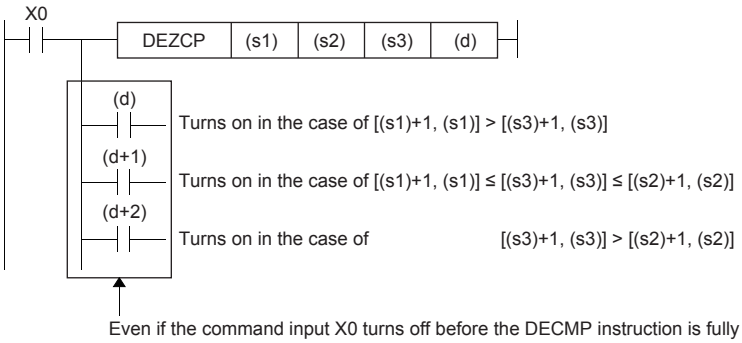
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Comparison data or the number of the device where the comparison data is stored | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Comparison data or the number of the device where the comparison data is stored | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s3) | Comparison data or the number of the device where the comparison data is stored | 0, $2^{-126} \leq \lvert(s3)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Start bit device number to which comparison result is output (Three devices are occupied). | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s3) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

### Processing details

- These instructions compare the comparison values (s1) and (s2) with the comparison source (s3) as floating point data, and one of the bits among (d), (d)+1, and (d)+2 turns on according to the result (smaller, within the range or larger).




- When the constant (K or H) is specified the device specified by (s1), (s2) and (s3), these instructions convert the binary

value into single-precision real number automatically.

## Precautions

- Three devices ((d), (d)+1, and (d)+2) specified by (d) are occupied. Note that these devices are not used for any other purpose.
- The size relationship of the comparison data should be [(s1)+1, (s1)] ≤ [(s2)+1, (s2)]. If the relationship is [(s1)+1, (s1)] > [(s2)+1, (s2)], the value of [(s2)+1, (s2)] is regarded as the same as that of [(s1)+1, (s1)], and is compared.

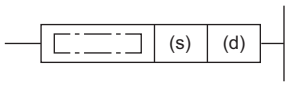
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |

# Adding single-precision real numbers

## E+(P) [For 2 operands]

These instructions add the single-precision real number in the device specified by (s) to the single-precision real number in the device specified by (d), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

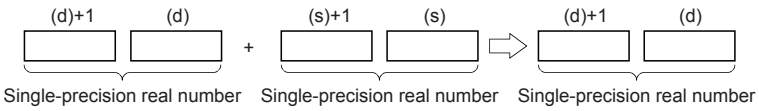
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Addend data or the head device number where the data that is added to another is stored | 0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number where the data to which another is added is stored | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions add the single-precision real number in the device specified by (s) to the single-precision real number in the device specified by (d), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s) and (d) should be 0 or $2^{-126} \leq$ |specified value (stored value)| $< 2^{128}$.
- When an input value is set from the engineering tool, a rounding error may occur.
- The table below shows the related devices.

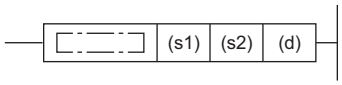
| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result $< 2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

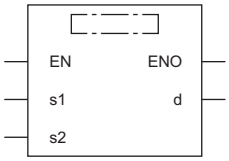
| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|Specified device value\| $< 2^{128}$ |

## E+(P) [For 3 operands]

These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=EPLUS(EN, s1, s2, d);<br>ENO:=EPLUSP(EN, s1, s2, d); |

**FBD/LD**




("EPLUS", "EPLUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

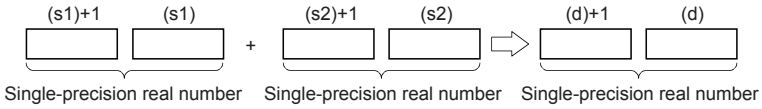
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Augend data or the head device number where the data to which another is added is stored | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Addend data or the head device number where the data that is added to another is stored | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s1), (s2), and (d) should be 0 or $2^{-126}$ |specified value (stored value)|$\leq 2^{128}$.
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

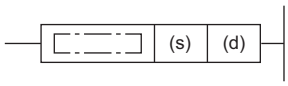
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

# Subtracting single-precision real numbers

## E-(P) [For 2 operands]

These instructions subtract the single-precision real number in the device specified by (s) from the single-precision real number in the device specified by (d), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

| FBD/LD |
|---|
| Not supported. |

### Setting data

#### ■Descriptions, ranges, and data types

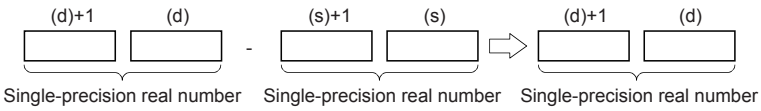
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Subtrahend data or the head device number where the data to be subtracted from another is stored | 0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number where the data from which another is to be subtracted is stored | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the single-precision real number in the device specified by (s) from the single-precision real number in the device specified by (d), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s) and (d) should be 0 or $2^{-126} \leq$ |specified value (stored value)| $< 2^{128}$.
- When an input value is set from the engineering tool, a rounding error may be occur.
- The table below shows the related devices.

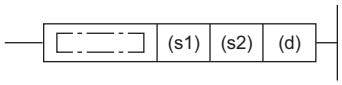
| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result $< 2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

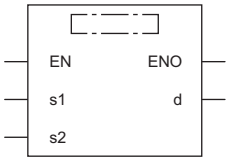
| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

## E-(P) [For 3 operands]

These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=EMINUS(EN, s1, s2, d);<br>ENO:=EMINUSP(EN, s1, s2, d); |

**FBD/LD**




("EMINUS", "EMINUSP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

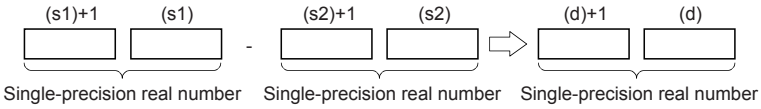
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Minuend data or head device number where the data from which another is to be subtracted is stored | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Subtrahend data or head device number where the data to be subtracted from another is stored | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s1), (s2), and (d) should be 0 or $2^{-126}$ |specified value (stored value)|$\leq 2^{128}$.
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

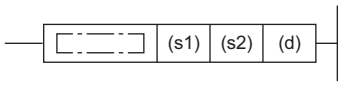
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

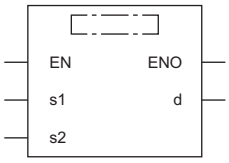
# Adding single-precision real numbers

## DEADD(P)

These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEADD(EN,s1,s2,d);<br>ENO:=DEADDP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

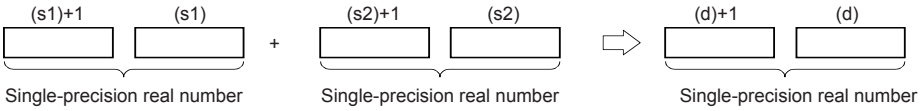
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Augend data or head device number where the data to which another is added is stored | 0, $2^{-126} \leq \lvert(s1)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Addend data or head device number where the data that is added to another is stored | 0, $2^{-126} \leq \lvert(s2)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

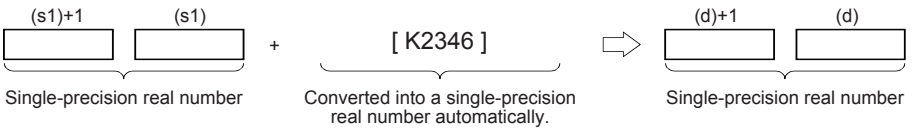
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- When the constant (K or H) is specified in (s1) and (s2), these instructions convert values into single-precision real number automatically.



• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Precautions

The same device number can be specified for (s1), (s2), and (d). In this case, note that the addition result changes in every operation cycle when the continuous operation type instruction (DEADD) is used.

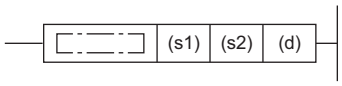
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |

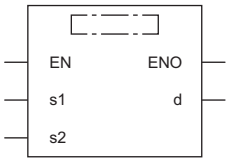
# Subtracting single-precision real numbers

## DESUB(P)

These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSUB(EN,s1,s2,d);<br>ENO:=DSUBP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

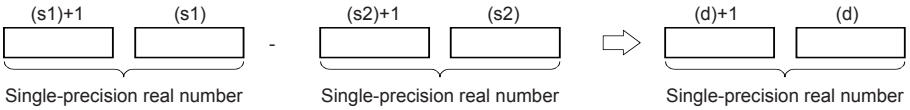
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Minuend data or head device number where the data from which another is subtracted is stored | 0, $2^{-126} \leq \lvert(s1)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Minuend data or head device number where the data that is subtracted another is stored | 0, $2^{-126} \leq \lvert(s2)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

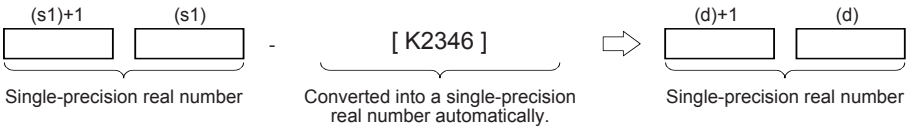
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- When the constant (K or H) is specified in (s1) and (s2), these instructions convert values into single-precision real number automatically.



• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Precautions

The same device number can be specified for (s1), (s2), and (d). In this case, note that the subtraction result changes in every operation cycle when the continuous operation type instruction (DESUB) is used.

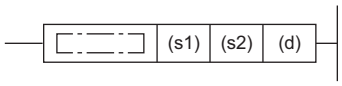
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |

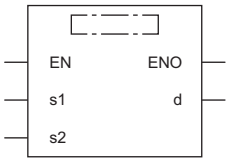
# Multiplying single-precision real numbers

## E*(P)

These instructions multiply the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("EMULTI", "EMULTIP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

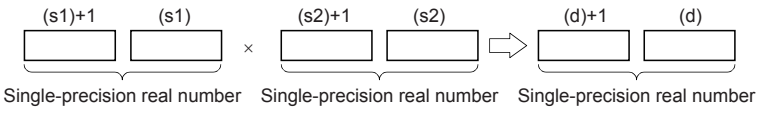
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Multiplicand data or head device number where the data to be multiplied by another is stored | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Multiplier data or head device number where the data by which another is to be multiplied is stored | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions multiply the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s1), (s2), and (d) should be 0 or $2^{-126}$ |specified value (stored value)|$\leq 2^{128}$.
- When an input value is set from the engineering tool, a rounding error may occur.

• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

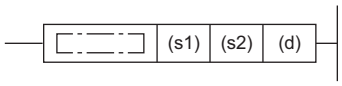
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

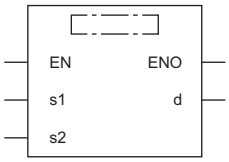
# Dividing single-precision real numbers

## E/(P)

These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




("EDIVISION", "EDIVISIONP" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

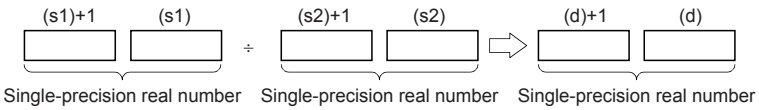
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Dividend data or head device number where the data which is divided by another is stored. | 0, $2^{-126}\leq\|(s1)\|<2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Divisor data or head device number where the data that divides another is stored. | 0, $2^{-126}\leq\|(s2)\|<2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the result in the device specified by (d).




- Values in the devices specified (stored) by (s1), (s2), and (d) should be 0 or $2^{-126}<$|specified value (stored value)|$\leq 2^{128}$.
- When an input value is set from the engineering tool, a rounding error may occur.

• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

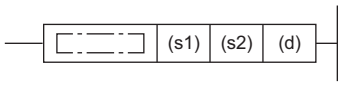
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The divisor is 0. |
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

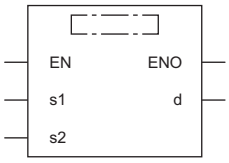
# Multiplying single-precision real numbers

## DEMUL(P)

These instructions multiply the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEMUL(EN,s1,s2,d);<br>ENO:=DEMULP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

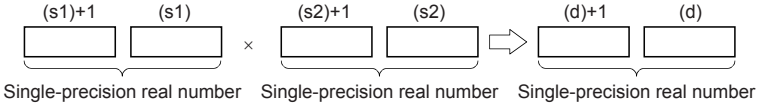
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Multiplicand data or head device number where the data to be multiplied by another is stored | 0, $2^{-126} \leq \lvert(s1)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Multiplier data or head device number where the data by which another is to be multiplied is stored | 0, $2^{-126} \leq \lvert(s2)\rvert \leq 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

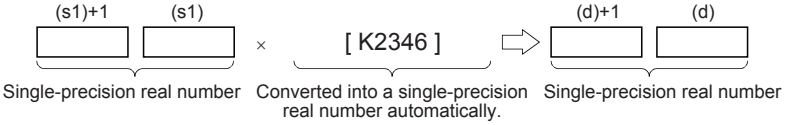
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions multiply the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).




- When the constant (K or H) is specified in (s1) and (s2), these instructions convert values into single-precision real number automatically.



• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

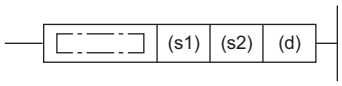
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

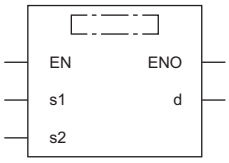
# Dividing single-precision real numbers

## DEDIV(P)

These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEDIV(EN,s1,s2,d);<br>ENO:=DEDIVP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

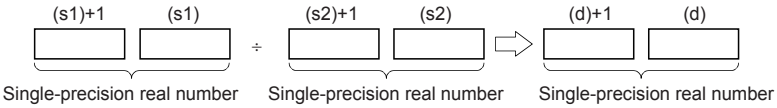
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Dividend data or head device number where the data which is divided by another is stored. | — | Single-precision real number | ANYREAL_32 |
| (s2) | Divisor data or head device number where the data that divides another is stored. | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the result in the device specified by (d).




- When the constant (K or H) is specified in (s1) and (s2), these instructions convert values into single-precision real number automatically.

• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

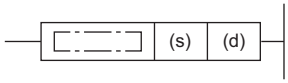
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The divisor is 0. |
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value stored in specified device is outside the following range<br>0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$ |

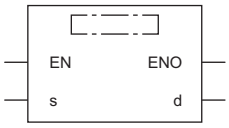
# Converting 16-bit signed binary data to single-precision real number

## INT2FLT(P)

These instructions convert the 16-bit signed binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

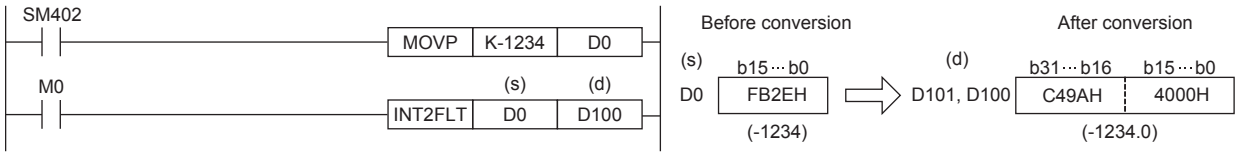
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| (d) | Data after conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).



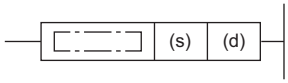

### Operation error

There is no operation error.

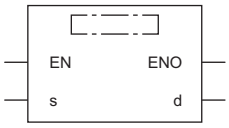
# Converting 16-bit unsigned binary data to single-precision real number

## UINT2FLT(P)

These instructions convert the 16-bit unsigned binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

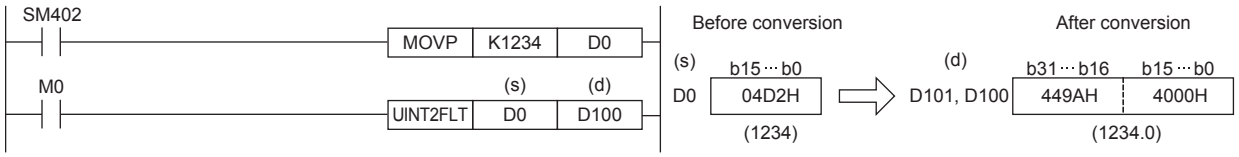
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 65535 | 16-bit unsigned binary | ANY32_U |
| (d) | Data after conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).



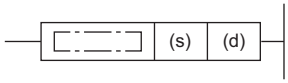

### Operation error

There is no operation error.

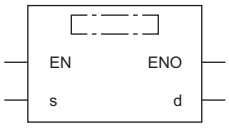
# Converting 32-bit signed binary data to single-precision real number

## DINT2FLT(P)

These instructions convert the 32-bit signed binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

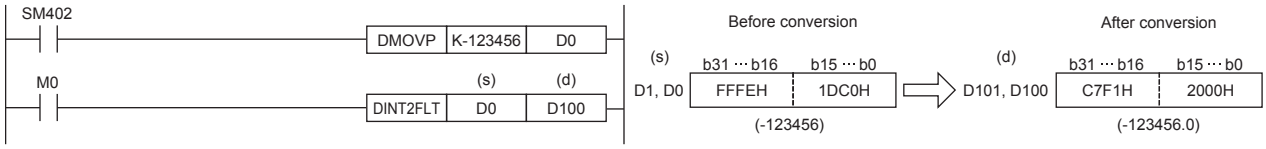
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| (d) | Data after conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).



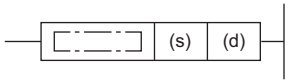

### Operation error

There is no operation error.

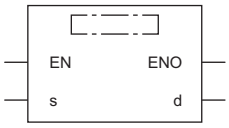
# Converting 32-bit unsigned binary data to single-precision real number

## UDINT2FLT(P)

These instructions convert the 32-bit unsigned binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | Not supported |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

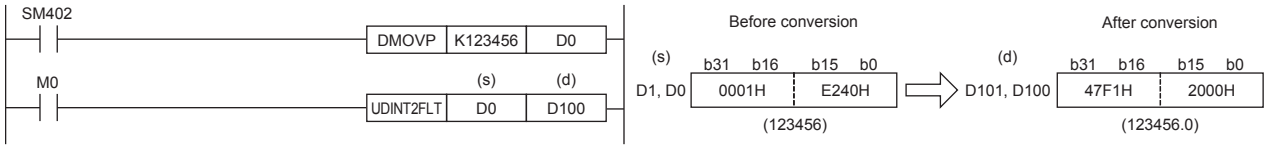
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data before conversion | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | Data after conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to single-precision real number, and store the converted data in the device specified by (d).




### Operation error

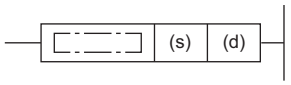
There is no operation error.

# Converting character string to single-precision real number

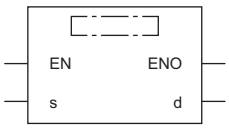
## EVAL(P)/DEVAL(P)

These instructions convert the character strings in the device areas specified by (s) and later to single-precision real number, and store the converted data in the device specified by (d).

The EVAL(P) instructions can also be used as DEVAL(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=EVAL(EN,s,d);<br>ENO:=EVALP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Character string data to be converted to single-precision real number or head device number where the character string data is stored | — | Character string | ANYSTRING_SINGLE |
| (d) | Head device number storing converted single-precision real number | — | Single-precision real number | ANYREAL_32 |

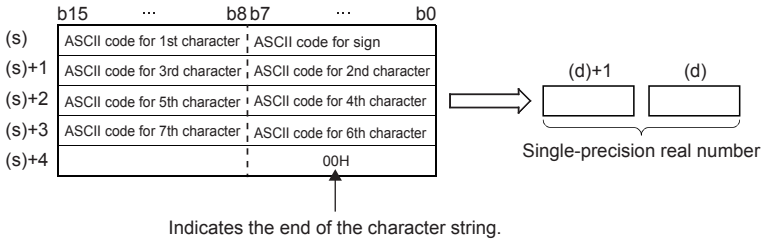
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | ○ | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

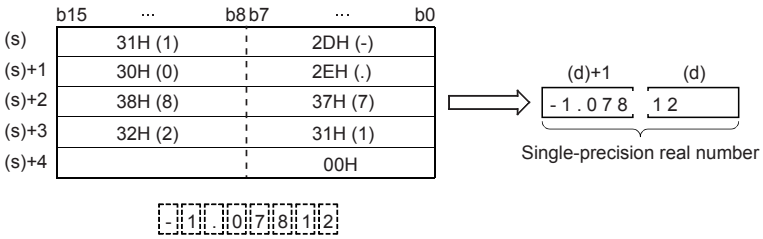
- These instructions convert the character strings in the device areas specified by (s) and later to single-precision real number, and store the converted data in the device specified by (d).
- A specified character string may be in the decimal point format or exponent format. A character string in either format can be converted into single-precision real number.



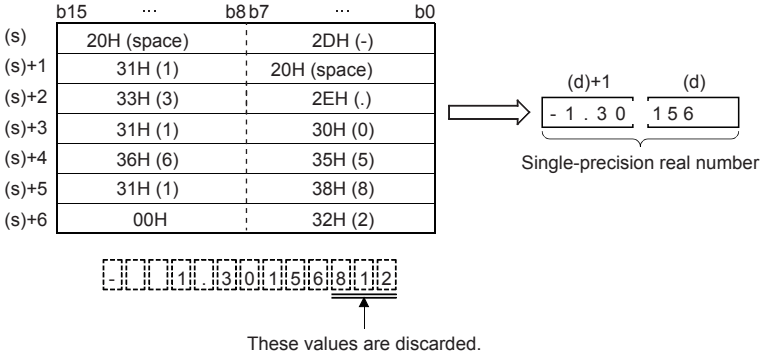

- A character string can consist of up to 24 characters. 20H (space) and 30H (0) in a character string are counted as one character each.

## ■Decimal point format

• When the character string specified by (s) is decimal point format, the operation is executed as follows.

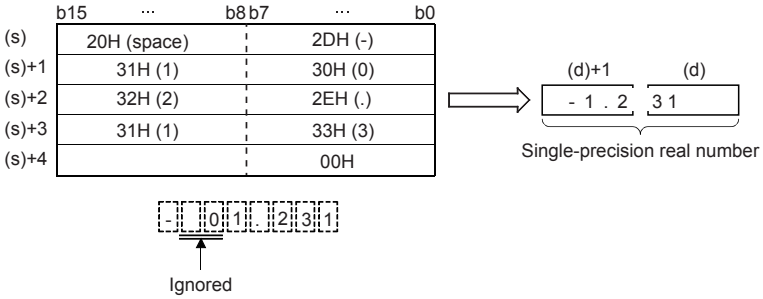



• With regard to character string, six digits excluding the sign, decimal point and exponent part are valid, and the 7th and later digits are discarded during conversion.



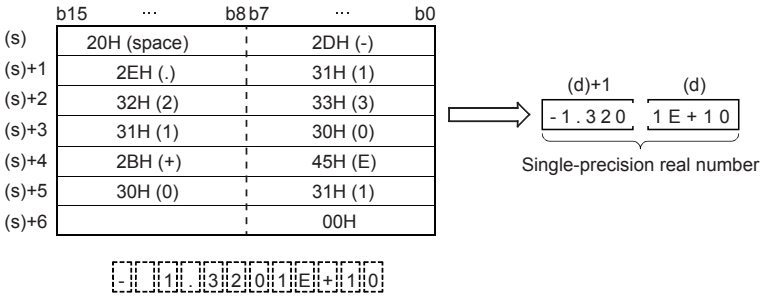

• When 2BH (+) is specified as the sign in the floating point format or when the sign is omitted, a character string is converted into a positive value. It is handled as negative value during conversion when the sign is set to 2DH (-).

• When 20H (space) or 30H (0) exists between numbers except the first 0 in a character string specified by (s), 20H or 30H is ignored during conversion.

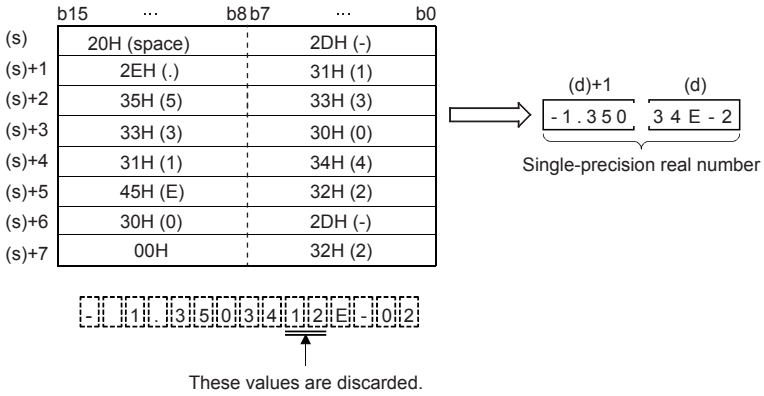



## ■Exponent format

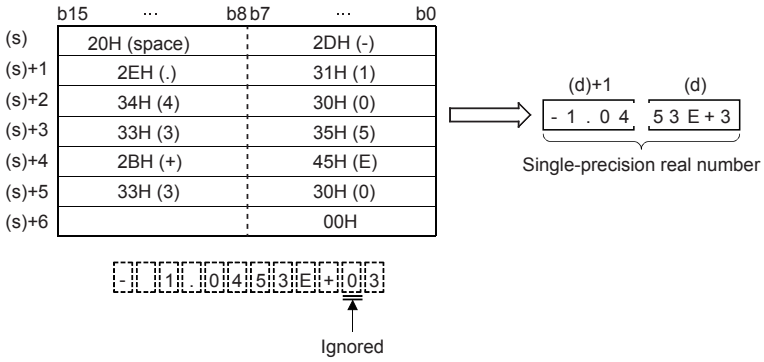
• When the character string specified by (s) is in exponent format, the operation is executed as follows.



- With regard to character string, six digits excluding the sign, decimal point and exponent part are valid, and the 7th and later digits are discarded during conversion.




- String data in the exponent format is handled as positive value during conversion when the sign of the exponent part is set to 2BH (+) or when the sign is omitted. When 2DH (-) is specified as the sign, a character string is converted into a negative value.
- When 20H (space) or 30H (0) exists between numbers except the first 0 in a character string specified by (s), 20H or 30H is ignored during conversion.
- When 30H (0) exists between a number and "E" in a character string in the exponent format, 30H is ignored during conversion.



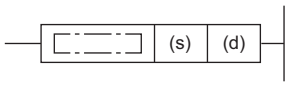

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | 00H does not exist in the corresponding device range starting from (s) |
| 3401H | Characters other than 30 (0) to 39 (9) exist in a character string specified by (s) |
| | 2EH (.) exists in two or more positions in a character string specified by (s) |
| | Any character other than 45H (E), 2BH (+), or 2DH (-) exists in the exponent part specified by (s), or two or more exponent parts exist |
| 3405H | The number of characters after (s) is 0 or more than 24 |

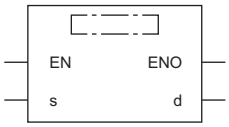
# Converting binary floating point to decimal floating point

## DEBCD(P)

These instructions convert the binary floating point specified by (s) to decimal floating point, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEBCD(EN,s,d);<br>ENO:= DEBCDP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

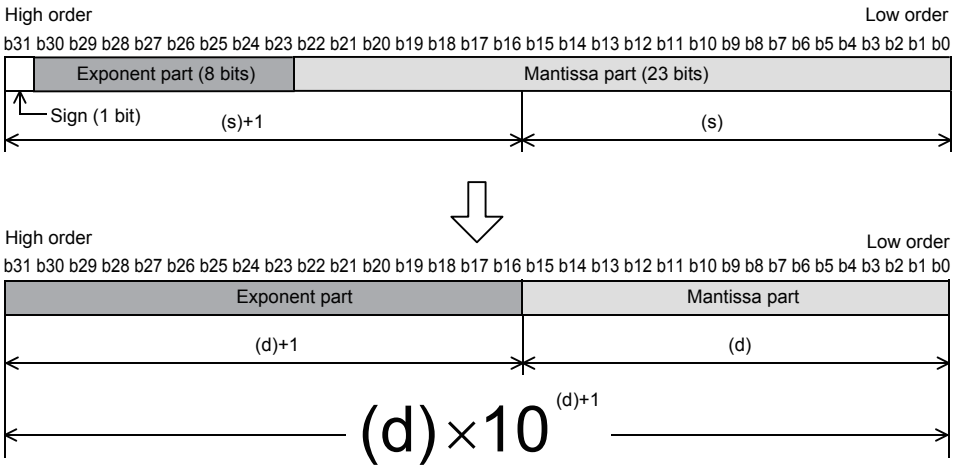
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing binary floating point data | — | Single-precision real number | ANYREAL_32 |
| (d) | Device number storing converted decimal floating point | — | Real number | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the binary floating point specified by (s) to decimal floating point, and store the converted data in the device specified by (d).




### Precautions

In floating point operations, all data is handled in binary floating point. Because binary floating point is difficult to understand (requiring a dedicated monitoring method), it is converted into scientific notation (decimal floating point) so that monitoring can be easily executed by peripheral equipment.

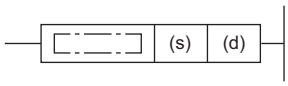
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |

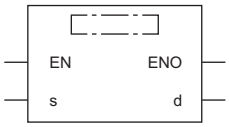
# Converting decimal floating point to binary floating point

## DEBIN(P)

These instructions convert the decimal floating point specified by (s) to the binary floating point, and store the converted data in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEBIN(EN,s,d);<br>ENO:= DEBINP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

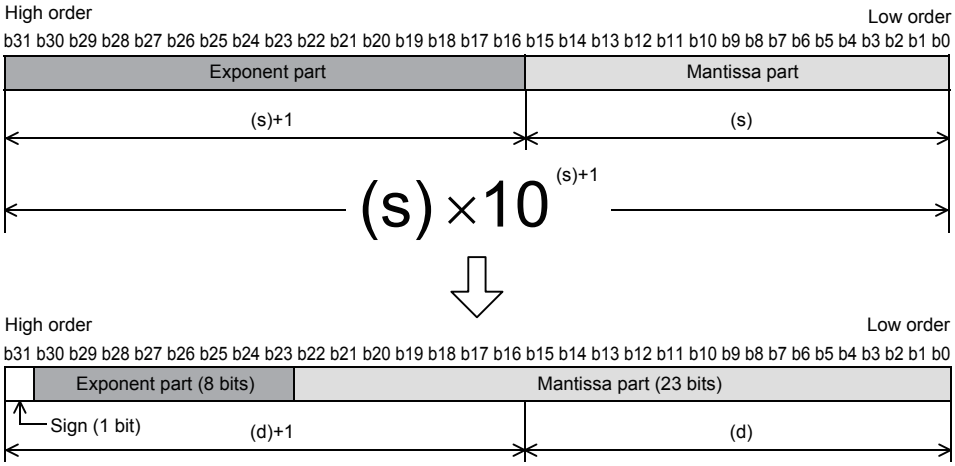
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing decimal floating-point data | — | Real number | ANY32 |
| (d) | Device number storing converted binary floating-point data | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert the decimal floating point specified by (s) to the binary floating point, and store the converted data in the device specified by (d).



• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

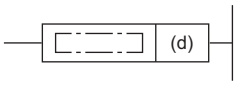
There is no operation error.

# Inverting the sign of single-precision real number

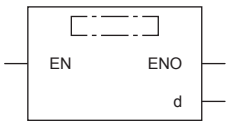
## ENEG(P)/DENEG(P)

These instructions invert the sign of the single-precision real number specified by (d), and store the data of the device specified by (d).

The ENEG(P) instructions can also be used as DENEG(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ENEG(EN,d);<br>ENO:=ENEGP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

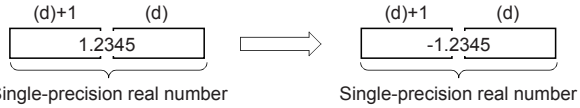
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number storing single-precision real number whose sign is to be inverted | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions invert the sign of the single-precision real number specified by (d), and store the data in the device specified by (d).




- Use these instructions for inverting the positive and negative sign.

### Operation error

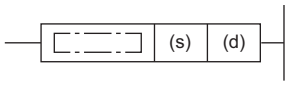
There is no operation error.

# Transferring single-precision real number data

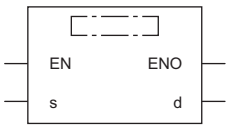
## EMOV(P)/DEMOV(P)

These instructions transfer the single-precision real number data stored in the device specified by (s) to the device specified by (d).

The EMOV(P) instructions can also be used as DEMOV(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=EMOV(EN,s,d);<br>ENO:=EMOVP(EN,s,d) |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

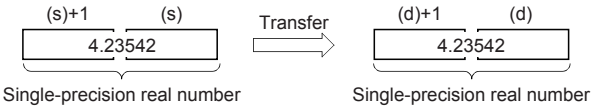
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data to be transferred or head device number where the data to be transferred is stored | 0, $2^{-126}$<\|(s)\|<$2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Device number storing the data in transfer destination | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions transfer the single-precision real number data stored in the device specified by (s) to the device specified by (d).




### Operation error

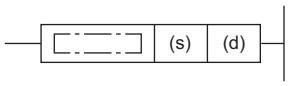
There is no operation error.

# Calculating the sine of single-precision real number

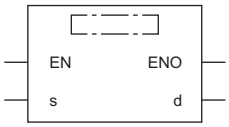
## SIN(P)/DSIN(P)

These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).

The SIN(P) instructions can also be used as DSIN(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=SINP(EN,s,d); |

**FBD/LD*1**




*1 The SIN instruction is not supported by the ST language and the FBD/LD language. Use SIN of the standard function.
☞ Page 866 SIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

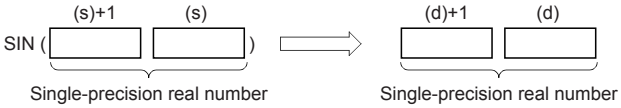
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Angle data or head device number where the angle data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).




- Set the angle data in radians (angle×π÷180).

• The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

**Point**

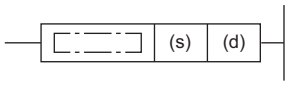
For the angle↔radian conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Calculating the cosine of single-precision real number

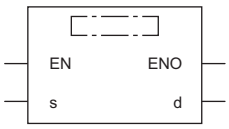
## COS(P)/DCOS(P)

These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).

The COS(P) instructions can also be used as DCOS(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=COSP(EN,s,d); |

**FBD/LD*1**




*1 The COS instruction is not supported by the ST language and the FBD/LD language. Use COS of the standard function.
☞ Page 867 COS(_E)

### Setting data

#### ■Descriptions, ranges, and data types

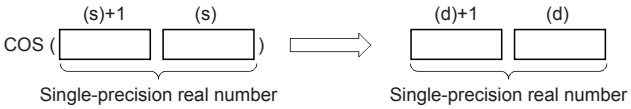
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Angle data or head device number where the angle data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).




- Set the angle data in radians (angle×π÷180).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

**Point**

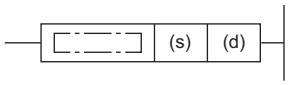
For the angle↔radian conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Calculating the tangent of single-precision real number

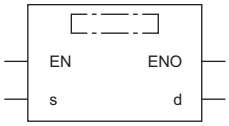
## TAN(P)/DTAN(P)

These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).

The TAN(P) instructions can also be used as DTAN(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=TANP(EN,s,d); |

**FBD/LD*1**




*1 The TAN instruction is not supported by the ST language and the FBD/LD language. Use TAN of the standard function.
☞ Page 868 TAN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

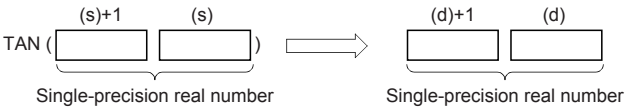
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Angle data or head device number where the angle data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).




- Set the angle data in radians (angle×π÷180).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Precautions

When the angle specified by (s) is π/2 radian or (3/2)π radian, no error occurs because an operation error occurs in a radian value.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

**Point**

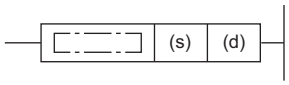
For the angle↔radian conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Calculating the arc sine of single-precision real number

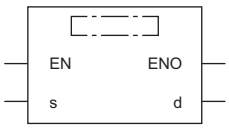
## ASIN(P)/DASIN(P)

These instructions calculate the angle from the sine of the angle specified by (s), and store the operation result in the word device specified by (d).

The ASIN(P) instructions can also be used as DASIN(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=ASINP(EN,s,d); |

**FBD/LD*1**




*1 The ASIN instruction is not supported by the ST language and the FBD/LD language. Use ASIN of the standard function.
☞ Page 869 ASIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

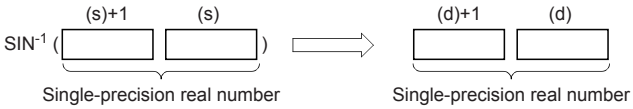
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | A sine value used in $SIN^{-1}$ (arc sine) operation or head device number storing the sine value | -1.0 to +1.0 | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | $-\pi/2$ to $+\pi/2$ | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the angle from the sine of the angle specified by (s), and store the operation result in the device specified by (d).




- The sine value specified by (s) can be set ranging from -1.0 to 1.0.
- The angle (operation result) stored in (d) is expressed in radians (from $(-\pi/2)$ to $(\pi/2)$).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result $< 2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result $\geq 2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| 3405H | A value specified in (s) is outside the range from -1.0 to 1.0. |

**Point**

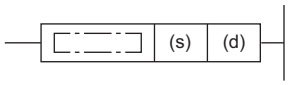
For the radian↔angle conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Calculating the arc cosine of single-precision real number

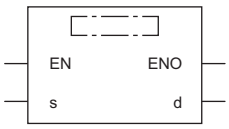
## ACOS(P)/DACOS(P)

These instructions calculate the angle from the cosine of the angle specified by (s), and store the operation result in the word device specified by (d).

The ACOS(P) instructions can also be used as DACOS(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=ACOSP(EN,s,d) |

**FBD/LD*1**




*1 The ACOS instruction is not supported by the ST language and the FBD/LD language. Use ACOS of the standard function.
☞ Page 870 ACOS(_E)

### Setting data

#### ■Descriptions, ranges, and data types

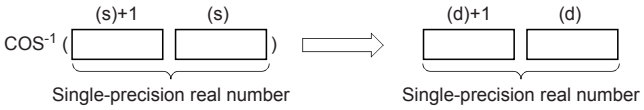
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | A cosine value used in $COS^{-1}$ (arc cosine) operation or head device number storing the cosine value | -1.0 to +1.0 | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | 0 to π | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the angle from the cosine of the angle specified by (s), and store the operation result in the device specified by (d).




- The cosine value specified by (s) can be set ranging from -1.0 to 1.0
- The angle (operation result) stored in (d) is expressed in radians (0 to π).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |
| 3405H | A value specified in (s) is outside the range from -1.0 to 1.0. |

**Point**

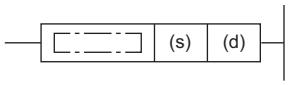
For the radian↔angle conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Calculating the arc tangent of single-precision real number

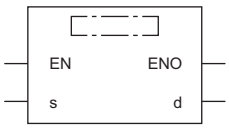
## ATAN(P)/DATAN(P)

These instructions calculate the angle from the tangent of the angle specified by (s), and store the operation result in the word device specified by (d).

The ATAN(P) instructions can also be used as DATAN(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=ATANP(EN,s,d); |

**FBD/LD*1**




*1 The ATAN instruction is not supported by the ST language and the FBD/LD language. Use ATAN of the standard function.
☞ Page 871 ATAN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

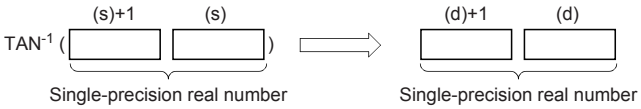
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | A tangent value used in the $TAN^{-1}$ (arc tangent) operation or head device number storing the tangent value | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | $-\pi/2$ to $+\pi/2$ | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the angle from the tangent of the angle specified by (s), and store the operation result in the device specified by (d).




- The angle (operation result) stored in (d) is expressed in radians (from $-\pi/2$ to $\pi/2$).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

**Point**

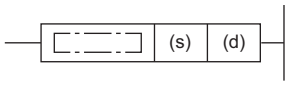
For the radian↔angle conversion, refer to the DRAD(P) and DDEG(P) instructions.
(☞ Page 492 Converting single-precision real number angle to radian, Page 494 Converting single-precision real number radian to angle)

# Converting single-precision real number angle to radian

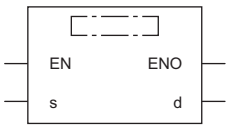
## RAD(P)/DRAD(P)

These instructions convert a unit of angle from degrees (DEG.) specified by (s) into radians, and store the converted angle in the device specified by (d).

The RAD(P) instructions can also be used as DRAD(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RAD(EN,s,d);<br>ENO:=RADP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

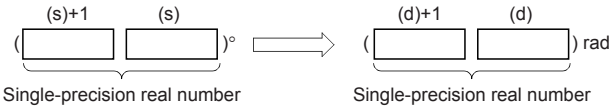
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | A value in degrees to be converted into a value in radians or the start number storing the value in degrees | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number storing a value in radians acquired by conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert a unit of angle from degrees (DEG.) specified by (s) into radians, and store the converted angle in the device specified by (d).




- The conversion from degrees into radians is executed as follows:

$$\text{Radians} = \text{Degrees} \times \frac{\pi}{180}$$

- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0".<br>(The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

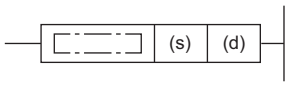
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

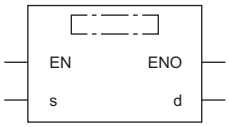
# Converting single-precision real number radian to angle

## DEG(P)/DDEG(P)

These instructions convert a unit of angle from radians specified by (s) into degrees (DEG.), and store the converted angle in the device specified by (d). The DEG(P) instructions can also be used as DDEG(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DEG(EN,s,d);<br>ENO:=DEGP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | A value in radians to be converted into a value in degrees or the head device number storing a value in radians | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number storing a value in degrees acquired by conversion | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions convert a unit of angle from radians specified by (s) into degrees (DEG.), and store the converted angle in the device specified by (d).


Single-precision real number　　Single-precision real number

- The conversion from radians into degrees is executed as follows:

$$\text{Degrees} = \text{Radians} \times \frac{180}{\pi}$$

- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

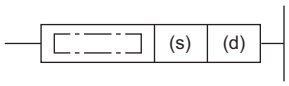
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

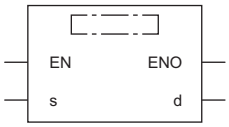
# Calculating the square root of single-precision real number

## DESQR(P)/ESQRT(P)

These instructions calculate the square root of a value specified by (s), and store the operation result in the device specified by (d). The DESQR(P) instructions can also be used as ESQRT(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DESQR(EN,s,d);<br>ENO:=DESQRP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

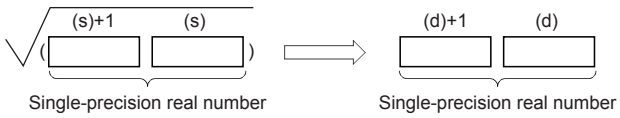
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data whose square root is calculated or head device number where the data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | ○ | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the square root of a value specified by (s), and store the operation result in the device specified by (d).




- Only a positive value can be set in (s). (The square root operation cannot be executed for a negative value).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |

### Operation error

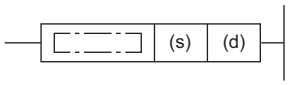
| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |
| 3405H | The value stored in a device specified in (s) is negative. |

# Calculating the exponent of single-precision real number

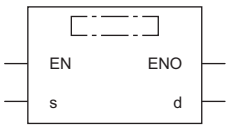
## EXP(P)/DEXP(P)

These instructions calculate the exponent of a value specified by (s), and store the operation result in the device specified by (d).

The EXP(P) instructions can also be used as DEXP(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=EXPP(EN,s,d); |

**FBD/LD*1**




*1 The EXP instruction is not supported by the ST language and the FBD/LD language. Use EXP of the standard function.
☞ Page 865 EXP(_E)

### Setting data

#### ■Descriptions, ranges, and data types

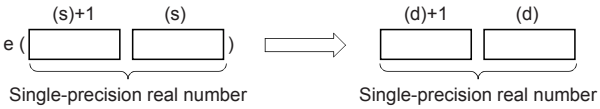
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data whose exponent is calculated or head device number where the data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the exponent of a value specified by (s), and store the operation result in the device specified by (d).




- In the exponential operation, the base (e) is set to "2.71828".
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is -0, denormalized number, NaN (not a number), or ±∞. |

**Point**

- The EXP(P) instructions execute operations in natural logarithm. For obtaining a value in common logarithm, specify a common logarithm value divided by 0.4342945 in (s).

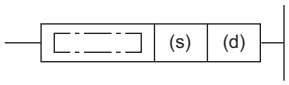
$$10^{X} = e^{\frac{X}{0.4342945}}$$

# Calculating the natural logarithm of single-precision real number

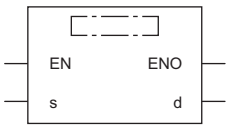
## LOG(P)/DLOGE(P)

These instructions calculate the logarithm whose base is natural logarithm e of a value specified by (s), and store the operation result in the device specified by (d).

The LOG(P) instructions can also be used as DLOGE(P).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=LOGP(EN,s,d); |

**FBD/LD*1**




*1 The LOG instruction is not supported by the ST language and the FBD/LD language. Use LOG of the standard function.
☞ Page 863 LOG(_E)

### Setting data

#### ■Descriptions, ranges, and data types

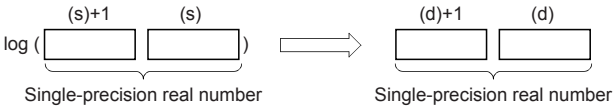
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data whose natural logarithm is calculated or head device number where the data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the logarithm whose base is natural logarithm e of a value specified by (s), and store the operation result in the device specified by (d).




- Only a positive value can be set in (s). (The natural logarithm operation cannot be executed for a negative value).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

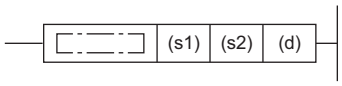
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |
| 3405H | The value stored in a device specified in (s) is negative. |
| | The value stored in a device specified in (s) is 0. |

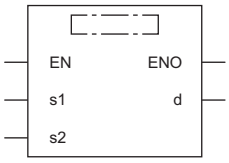
# Calculating the exponentiation of single-precision real number

## POW(P)

These instructions raise float (single precision) data stored in a device specified by (s1) by the single-precision real number specified by (s2), and store the operation result in a device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=POW(EN,s1,s2,d);<br>ENO:=POWP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

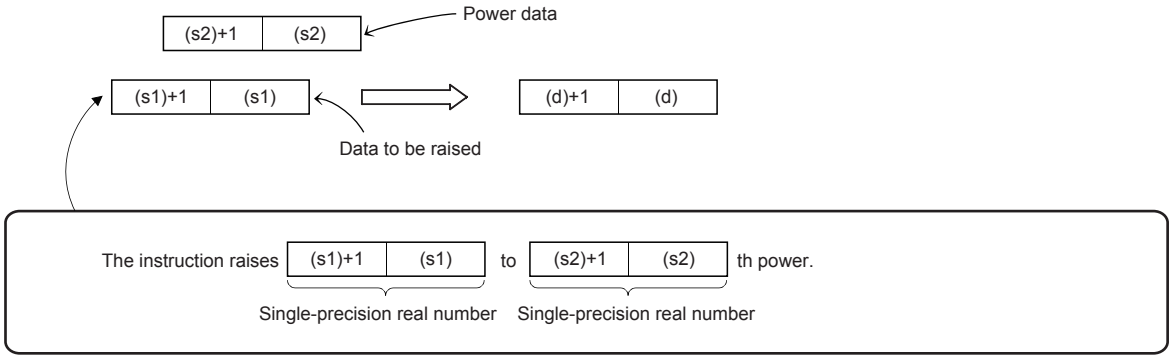
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data to be raised, or head device number which stores such data | 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (s2) | Power data, or head device number which stores such data | 0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$ | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (s2) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions raise float (single precision) data stored in a device specified by (s1) by the single-precision real number specified by (s2), and store the operation result in a device specified by (d).




- Values in the devices specified (stored) by (s1) and (s2) should be 0 or $2^{-126} \leq$ |specified value (stored value)| $< 2^{128}$.
- When the operation result is -0 or underflow occurs, the operation result is regarded as 0.
- When the operation result is within the following range, the operation result is regarded as $2^{128}$, and the carry flag SM716 turns on.

  $2^{128} \leq$ |operation result|
- When an input value is set from the engineering tool, a rounding error may be occur.

## Operation error

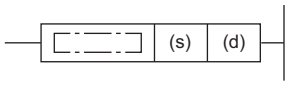
| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value specified by (s1) or (s2) is outside the following range.<br>0, $2^{-126} \leq$ \|specified value (stored value)\| $< 2^{128}$ |
| | The specified device value is -0, denormalized number, NaN (not a number), or $\pm\infty$. |
| 3403H | The operation result is within the following range. (An overflow has occurred.)<br>$2^{128} \leq$ \|operation result\| |

# Calculating the common logarithm of single-precision real number

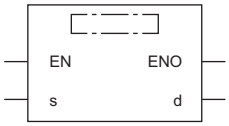
## LOG10(P)/DLOG10(P)

These instructions calculate the common logarithm (the logarithm whose base is 10) of a value specified by (s), and store the operation result in the device specified by (d).

The LOG10(P) instructions can also be used as DLOG10(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=LOG10(EN,s,d);<br>ENO:=LOG10P(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Data whose common logarithm is calculated or head device number where the data is stored | — | Single-precision real number | ANYREAL_32 |
| (d) | Head device number for storing the operation result | — | Single-precision real number | ANYREAL_32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | ○ | — | — |
| (d) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the common logarithm (the logarithm whose base is 10) of a value specified by (s), and store the operation result in the device specified by (d).


- Only a positive value can be set in (s). (The common logarithm operation cannot be executed for a negative value).
- The table below shows the related devices.

| Device | Name | Description | |
|---|---|---|---|
| | | Condition | Operation |
| SM700 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM700 turns on. |
| SM8020 | Zero | The operation result is true "0". (The mantissa part is "0"). | The zero flag SM8020 turns on. |
| SM8021 | Borrow | The absolute value of the operation result < $2^{-126}$ | The value of (d) is the minimum value ($2^{-126}$) of 32-bit real numbers and the borrow flag SM8021 turns on. |
| SM8022 | Carry | The absolute value of the operation result ≥ $2^{128}$ | The value of (d) is the maximum value ($2^{128}$) of 32-bit real numbers and the carry flag SM8022 turns on. |

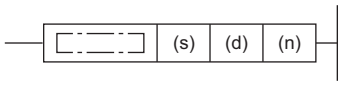
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The specified device value is denormalized number, NaN (not a number), or ±∞. |
| 3405H | The value stored in a device specified in (s) is negative. |
| | The value stored in a device specified in (s) is 0. |

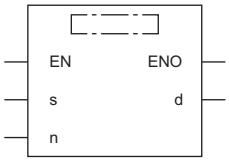
# Searching the maximum value of single-precision real number

## EMAX(P)

These instructions search for the maximum value in the (n) point(s) of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device areas specified by (d) and (d)+1.
These instructions also store the location of the first maximum value from (s) in the device specified by (d)+2 and the number of maximum values in the device specified by (d)+3.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=EMAXP(EN,s,n,d); |

**FBD/LD*1**




*1 The EMAX instruction is not supported by the ST language and the FBD/LD language. Use MAX of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Search target data | — | Single-precision real number | ANYREAL_32 |
| (d) | Search result | — | Single-precision real number | ANY_REAL_32_ARRAY (Number of elements: 4) |
| (n) | Number of search target data points | — | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search for the maximum value in the (n) point(s) of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device areas specified by (d). These instructions also store the location of the first maximum value from (s) in the device specified by (d)+2 and the number of maximum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as 1st point when the location is counted.

• The following values are stored in (d).

| | Data type | Description |
|---|---|---|
| (d) | Single-precision real number | Maximum value |
| (d)+1 | | |
| (d)+2 | 16-bit data | Maximum value position |
| (d)+3 | 16-bit data | Number of maximum values |

• When (n) is 0, the processing is not performed.

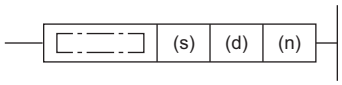
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device areas specified by (s) exceed the corresponding device range. |
| | The device areas specified by (d) exceed the corresponding device range. |
| 3402H | The block data in the device areas specified by (s) includes a value other than single-precision real number. |

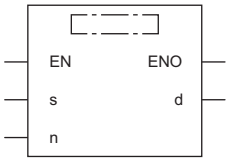
# Searching the minimum value of single-precision real number

## EMIN(P)

These instructions search for the minimum value in the (n) point(s) of single-precision real number block data in the device starting from the one specified by (s), and store the minimum value in the device areas specified by (d) and (d)+1.
These instructions also store the location of the first minimum value from (s) in the device specified by (d)+2 and the number of minimum values in the device specified by (d)+3.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=EMINP(EN,s,n,d); |

**FBD/LD*1**




*1 The EMIN instruction is not supported by the ST language and the FBD/LD language. Use MIN of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

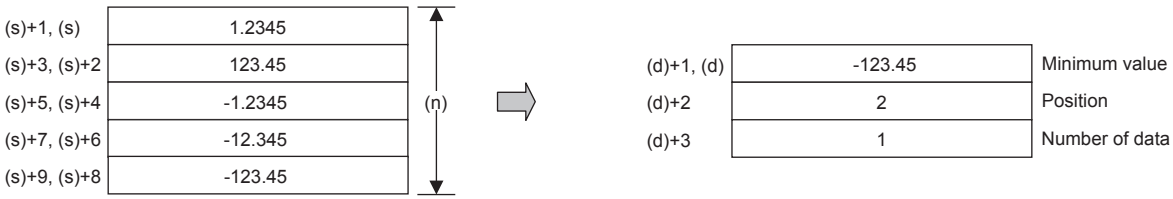
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Search target data | — | Single-precision real number | ANYREAL_32 |
| (d) | Search result | — | Single-precision real number | ANY_REAL_32_ARRAY (Number of elements: 4) |
| (n) | Number of search target data points | — | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search for the minimum value in the (n) point(s) of single-precision real number block data in the device starting from the one specified by (s), and store the minimum value in the device areas specified by (d) and (d)+1. These instructions also store the location of the first minimum value from (s) in the device specified by (d)+2 and the number of minimum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as 1st point when the location is counted.



• The following values are stored in (d).

| | Data type | Description |
|---|---|---|
| (d) | Single-precision real number | Minimum value |
| (d)+1 | | |
| (d)+2 | 16-bit data | Minimum value position |
| (d)+3 | 16-bit data | Number of minimum values |

• When (n) is 0, the processing is not performed.

## Operation error

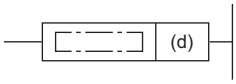
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device areas specified by (s) exceed the corresponding device range. |
| | The device areas specified by (d) exceed the corresponding device range. |
| 3402H | The block data in the device areas specified by (s) includes a value other than single-precision real number. |

# 8.8 Random Number Instruction

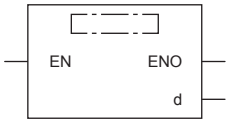
## Generating random number

### RND(P)

These instructions generate a pseudo-random number ranging from 0 to 32767, and store it as a random number to a device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RND(EN,d);<br>ENO:=RNDP(EN,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number storing a random number | — | 16-bit signed binary | ANY16 |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions generate a pseudo-random number ranging from 0 to 32767, and store it as a random number to a device specified by (d).
- In the pseudo-random number sequence, the source value of a random number is calculated every time, and this instruction calculates a pseudo-random number using the source value.

  Pseudo-random number calculation equation:

  (SD8311, SD8310)=(SD8311, SD8310)*1×1103515245+12345

  (d) ="([SD8311, SD8310]>>16)&<logical product>00007FFFh"

*1 To (SD8311, SD8310), write a non-negative value (0 to 2147483647) only once when the CPU module mode switches from STOP to RUN. (K1 is written to (SD8311, SD8310) as the initial value when the power is restored.)

#### Operation error

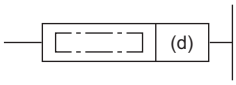
There is no operation error.

# 8.9 Index Register Operation Instruction

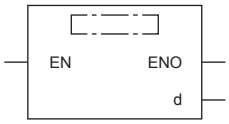
## Saving all data of the index register

### ZPUSH(P)

These instructions save the contents of index registers and long index registers in the devices specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ZPUSH(EN,d);<br>ENO:=ZPUSHP(EN,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

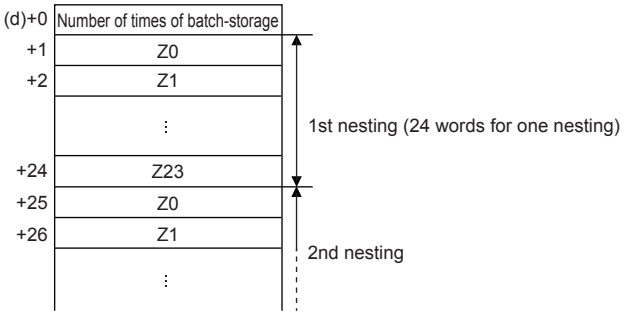
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number for saving the data of index registers and long index registers | — | 16-bit signed binary | ANY16 |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions save the contents of index registers and long index registers in the devices specified by (d) and later.
- When the contents of index registers are saved, "1" is added to (d).
- These instructions save the contents of index registers and long index registers for 24 words regardless of the assignment of the number of the registers. Thus, when the number of index registers is 0, the contents of long index registers are saved for 12 points.
- The ZPOP(P) instructions are used to return the data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs, and by using the same device in (d) a nesting structure can be adopted. (☞ Page 512 Returning all data of the index register)
- When a nesting structure is adopted, the areas to be used are added to (d) and later every time the ZPUSH(P) instructions are used. Check the number of index registers and long index registers by SD300 and SD302, and secure the areas for the number of instructions to be used in advance.
- The following shows the areas of (d) and later to be used.



## Precautions

- When a nesting structure is not adopted, clear (d) before executing the ZPUSH(P) instructions.
- When a nesting structure is adopted, clear (d) before executing the first ZPUSH(P) instructions.
- When the ZPOP(P) instructions are used to return the data of index registers, use the ZPOP(P) instructions corresponding to the ZPUSH(P) instructions that were used for saving the data.
  ZPUSH(P) (One setting data) → ZPOP(P) (One setting data)
  ZPUSH(P) (Two setting data) → ZPOP(P) (Two setting data)
- Secure the areas so that the save destination specified by (d) do not exceed the device range.

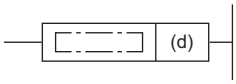
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of points used in (d) or later exceeds the range of the target device/label area. |
| 3405H | (d) is negative. |

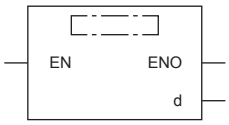
# Returning all data of the index register

## ZPOP(P)

These instructions read the data saved in the devices specified by (d) and later to index registers and long index registers.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ZPOP(EN,d);<br>ENO:=ZPOPP(EN,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number for returning the data of index registers | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

### Processing details

- These instructions read the data saved in the devices specified by (d) and later to index registers and long index registers.
- When the saved contents of the index registers and long index registers are read, "1" is subtracted from (d).
- The ZPUSH(P) instructions are used to temporarily save the data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs.

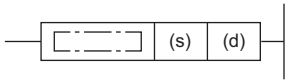
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of points used in (d) or later exceeds the range of the target device/label area. |
| 3405H | (d) is 0 or negative. |

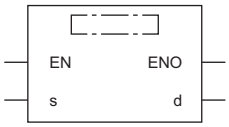
# Saving the selected data of the index register and long index register

## ZPUSH(P)

These instructions save the contents of index registers and long index registers within the range specified by (s) in the devices specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ZPUSH_2(EN,s,d);<br>ENO:=ZPUSHP_2(EN,s,d); |

**FBD/LD**




("ZPUSH_2", "ZPUSHP_2" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Type of the index register or long index register to be saved | 0 to 2 | 16-bit unsigned binary | ANY16 |
| (d) | Head device number for saving the data of index registers | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

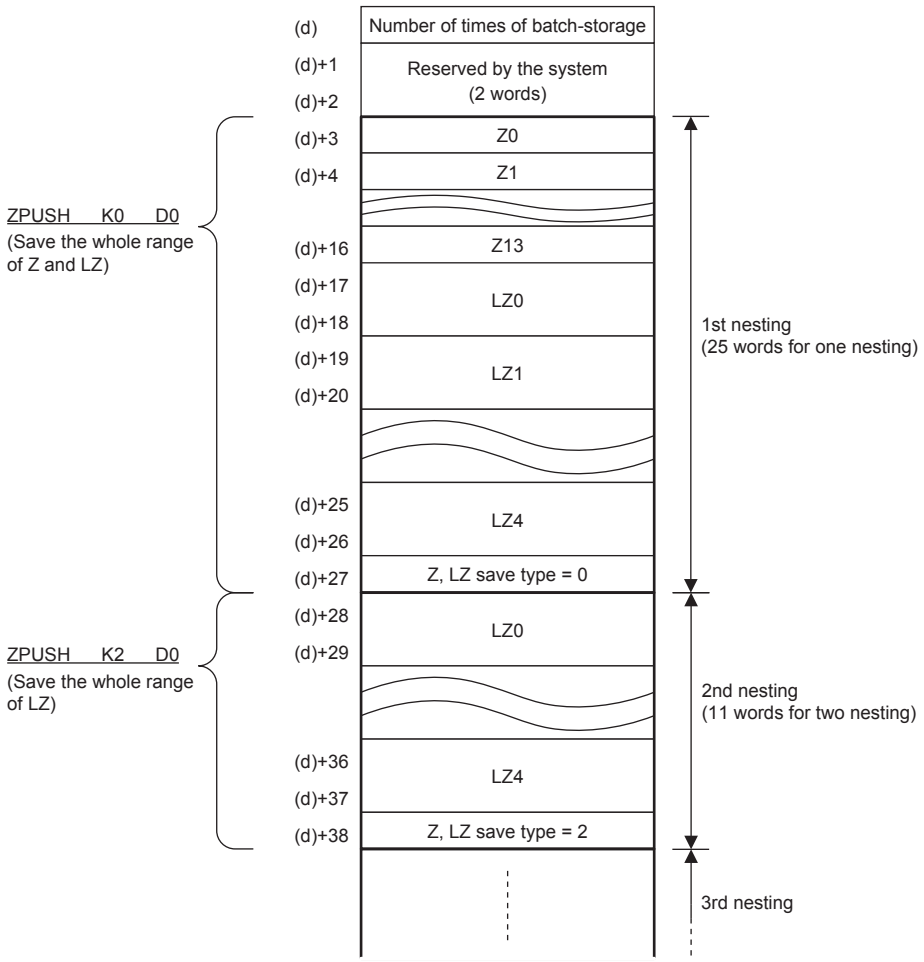
### Processing details

- These instructions save the contents of index registers and long index registers within the range specified by (s) in the devices specified by (d) and later. The type of the index register or long index register saved is stored in the end of the saved data.
- When the contents of the index registers and long index registers are saved, "1" is added to (d).
- The following shows values specified by (s) and the index register or long index register to be saved.

| (s) value | Z or LZ to be saved |
|---|---|
| 0 | Z, LZ (whole range) |
| 1 | Z (whole range) |
| 2 | LZ (whole range) |

- The selected data of index register/long index register return instructions (ZPOP(P) instructions) are used to return the data. The selected data of index register/long index register save instructions (ZPUSH(P) instructions) and the selected data of index register/long index register return instructions (ZPOP(P) instructions) can be used in pairs and to adopt a nesting structure. (☞ Page 515 Returning the selected data of the index register and long index register)
- When a nesting structure is adopted, the areas to be used are added to (d) and later every time the selected data of index register/long index register save instructions (ZPUSH(P) instructions) are executed. Check the number of index registers and long index registers by SD300 and SD302, and secure the areas for the number of instructions to be used in advance.

• The following shows the areas of (d) and later used for the instructions (when Z0 to 13 and LZ0 to 4 are used).




## Precautions

- When a nesting structure is not adopted, clear (d) before executing the ZPUSH(P) instructions.
- When a nesting structure is adopted, clear (d) before executing the first ZPUSH(P) instructions.
- When the ZPOP(P) instructions are used to return the data of index registers, use the ZPOP(P) instructions corresponding to the ZPUSH(P) instructions that were used for saving the data.
  ZPUSH(P) (One setting data) → ZPOP(P) (One setting data)
  ZPUSH(P) (Two setting data) → ZPOP(P) (Two setting data)
- Do not change the values of (d)+1 and (d)+2 because they are used by the system. Do not change the values of the Z and LZ save types stored in the devices specified by (d) and later because they are used by the system.
- Secure the areas so that the save destination specified by (d) does not exceed the device range.

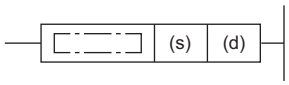
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of points used in (d) or later exceeds the range of the target device/label area. |
| 3405H | A value other than 0 to 2 is specified in (s). |
| | When the number of index registers is 0, "1" is specified in (s). |
| | When the number of long index registers is 0, "2" is specified in (s). |

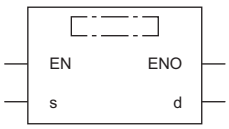
# Returning the selected data of the index register and long index register

## ZPOP(P)

These instructions read the data saved in the devices specified by (d) and later to index registers and long index registers.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ZPOP_2(EN,s,d);<br>ENO:=ZPOPP_2(EN,s,d); |

**FBD/LD**




("ZPOP_2, ZPOPP_2" enters ☐.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Dummy | — | 16-bit unsigned binary | ANY16 |
| (d) | Head device number for returning the data of index registers | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | — | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

### Processing details

- These instructions read the data saved in the devices specified by (d) and later to index registers and long index registers.
- When the saved contents of the index registers and long index registers are read, "1" is subtracted from (d).
- The data specified by (s) is regarded as dummy data and ignored.

### Operation error

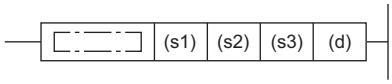
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The range of points used in (d) or later exceeds the range of the target device/label area. |
| 3405H | (d) is 0 or negative. |

# 8.10 Data Control Instruction

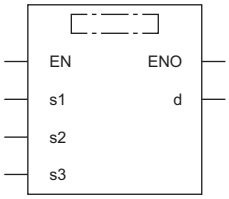
## Upper and lower limit control of 16-bit binary data

### LIMIT(P)(_U)

These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2).

| Ladder diagram | Structured text[*1] | |
|---|---|---|
|  | ENO:=LIMITP(EN,s1,s2,s3,d); | ENO:=LIMITP_U(EN,s1,s2,s3,d); |

**FBD/LD[*1]**




*1 The LIMIT and LIMIT_U instructions are not supported by the ST language and the FBD/LD language. Use LIMIT of the standard function.
☞ Page 901 LIMIT(_E)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | LIMIT(P) | Lower limit value (minimum output value) | -32768 to 32767 | 16-bit signed binary | ANY16_S |
| | LIMIT(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | LIMIT(P) | Upper limit value (maximum output value) | -32768 to 32767 | 16-bit signed binary | ANY16_S |
| | LIMIT(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s3) | LIMIT(P) | Input value controlled by the upper and lower limit values | -32768 to 32767 | 16-bit signed binary | ANY16_S |
| | LIMIT(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | LIMIT(P) | Head device number storing the output value controlled by the upper and lower limit values | — | 16-bit signed binary | ANY16_S |
| | LIMIT(P)_U | | | 16-bit unsigned binary | ANY16_U |

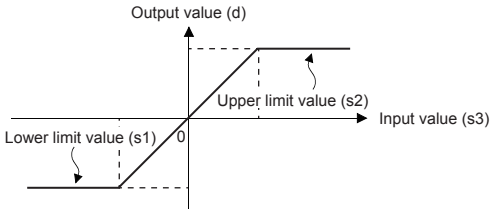
##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

## Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

| Condition | Output value |
|---|---|
| Lower limit value (s1) > Input value (s3) | Lower limit value (s1) |
| Upper limit value (s2) < Input value (s3) | Upper limit value (s2) |
| Lower limit value (s1) ≤ Input value (s3) ≤ Upper limit value (s2) | Input value (s3) |




- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

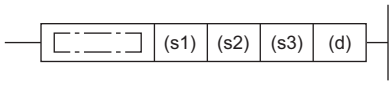
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

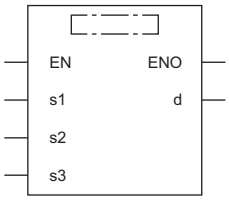
# Upper and lower limit control of 32-bit binary data

## DLIMIT(P)(_U)

These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2).

| Ladder diagram | Structured text[*1] | |
|---|---|---|
|  | ENO:=DLIMITP(EN,s1,s2,s3,d); | ENO:=DLIMITP_U(EN,s1,s2,s3,d); |

**FBD/LD[*1]**




*1 The DLIMIT and DLIMIT_U instructions are not supported by the ST language and the FBD/LD language. Use LIMIT of the standard function.
☞ Page 901 LIMIT(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DLIMIT(P) | Lower limit value (minimum output value) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DLIMIT(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DLIMIT(P) | Upper limit value (maximum output value) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DLIMIT(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s3) | DLIMIT(P) | Input value controlled by the upper and lower limit values | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DLIMIT(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DLIMIT(P) | Head device number storing the output value controlled by the upper and lower limit values | — | 32-bit signed binary | ANY32_S |
| | DLIMIT(P)_U | | | 32-bit unsigned binary | ANY32_U |

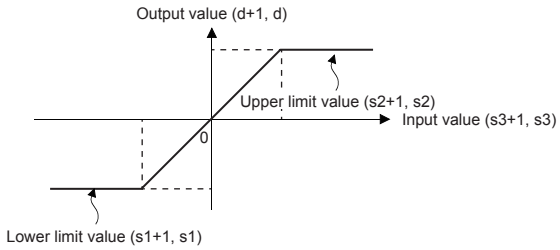
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

## Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

| Condition | Output value |
|---|---|
| Lower limit value ((s1), (s1)+1) > Input value ((s3), (s3)+1) | Lower limit value ((s1), (s1)+1) |
| Upper limit value ((s2), (s2)+1) < Input value ((s3), (s3)+1) | Upper limit value ((s2), (s2)+1) |
| Lower limit value ((s1), (s1)+1) ≤ Input value ((s3), (s3)+1) ≤ Upper limit value ((s2), (s2)+1) | Input value ((s3), (s3)+1) |




- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

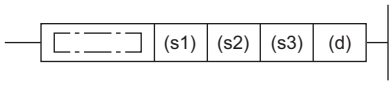
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

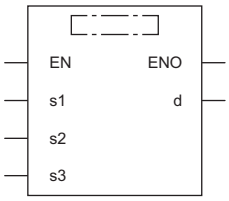
# Dead band control of 16-bit binary data

## BAND(P)(_U)

These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values of the dead band specified by (s1) and (s2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=BAND(EN,s1,s2,s3,d);<br>ENO:=BANDP(EN,s1,s2,s3,d); | ENO:=BAND_U(EN,s1,s2,s3,d);<br>ENO:=BANDP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | BAND(P) | Lower limit value of the dead band (no-output band) | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BAND(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | BAND(P) | Upper limit value of the dead band (no-output band) | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BAND(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s3) | BAND(P) | Input value controlled by the dead band | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | BAND(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | BAND(P) | Head device number for storing the output value controlled by the dead band | — | 16-bit signed binary | ANY16_S |
| | BAND(P)_U | | | 16-bit unsigned binary | ANY16_U |

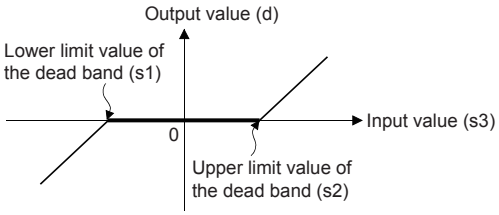
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

## Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values of the dead band specified by (s1) and (s2). The output value is controlled as follows.

| Condition | Output value |
|---|---|
| Lower limit value of the dead band (s1) > Input value (s3) | Input value (s3) - Lower limit value of the dead band (s1) |
| Upper limit value of the dead band (s2) < Input value (s3) | Input value (s3) - Upper limit value of the dead band (s2) |
| Lower limit value of the dead band (s1) ≤ Input value (s3) ≤ Upper limit value of the dead band (s2) | 0 |




- When the output value to be stored in the device specified by (d) is a 16-bit signed binary value and the operation result exceeds the range of -32768 to 32767, the output value is calculated as follows.

**Ex.**

When (s1) is 10 and (s3) is -32768: Output value = -32768-10 = 8000H-000AH = 7FFFH = 32758

- When the output value to be stored in the device specified by (d) is a 16-bit unsigned binary value and the operation result exceeds the range of 0 to 65535, the output value is calculated as follows.

**Ex.**

When (s1) is 100 and (s3) is 50: Output value = 50-100 = 0032H-0064H = FFCEH = 65486

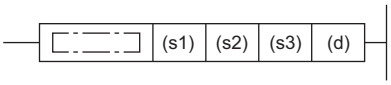
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

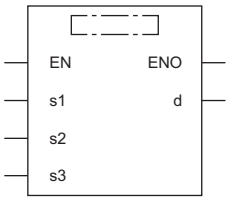
# Dead band control of 32-bit binary data

## DBAND(P)(_U)

These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values of the dead band specified by (s1) and (s2).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DBAND(EN,s1,s2,s3,d);<br>ENO:=DBANDP(EN,s1,s2,s3,d); | ENO:=DBAND_U(EN,s1,s2,s3,d);<br>ENO:=DBANDP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DBAND(P) | Lower limit value of the dead band (no-output band) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBAND(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DBAND(P) | Upper limit value of the dead band (no-output band) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBAND(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s3) | DBAND(P) | Input value controlled by the dead band | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DBAND(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DBAND(P) | Head device number for storing the output value controlled by the dead band | — | 32-bit signed binary | ANY32_S |
| | DBAND(P)_U | | | 32-bit unsigned binary | ANY32_U |

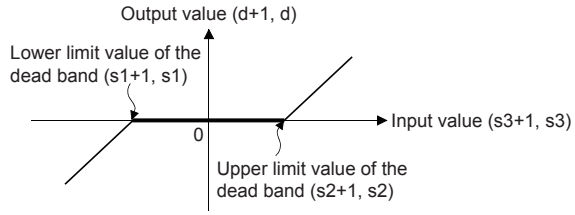
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

## Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values of the dead band specified by (s1) and (s2). The output value is controlled as follows.

| Condition | Output value |
|---|---|
| Lower limit value of the dead band ((s1), (s1)+1) > Input value ((s3), (s3)+1) | Input value ((s3), (s3)+1) - Lower limit value of the dead band ((s1), (s1)+1) |
| Upper limit value of the dead band ((s2), (s2)+1) < Input value ((s3), (s3)+1) | Input value ((s3), (s3)+1) - Upper limit value of the dead band ((s2), (s2)+1) |
| Lower limit value of the dead band ((s1), (s1)+1) ≤ Input value ((s3), (s3)+1) ≤ Upper limit value of the dead band ((s2), (s2)+1) | 0 |




- When the output value to be stored in the device specified by (d) is a 32-bit signed binary value and the operation result exceeds the range of -2147483648 to 2147483647, the output value is calculated as follows.

Ex.

When (s1) and (s1)+1 are 1000, and (s3) and (s3)+1 are -2147483648: Output value = -2147483648-1000 = 80000000H-000003E8H = 7FFFFC18H = 2147482648

- When the output values to be stored in the devices specified by (d) and (d)+1 are 32-bit unsigned binary values and the operation result exceeds the range of 0 to 4294967295, the output value is calculated as follows.

Ex.

When (s1) and (s1)+1 are 100, and (s3) and (s3)+1 are 50: Output value = 50-100 = 00000032H-00000064H = FFFFFFCEH = 4294967246

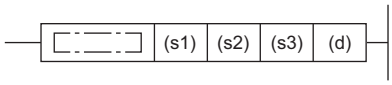
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

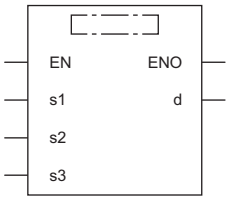
# Zone control of 16-bit binary data

## ZONE(P)(_U)

These instructions add the bias value specified by (s1) or (s2) to the input value specified by (s3), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=ZONE(EN,s1,s2,s3,d);<br>ENO:=ZONEP(EN,s1,s2,s3,d); | ENO:=ZONE_U(EN,s1,s2,s3,d);<br>ENO:=ZONEP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | ZONE(P) | Negative bias value to be added to the input value | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZONE(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | ZONE(P) | Positive bias value to be added to the input value | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZONE(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s3) | ZONE(P) | Input value for performing the zone control | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | ZONE(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (d) | ZONE(P) | Head device number storing the output value controlled by the zone | — | 16-bit signed binary | ANY16_S |
| | ZONE(P)_U | | | 16-bit unsigned binary | ANY16_U |

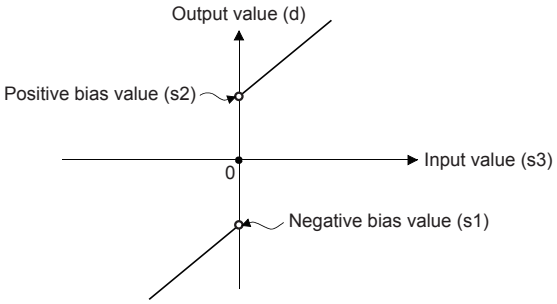
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

## Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (16-bit binary data) specified by (s3), and store the operation result in the device specified by (d). The bias value is controlled as follows.

| Condition | Output value |
|---|---|
| Input value (s3) < 0 | Input value (s3) + Negative bias value (s1) |
| Input value (s3) = 0 | 0 |
| Input value (s3) > 0 | Input value (s3) + Positive bias value (s2) |




- When the output value to be stored in the device specified by (d) is a 16-bit signed binary value and the operation result exceeds the range of -32768 to 32767, the output value is calculated as follows.

**Ex.**

When (s1) is -100 and (s3) is -32768: Output value = -32768+(-100) = 8000H-FF9CH = 7F9CH = 32668

- When the output value to be stored in the device specified by (d) is a 16-bit unsigned binary value and the operation result exceeds the range of 0 to 65535, the output value is calculated as follows.

**Ex.**

When (s2) is 100 and (s3) is 65535: Output value =65535+100 = FFFFH-0064H = 0063H = 99

- When the ZONE(P)_U instructions are used, (s1) is regarded as dummy data and ignored.

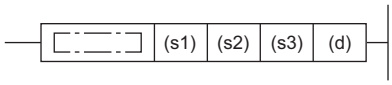
## Operation error

There is no operation error.

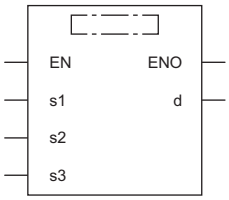
# Zone control of 32-bit binary data

## DZONE(P)(_U)

These instructions add the bias value specified by (s1) or (s2) to the input value specified by (s3), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DZONE(EN,s1,s2,s3,d);<br>ENO:=DZONEP(EN,s1,s2,s3,d); | ENO:=DZONE_U(EN,s1,s2,s3,d);<br>ENO:=DZONEP_U(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DZONE(P) | Negative bias value to be added to the input value | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZONE(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DZONE(P) | Positive bias value to be added to the input value | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZONE(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s3) | DZONE(P) | Input value for performing the zone control | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DZONE(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (d) | DZONE(P) | Head device number storing the output value controlled by the zone | — | 32-bit signed binary | ANY32_S |
| | DZONE(P)_U | | | 32-bit unsigned binary | ANY32_U |

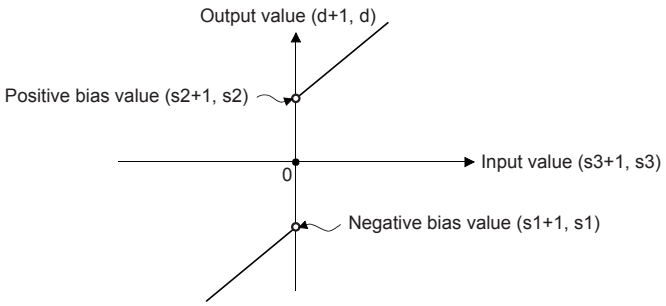
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

## Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (32-bit binary data) specified by (s3), and store the operation result in the device specified by (d). The bias value is controlled as follows.

| Condition | Output value |
|---|---|
| Input value ((s3), (s3)+1) < 0 | Input value ((s3), (s3)+1) + Negative bias value (s1), (s1)+1 |
| Input value ((s3), (s3)+1) = 0 | 0 |
| Input value ((s3), (s3)+1) > 0 | Input value ((s3), (s3)+1) + Positive bias value (s2), (s2)+1 |




- When the output values to be stored in the devices specified by (d) and (d)+1 are 32-bit signed binary values and the operation result exceeds the range of -2147483648 to 2147483647, the output value is calculated as follows.

**Ex.**

When (s1) and (s1)+1 are -1000, and (s3) and (s3)+1 are -2147483648: Output value = -2147483648+(-1000) = 80000000H-FFFFFC18H = 7FFFFC18H = 2147482648

- When the output values to be stored in the devices specified by (d) and (d)+1 are 32-bit unsigned binary values and the operation result exceeds the range of 0 to 4294967295, the output value is calculated as follows.

**Ex.**

When (s2) and (s2)+1 are 1000, and (s3) and (s3)+1 are 4294967295: Output value =4294967295+1000 = FFFFFFFFH-00003E8H = 000003E7H = 999

- When the DZONE(P)_U instructions are used, (s1) and (s1)+1 are regarded as dummy data and ignored.

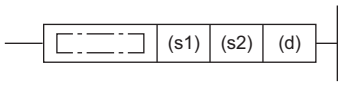
## Operation error

There is no operation error.

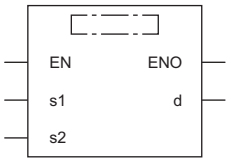
# Scaling 16-bit binary data (point coordinates)

## SCL(P)(_U)

These instructions process the scaling conversion data (in 16-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=SCL(EN,s1,s2,d);<br>ENO:=SCLP(EN,s1,s2,d); | ENO:=SCL_U(EN,s1,s2,d);<br>ENO:=SCLP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | SCL(P) | Input value used in scaling or head device number storing the input value | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | SCL(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | SCL(P) | Head device number where the scaling conversion data is stored | — | 16-bit signed binary*1 | ANY16_S |
| | SCL(P)_U | | | 16-bit unsigned binary*1 | ANY16_U |
| (d) | SCL(P) | Head device number storing the output value controlled by scaling | — | 16-bit signed binary | ANY16_S |
| | SCL(P)_U | | | 16-bit unsigned binary | ANY16_U |

*1 The number of coordinate points of (s2) is 16-bit unsigned binary data.

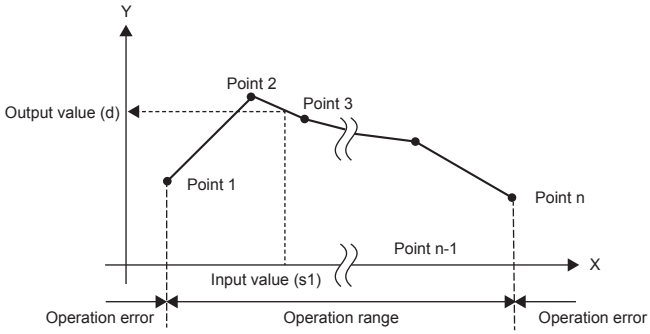
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

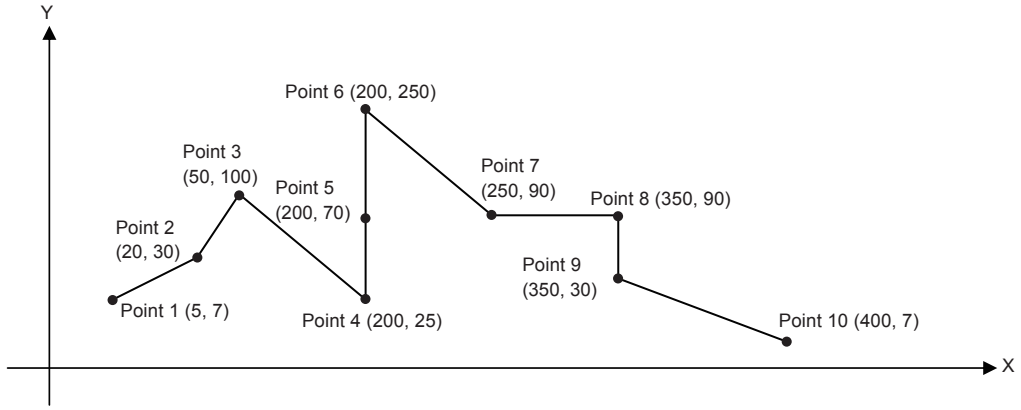
- These instructions process the scaling conversion data (in 16-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device number specified by (d). The scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

| Setting item ("n" indicates the number of coordinate points specified by (s2).) | | Device assignment |
|---|---|---|
| Number of coordinate points | | (s2) |
| Point 1 | X coordinate | (s2)+1 |
| | Y coordinate | (s2)+2 |
| Point 2 | X coordinate | (s2)+3 |
| | Y coordinate | (s2)+4 |
| ⋮ | | |
| Point n | X coordinate | (s2)+2n-1 |
| | Y coordinate | (s2)+2n |




- If the operation result is not an integer, the number in the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in the ascending order.
- Set (s1) within the scaling conversion data range (device value of (s2)).
- If the same X coordinate is specified by multiple points, the Y coordinate value of the point whose number is the largest is output.
- Set the number of coordinate points for the scaling conversion data within the range of 1 to 65535.
- Setting example of the conversion table for scaling

  In the case of the conversion characteristics for scaling shown in the figure below, set each value as shown in the following data table.




| Setting item | | Setting device and setting contents | | | Remarks |
|---|---|---|---|---|---|
| | | When R0 is specified in (s2) | | Setting details | |
| Number of coordinate points | | (s2) | R0 | K10 | |
| Point 1 | X coordinate | (s2)+1 | R1 | K5 | |
| | Y coordinate | (s2)+2 | R2 | K7 | |
| Point 2 | X coordinate | (s2)+3 | R3 | K20 | |
| | Y coordinate | (s2)+4 | R4 | K30 | |
| Point 3 | X coordinate | (s2)+5 | R5 | K50 | |
| | Y coordinate | (s2)+6 | R6 | K100 | |
| Point 4 | X coordinate | (s2)+7 | R7 | K200 | When coordinates are specified using three points in this way, the output value can be set to an intermediate value.<br>In this example, the output value (intermediate value) is specified by the Y coordinate of the point 5.<br>Even if the X coordinate is the same at three points or more, the value at the second point is output. |
| | Y coordinate | (s2)+8 | R8 | K25 | |
| Point 5 | X coordinate | (s2)+9 | R9 | K200 | |
| | Y coordinate | (s2)+10 | R10 | K70 | |
| Point 6 | X coordinate | (s2)+11 | R11 | K200 | |
| | Y coordinate | (s2)+12 | R12 | K250 | |
| Point 7 | X coordinate | (s2)+13 | R13 | K250 | |
| | Y coordinate | (s2)+14 | R14 | K90 | |
| Point 8 | X coordinate | (s2)+15 | R15 | K350 | When coordinates are specified using two points in this way, the output value is the Y coordinate at the next point.<br>In this example, the output value is specified by the Y coordinate of the point 9. |
| | Y coordinate | (s2)+16 | R16 | K90 | |
| Point 9 | X coordinate | (s2)+17 | R17 | K350 | |
| | Y coordinate | (s2)+18 | R18 | K30 | |
| Point 10 | X coordinate | (s2)+19 | R19 | K400 | |
| | Y coordinate | (s2)+20 | R20 | K7 | |

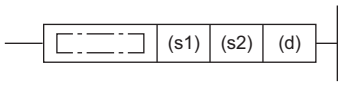
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The Xn data is not set in the ascending order in the data table.<br>However, the instructions before the occurrence of an error are executed. |
| | The input value specified by (s1) is out of the range for the set scaling conversion data. |
| | The value in the middle of operation exceeds the 32-bit data range.<br>In this case, verify that the distance between points is not "65535" or more.<br>If the distance is "65535" or more, reduce the distance between points. |
| | The number of coordinate points from the device specified by (s2) is 0 or less. |

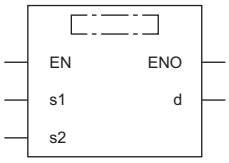
# Scaling 32-bit binary data (point coordinates)

## DSCL(P)(_U)

These instructions process the scaling conversion data (in 32-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DSCL(EN,s1,s2,d);<br>ENO:=DSCLP(EN,s1,s2,d); | ENO:=DSCL_U(EN,s1,s2,d);<br>ENO:=DSCLP_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DSCL(P) | Input value used in scaling or head device number storing the input value | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DSCL(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DSCL(P) | Head device number where the scaling conversion data is stored | — | 32-bit signed binary*1 | ANY32_S |
| | DSCL(P)_U | | | 32-bit unsigned binary*1 | ANY32_U |
| (d) | DSCL(P) | Head device number storing the output value controlled by scaling | — | 32-bit signed binary | ANY32_S |
| | DSCL(P)_U | | | 32-bit unsigned binary | ANY32_U |

*1 The numbers of coordinate points of (s2)+1 and (s2) are 32-bit unsigned binary data.

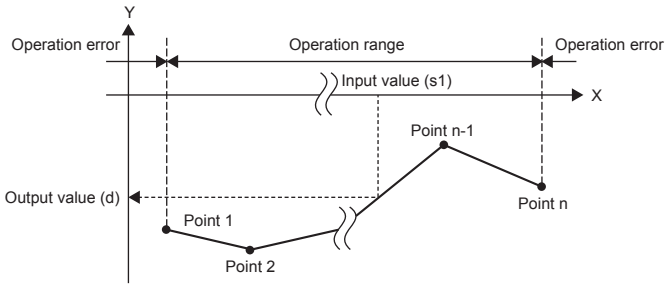
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

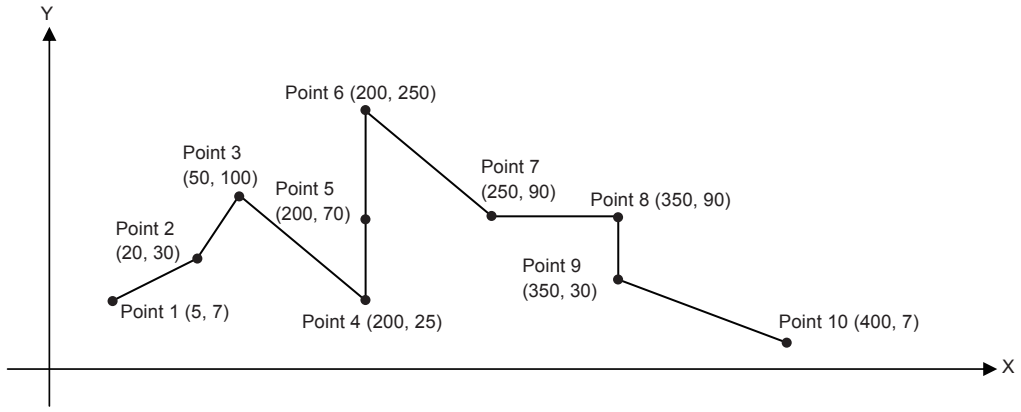
- These instructions process the scaling conversion data (in 32-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device number specified by (d). The scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

| Setting item ("n" indicates the number of coordinate points specified by (s2).) | | Device assignment |
|---|---|---|
| Number of coordinate points | | (s2)+1, (s2) |
| Point 1 | X coordinate | (s2)+3, (s2)+2 |
| | Y coordinate | (s2)+5, (s2)+4 |
| Point 2 | X coordinate | (s2)+7, (s2)+6 |
| | Y coordinate | (s2)+9, (s2)+8 |
| ⋮ | | |
| Point n | X coordinate | (s2)+4n-1, (s2)+4n-2 |
| | Y coordinate | (s2)+4n+1, (s2)+4n |




- If the operation result is not an integer, the number in the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in the ascending order.
- Set (s1) within the scaling conversion data range (device values of (s2) and (s2)+1).
- If the same X coordinate is specified by multiple points, the Y coordinate value of the point whose number is the largest is output.
- Set the number of coordinate points for the scaling conversion data within the range of 1 to 4294967295.
- Setting example of the conversion table for scaling

  In the case of the conversion characteristics for scaling shown in the figure below, set each value as shown in the following data table.




| Setting item | | Setting device and setting contents | | | Remarks |
|---|---|---|---|---|---|
| | | When R0 is specified in (s2) | | Setting details | |
| Number of coordinate points | | (s2)+1, (s2) | R1, R0 | K10 | |
| Point 1 | X coordinate | (s2)+3, (s2)+2 | R3, R2 | K5 | |
| | Y coordinate | (s2)+5, (s2)+4 | R5, R4 | K7 | |
| Point 2 | X coordinate | (s2)+7, (s2)+6 | R7, R6 | K20 | |
| | Y coordinate | (s2)+9, (s2)+8 | R9, R8 | K30 | |
| Point 3 | X coordinate | (s2)+11, (s2)+10 | R11, R10 | K50 | |
| | Y coordinate | (s2)+13, (s2)+12 | R13, R12 | K100 | |
| Point 4 | X coordinate | (s2)+15, (s2)+14 | R15, R14 | K200 | When coordinates are specified using three points in this way, the output value can be set to an intermediate value.<br>In this example, the output value (intermediate value) is specified by the Y coordinate of the point 5.<br>Even if the X coordinate is the same at three points or more, the value at the second point is output. |
| | Y coordinate | (s2)+17, (s2)+16 | R17, R16 | K25 | |
| Point 5 | X coordinate | (s2)+19, (s2)+18 | R19, R18 | K200 | |
| | Y coordinate | (s2)+21, (s2)+20 | R21, R20 | K70 | |
| Point 6 | X coordinate | (s2)+23, (s2)+22 | R23, R22 | K200 | |
| | Y coordinate | (s2)+25, (s2)+24 | R25, R24 | K250 | |
| Point 7 | X coordinate | (s2)+27, (s2)+26 | R27, R26 | K250 | |
| | Y coordinate | (s2)+29, (s2)+28 | R29, R28 | K90 | |
| Point 8 | X coordinate | (s2)+31, (s2)+30 | R31, R30 | K350 | When coordinates are specified using two points in this way, the output value is the Y coordinate at the next point.<br>In this example, the output value is specified by the Y coordinate of the point 9. |
| | Y coordinate | (s2)+33, (s2)+32 | R33, R32 | K90 | |
| Point 9 | X coordinate | (s2)+35, (s2)+34 | R35, R34 | K350 | |
| | Y coordinate | (s2)+37, (s2)+36 | R37, R36 | K30 | |
| Point 10 | X coordinate | (s2)+39, (s2)+38 | R39, R38 | K400 | |
| | Y coordinate | (s2)+41, (s2)+40 | R41, R40 | K7 | |

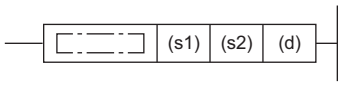
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The Xn data is not set in the ascending order in the data table.<br>However, the instructions before the occurrence of an error are executed. |
| | The input value specified by (s1) is out of the range for the set scaling conversion data. |
| | The value in the middle of operation exceeds the 32-bit data range.<br>In this case, verify that the distance between points is not "65535" or more.<br>If the distance is "65535" or more, reduce the distance between points. |
| | The number of coordinate points from the device specified by (s2) is 0 or less. |

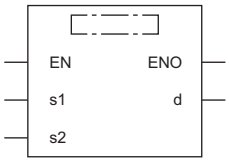
# Scaling 16-bit binary data (XY coordinates)

## SCL2(P)(_U)

These instructions process the scaling conversion data (in 16-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=SCL2(EN,s1,s2,d);<br>ENO:=SCL2P(EN,s1,s2,d); | ENO:=SCL2_U(EN,s1,s2,d);<br>ENO:=SCL2P_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | SCL2(P) | Input value used in scaling or head device number storing the input value | -32768 to +32767 | 16-bit signed binary | ANY16_S |
| | SCL2(P)_U | | 0 to 65535 | 16-bit unsigned binary | ANY16_U |
| (s2) | SCL2(P) | Head device number where the scaling conversion data is stored | — | 16-bit signed binary*1 | ANY16_S |
| | SCL2(P)_U | | | 16-bit unsigned binary*1 | ANY16_U |
| (d) | SCL2(P) | Head device number storing the output value controlled by scaling | — | 16-bit signed binary | ANY16_S |
| | SCL2(P)_U | | | 16-bit unsigned binary | ANY16_U |

*1 The number of coordinate points of (s2) is 16-bit unsigned binary data.

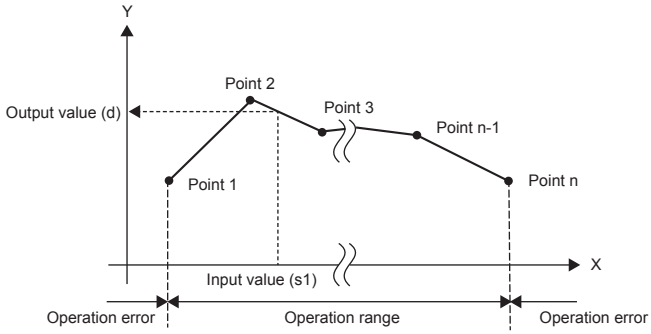
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

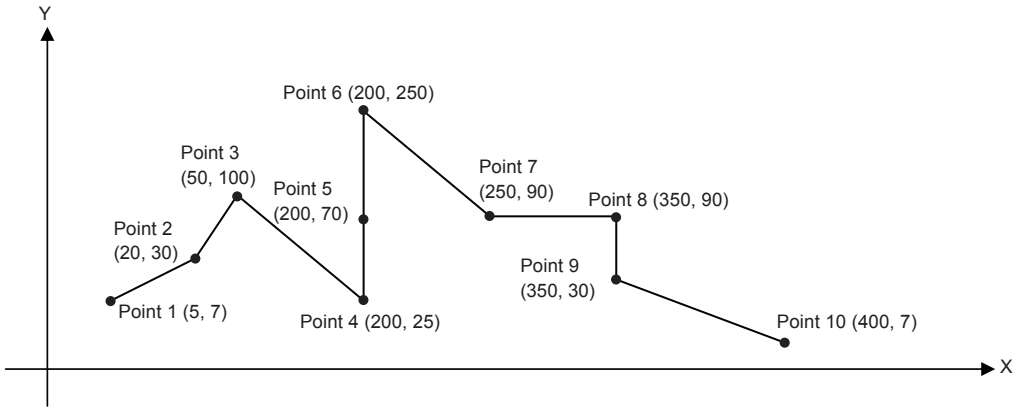
- These instructions process the scaling conversion data (in 16-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device number specified by (d). The scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

| Setting item ("n" indicates the number of coordinate points specified by (s2).) | | Device assignment |
|---|---|---|
| Number of coordinate points | | (s2) |
| X coordinate | Point 1 | (s2)+1 |
| | Point 2 | (s2)+2 |
| | ⋮ | ⋮ |
| | Point n | (s2)+n |
| Y coordinate | Point 1 | (s2)+n+1 |
| | Point 2 | (s2)+n+2 |
| | ⋮ | ⋮ |
| | Point n | (s2)+2n |




- If the operation result is not an integer, the number in the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in the ascending order.
- Set (s1) within the scaling conversion data range (device value of (s2)).
- If the same X coordinate is specified by multiple points, the Y coordinate value of the point whose number is the largest is output.
- Set the number of coordinate points for the scaling conversion data within the range of 1 to 65535.
- Setting example of the conversion table for scaling

  In the case of the conversion characteristics for scaling shown in the figure below, set each value as shown in the following data table.



| Setting item | | Setting device and setting contents | | | Remarks |
|---|---|---|---|---|---|
| | | When R0 is specified in (s2) | | Setting details | |
| Number of coordinate points | | (s2) | R0 | K10 | |
| X coordinate | Point 1 | (s2)+1 | R1 | K5 | |
| | Point 2 | (s2)+2 | R2 | K20 | |
| | Point 3 | (s2)+3 | R3 | K50 | |
| | Point 4 | (s2)+4 | R4 | K200 | Refer to *1. |
| | Point 5 | (s2)+5 | R5 | K200 | |
| | Point 6 | (s2)+6 | R6 | K200 | |
| | Point 7 | (s2)+7 | R7 | K250 | |
| | Point 8 | (s2)+8 | R8 | K350 | Refer to *2. |
| | Point 9 | (s2)+9 | R9 | K350 | |
| | Point 10 | (s2)+10 | R10 | K400 | |
| Y coordinate | Point 1 | (s2)+11 | R11 | K7 | |
| | Point 2 | (s2)+12 | R12 | K30 | |
| | Point 3 | (s2)+13 | R13 | K100 | |
| | Point 4 | (s2)+14 | R14 | K25 | Refer to *1. |
| | Point 5 | (s2)+15 | R15 | K70 | |
| | Point 6 | (s2)+16 | R16 | K250 | |
| | Point 7 | (s2)+17 | R17 | K90 | |
| | Point 8 | (s2)+18 | R18 | K90 | Refer to *2. |
| | Point 9 | (s2)+19 | R19 | K30 | |
| | Point 10 | (s2)+20 | R20 | K7 | |

*1 When coordinates are specified using three points as shown in the points 4, 5 and 6, the output value can be set to an intermediate value.
In this example, the output value (intermediate value) is specified by the Y coordinate of the point 5.
Even if the X coordinate is the same at three points or more, the value at the second point is output.

*2 When coordinates are specified using two points as shown in the points 8 and 9, the output value is the Y coordinate at the next point.
In this example, the output value is specified by the Y coordinate of the point 9.

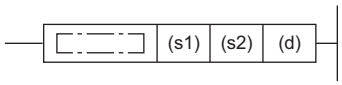
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The Xn data is not set in the ascending order in the data table.<br>However, the instructions before the occurrence of an error are executed. |
| | The input value specified by (s1) is out of the range for the set scaling conversion data. |
| | The value in the middle of operation exceeds the 32-bit data range.<br>In this case, verify that the distance between points is not "65535" or more.<br>If the distance is "65535" or more, reduce the distance between points. |
| | The number of coordinate points from the device specified by (s2) is 0 or less. |

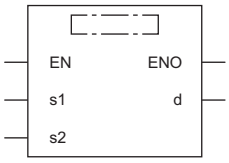
# Scaling 32-bit binary data (XY coordinates)

## DSCL2(P)(_U)

These instructions process the scaling conversion data (in 32-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DSCL2(EN,s1,s2,d);<br>ENO:=DSCL2P(EN,s1,s2,d); | ENO:=DSCL2_U(EN,s1,s2,d);<br>ENO:=DSCL2P_U(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | DSCL2(P) | Input value used in scaling or head device number storing the input value | -2147483648 to +2147483647 | 32-bit signed binary | ANY32_S |
| | DSCL2(P)_U | | 0 to 4294967295 | 32-bit unsigned binary | ANY32_U |
| (s2) | DSCL2(P) | Head device number where the scaling conversion data is stored | — | 32-bit signed binary*1 | ANY32_S |
| | DSCL2(P)_U | | | 32-bit unsigned binary*1 | ANY32_U |
| (d) | DSCL2(P) | Head device number storing the output value controlled by scaling | — | 32-bit signed binary | ANY32_S |
| | DSCL2(P)_U | | | 32-bit unsigned binary | ANY32_U |

*1 The numbers of coordinate points of (s2)+1 and (s2) are 32-bit unsigned binary data.

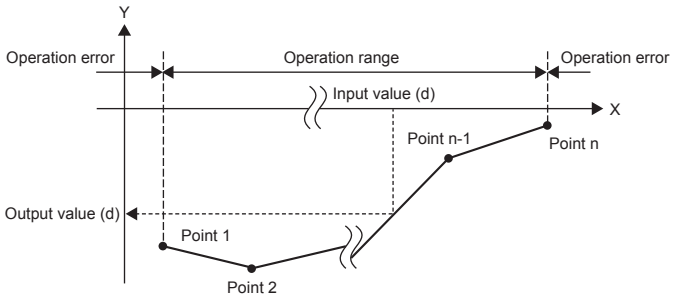
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

## Processing details

- These instructions process the scaling conversion data (in 32-bit data units) specified by (s2) by scaling it based on the input value specified by (s1), and store the operation result in the device number specified by (d). The scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

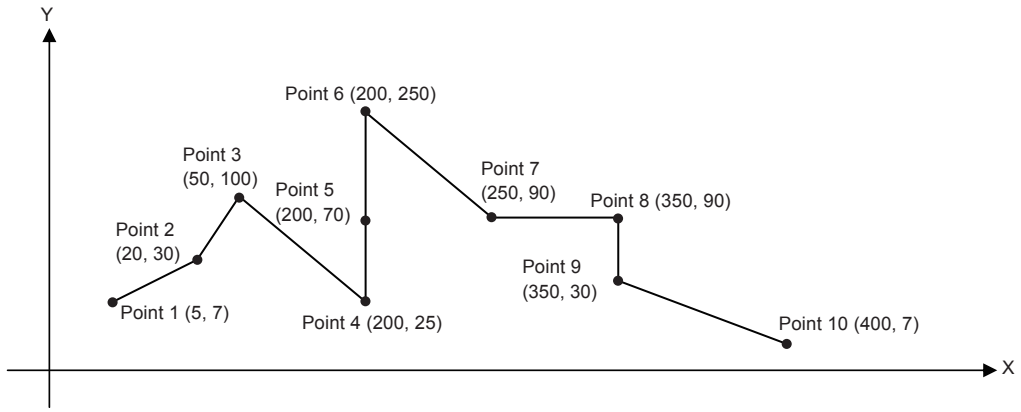
| Setting item ("n" indicates the number of coordinate points specified by (s2).) | | Device assignment |
|---|---|---|
| Number of coordinate points | | (s2)+1, (s2) |
| X coordinate | Point 1 | (s2)+3, (s2)+2 |
| | Point 2 | (s2)+5, (s2)+4 |
| | ⋮ | ⋮ |
| | Point n | (s2)+2n+1, (s2)+2n |
| Y coordinate | Point 1 | (s2)+2n+3, (s2)+2n+2 |
| | Point 2 | (s2)+2n+5, (s2)+2n+4 |
| | ⋮ | ⋮ |
| | Point n | (s2)+4n+1, (s2)+4n |




- If the operation result is not an integer, the number in the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in the ascending order.
- Set (s1) within the scaling conversion data range (device values of (s2) and (s2)+1).
- If the same X coordinate is specified by multiple points, the Y coordinate value of the point whose number is the largest is output.
- Set the number of coordinate points for the scaling conversion data within the range of 1 to 4294967295.

• Setting example of the conversion table for scaling

In the case of the conversion characteristics for scaling shown in the figure below, set each value as shown in the following data table.




| Setting item | | Setting device and setting contents | | | Remarks |
|---|---|---|---|---|---|
| | | When R0 is specified in (s2) | | Setting details | |
| Number of coordinate points | | (s2)+1, (s2) | R1, R0 | K10 | |
| X coordinate | Point 1 | (s2)+3, (s2)+2 | R3, R2 | K5 | |
| | Point 2 | (s2)+5, (s2)+4 | R5, R4 | K20 | |
| | Point 3 | (s2)+7, (s2)+6 | R7, R6 | K50 | |
| | Point 4 | (s2)+9, (s2)+8 | R9, R8 | K200 | Refer to *1. |
| | Point 5 | (s2)+11, (s2)+10 | R11, R10 | K200 | |
| | Point 6 | (s2)+13, (s2)+12 | R13, R12 | K200 | |
| | Point 7 | (s2)+15, (s2)+14 | R15, R14 | K250 | |
| | Point 8 | (s2)+17, (s2)+16 | R17, R16 | K350 | Refer to *2. |
| | Point 9 | (s2)+19, (s2)+18 | R19, R18 | K350 | |
| | Point 10 | (s2)+21, (s2)+20 | R21, R20 | K400 | |
| Y coordinate | Point 1 | (s2)+23, (s2)+22 | R23, R22 | K7 | |
| | Point 2 | (s2)+25, (s2)+24 | R25, R24 | K30 | |
| | Point 3 | (s2)+27, (s2)+26 | R27, R26 | K100 | |
| | Point 4 | (s2)+29, (s2)+28 | R29, R28 | K25 | Refer to *1. |
| | Point 5 | (s2)+31, (s2)+30 | R31, R30 | K70 | |
| | Point 6 | (s2)+33, (s2)+32 | R33, R32 | K250 | |
| | Point 7 | (s2)+35, (s2)+34 | R35, R34 | K90 | |
| | Point 8 | (s2)+37, (s2)+36 | R37, R36 | K90 | Refer to *2. |
| | Point 9 | (s2)+39, (s2)+38 | R39, R38 | K30 | |
| | Point 10 | (s2)+41, (s2)+40 | R41, R40 | K7 | |

*1 When coordinates are specified using three points as shown in the points 4, 5 and 6, the output value can be set to an intermediate value.
In this example, the output value (intermediate value) is specified by the Y coordinate of the point 5.
Even if the X coordinate is the same at three points or more, the value at the second point is output.

*2 When coordinates are specified using two points as shown in the points 8 and 9, the output value is the Y coordinate at the next point.
In this example, the output value is specified by the Y coordinate of the point 9.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The Xn data is not set in the ascending order in the data table.<br>However, the instructions before the occurrence of an error are executed. |
| | The input value specified by (s1) is out of the range for the set scaling conversion data. |
| | The value in the middle of operation exceeds the 32-bit data range.<br>In this case, verify that the distance between points is not "65535" or more.<br>If the distance is "65535" or more, reduce the distance between points. |
| | The number of coordinate points from the device specified by (s2) is 0 or less. |

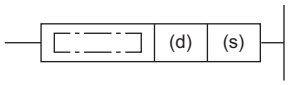
# 8.11 Special Timer Instruction

## Teaching timer

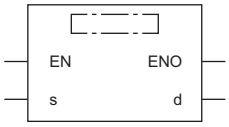
### TTMR

This instruction measures the period of time in which TTMR instruction is ON.

Use this instruction to adjust the set value of a timer by a pushbutton switch.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TTMR(EN,s,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Device storing the teaching data | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 2) |
| (s) | Magnification applied to the teaching data | 0 to 2 | 16-bit signed binary | ANY16 |

##### ■Applicable devices

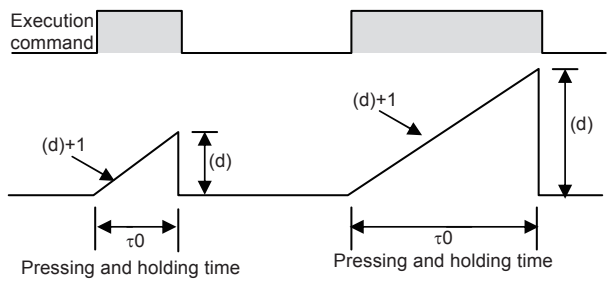
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

##### ■Control data

| Operand: (d) | | | |
|---|---|---|---|
| Device | Description | Setting range | Set by |
| +0 | Teaching time | — | System |
| +1 | Current value of the pressing and holding time | — | System |

#### Processing details

- This instruction measures the period of time to press and hold the command input (pushbutton switch) in 1-second units, multiplies the measured value by the magnification ($10^{s}$) which is specified by (s), and stores it in the device specified by (d).



- The table below shows the actual value indicated by (d) depending on the magnification specified by (s) and the pressing and holding time τ0.

| (s) | Magnification | (d) |
|---|---|---|
| K0 | τ0 | (d)×1 |
| K1 | 10τ0 | (d)×10 |
| K2 | 100τ0 | (d)×100 |

## Precautions

- When the command contact turns from on to off, the current value (d)+1 of the pressing and holding time is cleared, and the teaching time (d) will not change any more.
- Two devices are occupied from a device specified as the teaching time (d). Make sure that such devices are not used in other controls for the machine.

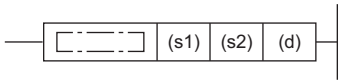
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (s) is outside the following range.<br>0 to 2 |

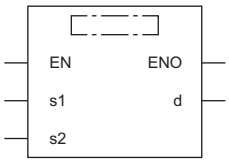
# Special function timer

## STMR

This instruction uses the four devices from the device specified by (d) to perform four types of timer output.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=STMR(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Used timer number (operates as a 100 ms timer) | — | Device name | ANY16 |
| (s2) | Timer set value | 1 to 32767 | 16-bit signed binary | ANY16 |
| (d) | Start bit number to be output | — | Bit | ANYBIT_ARRAY<br>(Number of elements: 4) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

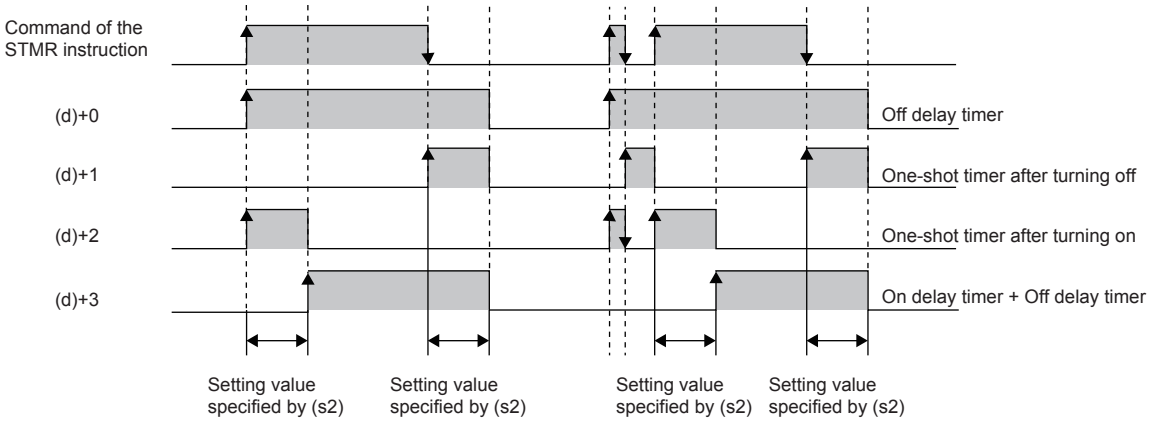
*1 Only T can be used.

#### ■Control data

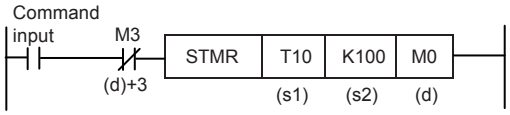
| Operand: (d) | | | |
|---|---|---|---|
| **Device** | **Description** | **Setting range** | **Set by** |
| +0 | Off delay timer output:<br>Turns on at the rising edge of the command of the STMR instruction and turns off when the time specified by (s2) elapses after the falling edge. | — | System |
| +1 | One-shot timer output after turning off:<br>Turns on at the falling edge of the command of the STMR instruction and turns off when the time specified by (s2) elapses. | — | System |
| +2 | One-shot timer output after turning on<br>Turns on at the rising edge of the command of the STMR instruction and turns off when the command of the STMR instruction is turned off or when the time specified by (s2) elapses. | — | System |
| +3 | On delay timer + Off delay timer output:<br>Turns on at the falling edge of the timer coil and turns off when the time specified by (s2) elapsed after the falling edge of the command of the STMR instruction. | — | System |

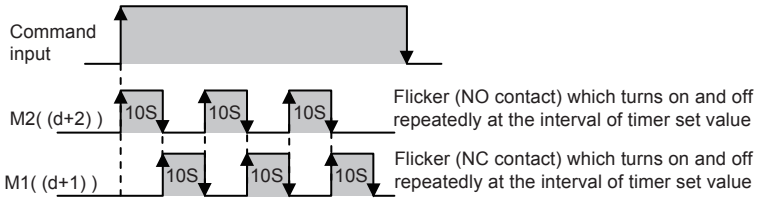
## Processing details

- This instruction uses the four devices from the device specified by (d) to perform four types of timer output.




- The flickering effect is produced using (d)+1 and (d)+2 with the following program, which turns on/off at the normally closed contact of (d)+3 (T10 is assigned to (s1), K100 is assigned to (s2), and M0 is assigned to (d)).




- A value in the range of 0 to 32767 (0 to 3276.7 seconds) can be specified in (s2).

## Precautions

- The timer number specified in this instruction cannot be used in other general circuits (such as OUT instruction). If the timer number is used in other general circuits, the timer malfunctions.
- The timer specified by (s1) starts counting as a 100 ms timer on the rising edge of the command contact.
- Four devices are occupied from a device specified in (d). Make sure that such devices are not used in other controls for the machine.
- If the command contact is turned off, (d), (d)+1, and (d)+3 turn off when the set time elapses. (d)+2 and the timer (s1) are immediately reset.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (s2) is outside the following range.<br>1 to 32767 |

# 8.12 Special Counter Instruction

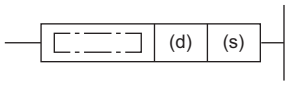
## Signed 32-bit bi-directional counters

### UDCNTF

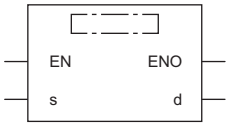
This instruction increments the current value of the counter specified by (d) by 1 when the operation result up to UDCNTF instruction changes from OFF to ON, and when the counter reaches the end of its count, NO contact becomes turns ON and NC contact becomes turns OFF.

When the long counter specified by (d) is a high-speed counter, up-counting and down-counting are enabled.

For details on the high-speed counter, refer to MELSEC iQ-F FX5 User's Manual. (Application).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:= UDCNTF(EN, s, d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Long counter number | — | Device name | ANY32 |
| (s) | Long counter set value | -2147483648 to 2147483647 | 32-bit signed binary | ANY32_S |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | — | — | — | ○ | — | — | — | — | — | — |
| (d) | — | — | — | ○*1 | ○ | — | — | — | — | ○*2 | — | — | — |

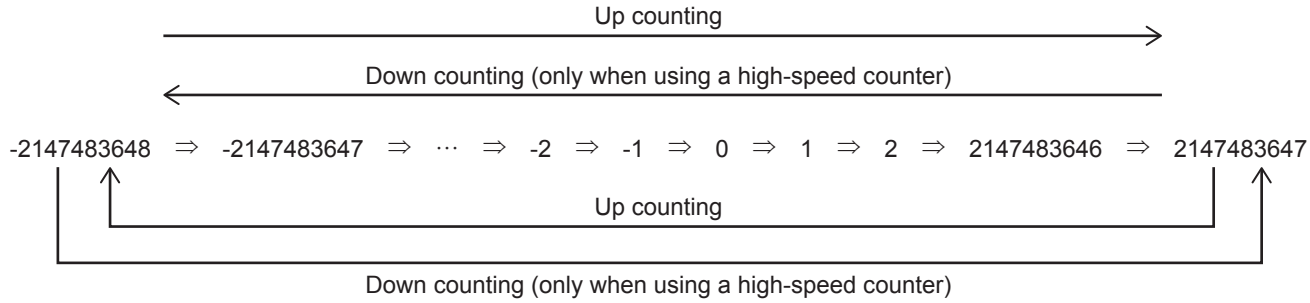
*1 T, ST, C cannot be used.

*2 Only decimal constant (K) can be used.

#### Processing details

- This instruction increments the current value of the counter specified by (d) by 1 when the operation result up to UDCNTF instruction changes from OFF to ON, and when the counter reaches the end of its count, NO contact becomes turns ON and NC contact becomes turns OFF.
- When the long counter specified by (d) is a high-speed counter, up-counting and down-counting are enabled.
- During up-counting, the output contact turns ON when the current value changes from a value less than the set value to a value not less than the set value.
- During down-counting (only when using a high-speed counter for (d)), the output contact turns OFF when the current value changes from a value not less than the set value to a value equivalent to the “set value -1” or less.
- Counting is continued also when the drive contact changes from OFF to ON after the output contact changes. (Only when specify other than a high-speed counter for (d))
- When unsigned (0 to 4294967295) is assigned to the LC, the OUT LC instruction is used. For the OUT LC instruction, refer to ☞ Page 114 OUT LC.

- The current value operates as a ring counter.




## Precautions

- The last number of a word device cannot be input as the long counter set value.
- Indirect specification cannot be input as the long counter set value.

## Operation error

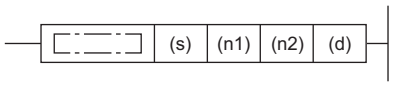
There is no operation error.

# 8.13 Shortcut Control Instruction

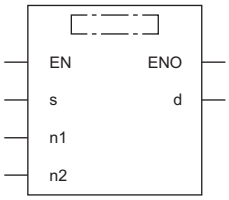
## Rotary table shortest direction control

### ROTC

This instruction is suitable for efficient control of the rotary table for putting/taking a product on/off the rotary table.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ROTC(EN,s,n1,n2,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

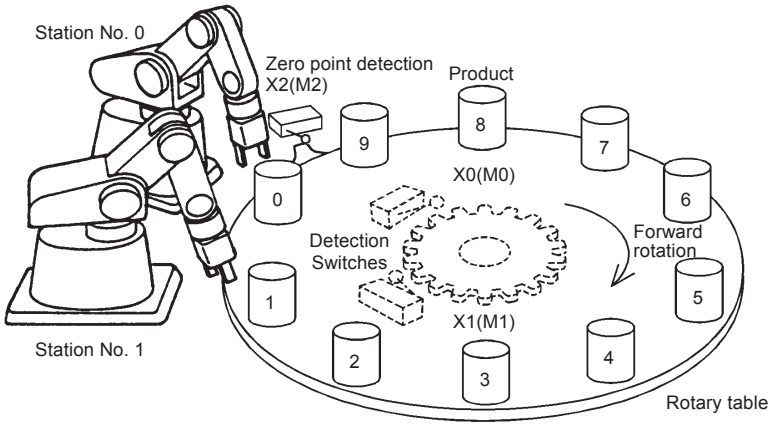
| Operand | Description | | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | Registers specifying the calling condition (Set them in advance using the transfer instruction.) | (s)+0: Works as a register for counting. | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| | | (s)+1: Sets the station No. to be called. | | | |
| | | (s)+2: Sets the product No. to be called. | | | |
| (n1) | Number of divisions | | 2 to 32767 | 16-bit unsigned binary | ANY16 |
| (n2) | Number of low-speed sections | | 0 to 32767 | 16-bit unsigned binary | ANY16 |
| (d) | Registers (bit devices) specifying the calling condition (Construct an internal contact circuit in advance which is driven by the input signal (X).) | (d): A phase signal | — | Bit | ANY16_ARRAY (Number of elements: 3) |
| | | (d)+1: B phase signal | | | |
| | | (d)+2: Zero point detection signal | | | |
| | | (d)+3: Forward rotation at high-speed | | | |
| | | (d)+4: Forward rotation at low-speed | | | |
| | | (d)+5: Stop | | | |
| | | (d)+6: Backward rotation at low-speed | | | |
| | | (d)+7: Backward rotation at high-speed | | | |

##### ■Applicable devices

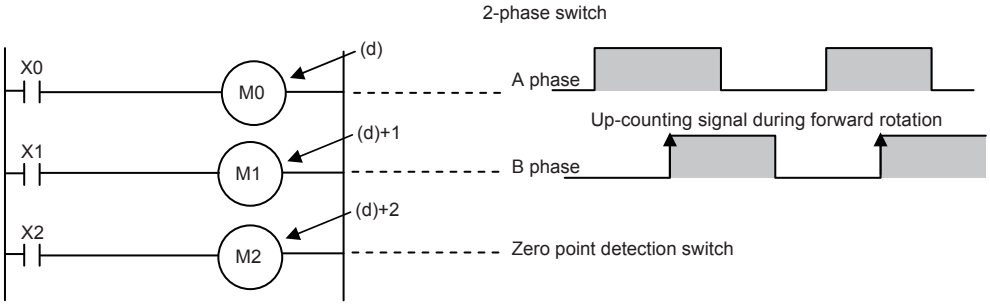
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | — | — | — | — | — | — | — | — | — | — |

## Processing details

- The table rotation is controlled by conditions of "n2", (s), and (d) so that a product can be efficiently put on or taken off the rotary table divided into "n1" (=10) sections as shown in the figure below. When the following conditions are specified, forward/backward rotation and high-speed/low-speed/stop are output to (d)+3 to (d)+7.




- Provide a 2-phase switch (X0 and X1) for detecting the rotation direction (forward or backward) of the table and the switch X2 which turns ON when the product No. 0 reaches the station No. 0. X0 to X2 are replaced with internal contacts of (d) to (d)+2. Any head device number can be specified by X or (d).




- The counter (s) detects which product number is located at the station No. 0.
- Set the station No. to be called in (s)+1.
- Set the product No. to be called in (s)+2.
- Specify the number of divisions (n1) of the table, and number of low-speed sections (n2).

## Precautions

- When the command input is set to ON and this instruction is executed, the result will be automatically output to (d)+3 to (d)+7. When the command input is set to OFF, (d)+3 to (d)+7 are set to OFF accordingly.
- For example, when the rotation detection signal ((d) to (d)+2) is activated 10 times in one division, set a value multiplied by "10" to each division, station No. to be called and product No. to be called. As a result, an intermediate value of the division number can be set to a low-speed section.
- When the zero point detection signal (M2) turns ON while the command input is ON, the contents of the register for counting (s) are cleared to "0". This clear operation should be executed before starting the operation.
- Up to four ROTC instructions can be used simultaneously.

## Operation error

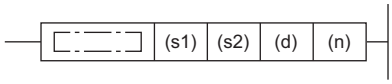
| Error code (SD0/SD8067) | Description |
|---|---|
| 1811H | The number of the ROTC instructions which are used simultaneously exceeds four. |
| 2820H | The device range specified by (s) exceeds the corresponding device range. |
| | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n1) is outside the following range.<br>2 to 32767 |
| | The value specified by (n2) is outside the following range.<br>0 to 32767 |
| | The value specified by (n1) or (n2) is in the following condition.<br>(n1) < (n2) |
| | Either (s), (s)+1, or (s)+2 is negative. |
| | Either (s), (s)+1, or (s)+2 is equal to (n1) or larger. |

# 8.14 Ramp Signal Instruction

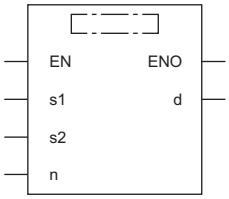
## Ramp signal

### RAMPF

This instruction obtains the data which changes between the start value (initial value) and the end value (target value) over the specified "n" times.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RAMPF(EN,s1,s2,n,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

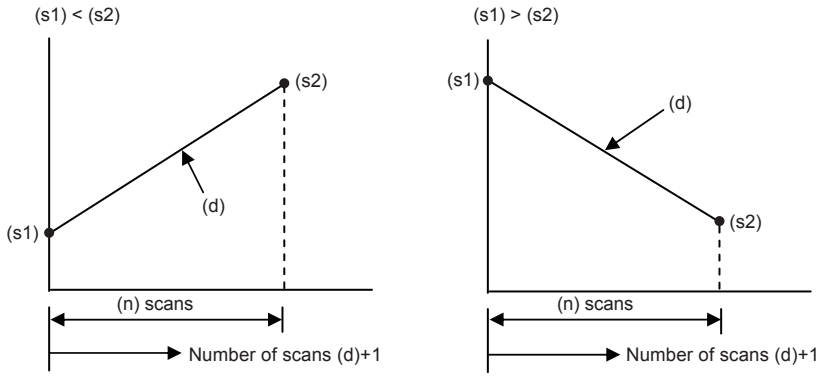
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Initial value of ramp | — | 16-bit signed binary | ANY16 |
| (s2) | Target value of ramp | — | 16-bit signed binary | ANY16 |
| (d) | (d)+0: Current value | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 2) |
| | (d)+1: Number of scans | | | |
| (n) | Ramp transfer time (scan) | 1 to 32767 | 16-bit unsigned binary | ANY16_U |

**■Applicable devices**

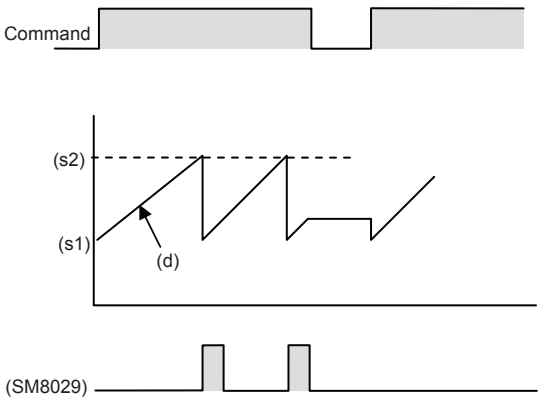
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- When the start value (s1) and the end value (s2) have been specified and the command input is set to ON, the value obtained by adding a value divided equally by "n" times to (s1) in the next operation cycle is stored to (d). By combining this instruction and an analog output, the cushion start/stop command can be output.




- The number of scans ("0" to "n") is stored in (d)+1.
- The time from start to the end value is the operation cycle multiplied by "n" times.
- If the command input is set to OFF in the middle of operation, execution is paused. (The current value stored in (d) is held, and the number of scans stored in (d)+1 is cleared.) When the command input is set to ON again, (d) is cleared, and the operation is started from (s1).
- After transfer is completed, the instruction execution complete flag SM8029 turns ON, and the (d) value is returned to the (s1) value.

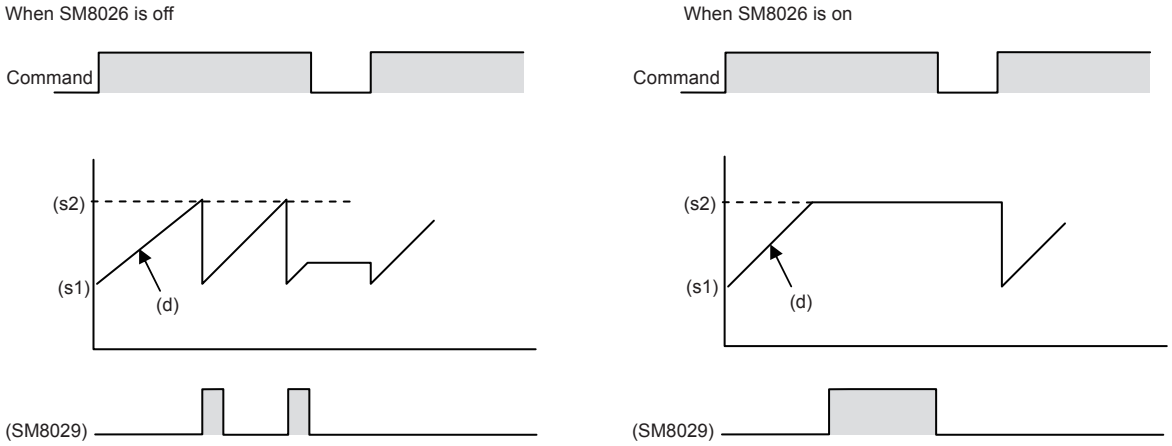



- When the operation result is acquired at a constant time interval (constant scan mode), write a prescribed scan time (which is longer than the actual scan time) to SD8039 and set SM8039 to ON. For example, when "20 ms" is written to SD8039 and "n" is set to 100, the (d) value will change from (s1) to (s2) in 2 seconds.
- The value used in the constant scan mode can be set in the parameter setting of an engineering tool (constant scan execution interval setting of CPU parameter).

  For details on the constant scan, refer to MELSEC iQ-F FX5 User's Manual (Application).

  For details on the engineering tool, refer to GX Works3 Operating Manual.

• The contents of (d) are changed as follows depending on the ON/OFF status of the mode flag SM8026.




## Precautions

To specify a latched (battery backed) type device as (d) when setting the CPU module to the RUN mode while the command input is ON, clear (d) in advance.

## Operation error

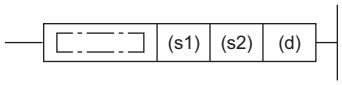
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n) is outside the following range.<br>1 to 32767 |

# 8.15 Pulse Related Instruction

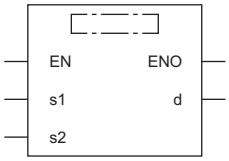
## Measuring the density of 16 bit binary pulses

### SPD

This instruction counts the number of times the device input specified by (s1) turns off → on only for the time (in 16-bit data units) specified by (s2) × 1ms and stores the operation result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SPD(EN,s1,s2,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Pulse input | — | Bit/Word | ANY_ELEMENTARY |
| (s2) | Measurement time (Unit: ms) | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d) | Head device number for storing the measurement result | — | 16-bit signed binary | ANY16 |

##### ■Applicable devices

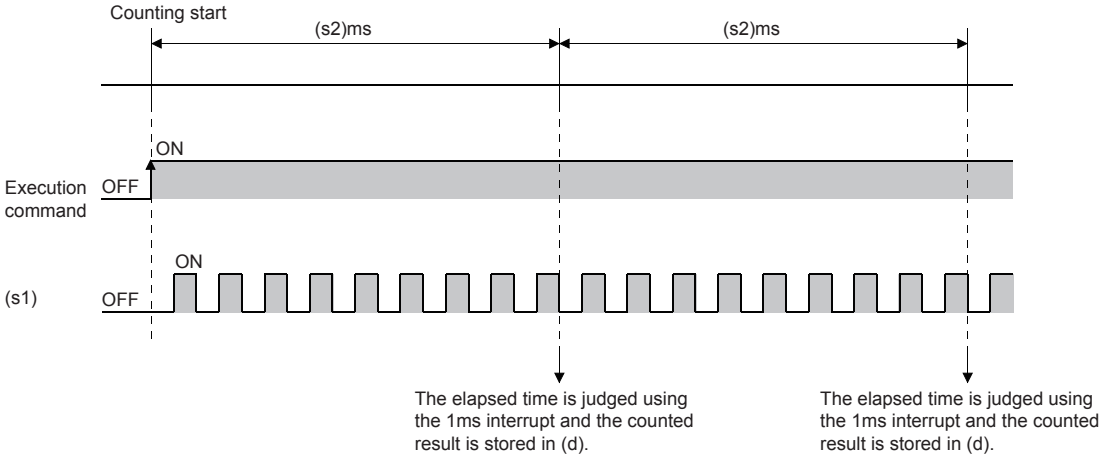
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○*1 | — | — | ○*2 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | ○ | — | — | ○ | — | — | — | — |

*1 When a bit device is specified, specify one of X0 to X17.
Only X can be used for a bit device.
The nibble of a bit device cannot be specified.

*2 When a word device is specified, specify one of the channel numbers (CH1 to CH8).
When FX3 compatible function of a high-speed counter is valid, a channel number cannot be specified.
If the channel numbers is specified in (s1), an error occurs. Only X can be used for a bit device.

## Processing details

- This instruction counts the number of times the device input specified by (s1) turns off → on only for the time (in 16-bit data units) specified by (s2) × 1ms and stores the operation result in the device specified by (d).




- The channel number of the high-speed counter specified by (s1) interlocks with the channel number in which parameters are set.
- When a word device is specified by (s1), this instruction counts the number of pulses by the high-speed counter setting of the channel number corresponding to each word device.
- When a bit device is specified by (s1), the following input assignment devices (shaded area) are valid.

General-purpose input assignment of the 1-phase 1-input counter (switching S/W up or down)

U/D: UP/DOWN pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | U/D(A) | | | | | | | | P | E | | | | | | |
| CH2 | | U/D(A) | | | | | | | | | P | E | | | | |
| CH3 | | | U/D(A) | | | | | | | | | | P | E | | |
| CH4 | | | | U/D(A) | | | | | | | | | | | P | E |
| CH5 | | | | | U/D(A) | | | | P | E | | | | | | |
| CH6 | | | | | | U/D(A) | | | | | P | E | | | | |
| CH7 | | | | | | | U/D(A) | | | | | | P | E | | |
| CH8 | | | | | | | | U/D(A) | | | | | | | P | E |

If one of X10 to X17 is specified as a device, an error occurs.

General-purpose input assignment of the 1-phase 1-input counter (switching H/W up or down)

C: Pulse input, D: Direction input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | C(A) | D(B) | | | | | | | P | E | | | | | | |
| CH2 | | | C(A) | D(B) | | | | | | | P | E | | | | |
| CH3 | | | | | C(A) | D(B) | | | | | | | P | E | | |
| CH4 | | | | | | | C(A) | D(B) | | | | | | | P | E |
| CH5 | | | | | | | | | C(A) | D(B) | P | E | | | | |
| CH6 | | | | | | | | | | | C(A) | D(B) | P | E | | |
| CH7 | | | | | | | | | | | | | C(A) | D(B) | P | E |
| CH8 | | | | | | | | | | | | | | | C(A) | D(B) |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

General-purpose input assignment of the 1-phase 2-input counter

U: UP pulse input, D: DOWN pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | U(A) | D(B) | | | | | | | P | E | | | | | | |
| CH2 | | | U(A) | D(B) | | | | | | | P | E | | | | |
| CH3 | | | | | U(A) | D(B) | | | | | | | P | E | | |
| CH4 | | | | | | | U(A) | D(B) | | | | | | | P | E |
| CH5 | | | | | | | | | U(A) | D(B) | P | E | | | | |
| CH6 | | | | | | | | | | | U(A) | D(B) | P | E | | |
| CH7 | | | | | | | | | | | | | U(A) | D(B) | P | E |
| CH8 | | | | | | | | | | | | | | | U(A) | D(B) |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

General-purpose input assignment of the 2-phase 2-input counter

A: A phase pulse input, B: B phase pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | A | B | | | | | | | P | E | | | | | | |
| CH2 | | | A | B | | | | | | | P | E | | | | |
| CH3 | | | | | A | B | | | | | | | P | E | | |
| CH4 | | | | | | | A | B | | | | | | | P | E |
| CH5 | | | | | | | | | A | B | P | E | | | | |
| CH6 | | | | | | | | | | | A | B | P | E | | |
| CH7 | | | | | | | | | | | | | A | B | P | E |
| CH8 | | | | | | | | | | | | | | | A | B |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

- The table below shows the related devices.

| Function | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | CH8 |
|---|---|---|---|---|---|---|---|---|
| Monitor in operation | SM4500 | SM4501 | SM4502 | SM4503 | SM4504 | SM4505 | SM4506 | SM4507 |
| High-speed counter pulse density | SD4507, SD4506 | SD4537, SD4536 | SD4567, SD4566 | SD4597, SD4596 | SD4627, SD4626 | SD4657, SD4656 | SD4687, SD4686 | SD4717, SD4716 |
| Measurement unit time | SD4517, SD4516 | SD4547, SD4546 | SD4577, SD4576 | SD4607, SD4606 | SD4637, SD4636 | SD4667, SD4666 | SD4697, SD4696 | SD4727, SD4726 |

- The table below shows the related device update timing.

| Function | R/W | Update timing | Clear |
|---|---|---|---|
| Monitor in operation | R | • When the SPD instruction is executed<br>• When the HIOEN instruction is executed | • Power-on<br>• Reset |
| High-speed counter pulse density | R | • Each time the measurement unit time elapses | • Power-on<br>• Reset |
| Measurement unit time | R/W | • When the SPD instruction is executed | • Power-on<br>• Reset |

## Precautions

- The maximum input frequency of turning the inputs ON and OFF is shown below:
  FX5U-32M□/ FX5UC-32M□ CPU module

| Used input number | Maximum input frequency |
|---|---|
| X0 to X5 | 200 kHz |
| X6, X7 | 10 kHz |

FX5U-64M□/FX5U-80M□ CPU module

| Used input number | Maximum input frequency |
|---|---|
| X0 to X7 | 200 kHz |
| X10 to X17 | 10 kHz |

- When the SPD instruction is used, the UP/DOWN pulse input, preset input and enable input operate in accordance with the contents set by the parameters of the high-speed counter.
- When the measurement time is changed while the SPD instruction is executed, the changed time is applied every time the measurement time ends.
- When the SPD instruction is started, the high-speed counter and pulse density measurement are started simultaneously. When the SPD instruction is stopped, only the pulse density measurement is stopped and the high-speed counter is not stopped.
- When the current value of the high-speed counter is overwritten, a preset input is executed, or the high-speed counter is reset by the DHCMOV instruction while the SPD instruction is executed, the operation continues, but the pulse density cannot be measured normally.
- When the SPD instruction is used, pulses per unit time which exceeds the ring length of the high-speed counter cannot be input.
- The measurement time specified by (s2) overwrites the value stored in the SD device specified for the measurement unit time.
- When the measurement time specified by (s2) is outside the range from 1 to 32767, the specified measurement time is rounded into “1” with the sign.

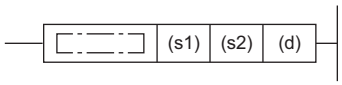
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3600H | The channel number or device number in which parameters are not set in (s1) is specified. |
| 3405H | An unavailable bit device is set in (s1). |
| | A channel number other than 1 to 8 is specified in (s1). |
| | When FX3 compatible function of a high-speed counter is valid and a channel number is specified to (s1). |
| 1810H | The input specified in (s1) is already used by another instruction. |

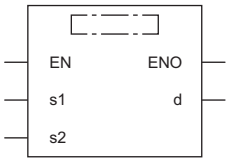
# Measuring the density of 32 bit binary pulses

## DSPD

This instruction counts the number of times the device input specified by (s1) turns off → on only for the time (in 32-bit data units) specified by (s2) × 1ms and stores the operation result in the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSPD(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Pulse input | — | Bit/Word/Double word | ANY_ELEMENTARY |
| (s2) | Measurement time (Unit: ms) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d) | Head device number for storing the measurement result | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

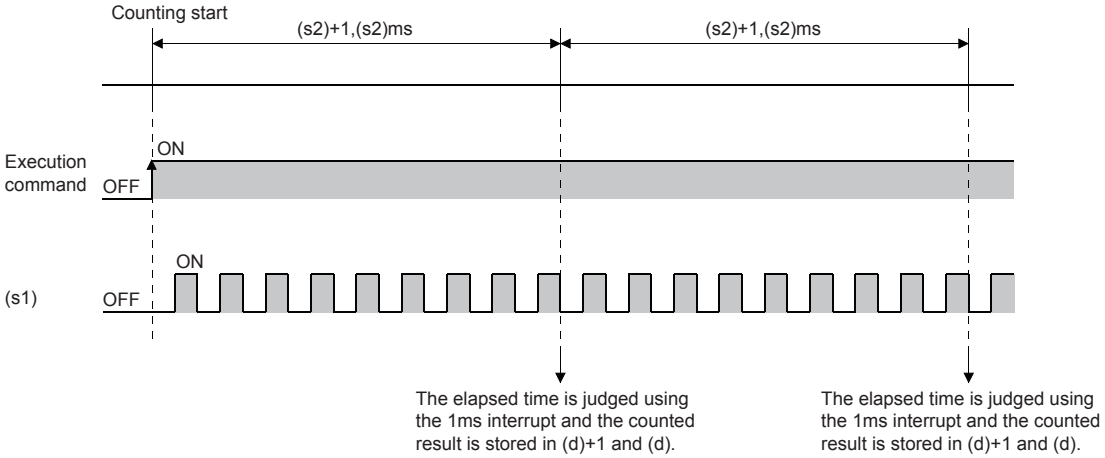
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○*1 | — | — | ○*2 | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | ○ | ○ | ○ | ○ | — | — | — | — |

*1 When a bit device is specified, specify one of X0 to X17.
Only X can be used for a bit device.
The nibble of a bit device cannot be specified.

*2 When a word device is specified, specify one of the channel numbers (CH1 to CH8).
When FX3 compatible function of a high-speed counter is valid, a channel number cannot be specified.
If the channel numbers is specified in (s1), an error occurs. Only X can be used for a bit device.

## Processing details

- This instruction counts the number of times the device input specified by (s1) turns off → on only for the time (in 32-bit data units) specified by (s2) × 1ms and stores the operation result in the device specified by (d).




- The channel number of the high-speed counter specified by (s1) interlocks with the channel number in which parameters are set.
- When a word device is specified by (s1), this instruction counts the number of pulses by the high-speed counter setting of the channel number corresponding to each word device.
- When a bit device is specified by (s1), the following input assignment devices (shaded area) are valid.

General-purpose input assignment of the 1-phase 1-input counter (switching S/W up or down)

U/D: UP/DOWN pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | U/D(A) | | | | | | | | P | E | | | | | | |
| CH2 | | U/D(A) | | | | | | | | | P | E | | | | |
| CH3 | | | U/D(A) | | | | | | | | | | P | E | | |
| CH4 | | | | U/D(A) | | | | | | | | | | | P | E |
| CH5 | | | | | U/D(A) | | | | P | E | | | | | | |
| CH6 | | | | | | U/D(A) | | | | | P | E | | | | |
| CH7 | | | | | | | U/D(A) | | | | | | P | E | | |
| CH8 | | | | | | | | U/D(A) | | | | | | | P | E |

If one of X10 to X17 is specified as a device, an error occurs.

General-purpose input assignment of the 1-phase 1-input counter (switching H/W up or down)

C: Pulse input, D: Direction input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | C(A) | D(B) | | | | | | | P | E | | | | | | |
| CH2 | | | C(A) | D(B) | | | | | | | P | E | | | | |
| CH3 | | | | | C(A) | D(B) | | | | | | | P | E | | |
| CH4 | | | | | | | C(A) | D(B) | | | | | | | P | E |
| CH5 | | | | | | | | | C(A) | D(B) | P | E | | | | |
| CH6 | | | | | | | | | | | C(A) | D(B) | P | E | | |
| CH7 | | | | | | | | | | | | | C(A) | D(B) | P | E |
| CH8 | | | | | | | | | | | | | | | C(A) | D(B) |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

General-purpose input assignment of the 1-phase 2-input counter

U: UP pulse input, D: DOWN pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | U(A) | D(B) | | | | | | | P | E | | | | | | |
| CH2 | | | U(A) | D(B) | | | | | | | P | E | | | | |
| CH3 | | | | | U(A) | D(B) | | | | | | | P | E | | |
| CH4 | | | | | | | U(A) | D(B) | | | | | | | P | E |
| CH5 | | | | | | | | | U(A) | D(B) | P | E | | | | |
| CH6 | | | | | | | | | | | U(A) | D(B) | P | E | | |
| CH7 | | | | | | | | | | | | | U(A) | D(B) | P | E |
| CH8 | | | | | | | | | | | | | | | U(A) | D(B) |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

General-purpose input assignment of the 2-phase 2-input counter

A: A phase pulse input, B: B phase pulse input, P: Preset input (reset), E: Enable input (start)

| | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1 | A | B | | | | | | | P | E | | | | | | |
| CH2 | | | A | B | | | | | | | P | E | | | | |
| CH3 | | | | | A | B | | | | | | | P | E | | |
| CH4 | | | | | | | A | B | | | | | | | P | E |
| CH5 | | | | | | | | | A | B | P | E | | | | |
| CH6 | | | | | | | | | | | A | B | P | E | | |
| CH7 | | | | | | | | | | | | | A | B | P | E |
| CH8 | | | | | | | | | | | | | | | A | B |

If one of X1, X3, X5, X7, X11, X13, X15, X17 is specified as a device, an error occurs.

- The table below shows the related devices.

| Function | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | CH8 |
|---|---|---|---|---|---|---|---|---|
| Monitor in operation | SM4500 | SM4501 | SM4502 | SM4503 | SM4504 | SM4505 | SM4506 | SM4507 |
| High-speed counter pulse density | SD4507, SD4506 | SD4537, SD4536 | SD4567, SD4566 | SD4597, SD4596 | SD4627, SD4626 | SD4657, SD4656 | SD4687, SD4686 | SD4717, SD4716 |
| Measurement unit time | SD4517, SD4516 | SD4547, SD4546 | SD4577, SD4576 | SD4607, SD4606 | SD4637, SD4636 | SD4667, SD4666 | SD4697, SD4696 | SD4727, SD4726 |

- The table below shows the related device update timing.

| Function | R/W | Update timing | Clear |
|---|---|---|---|
| Monitor in operation | R | • When the DSPD instruction is executed<br>• When the DHIOEN instruction is executed | • Power-on<br>• Reset |
| High-speed counter pulse density | R | • Each time the measurement unit time elapses | • Power-on<br>• Reset |
| Measurement unit time | R/W | • When the DSPD instruction is executed | • Power-on<br>• Reset |

## Precautions

- The maximum input frequency of turning the inputs ON and OFF is shown below:
  FX5U-32M□/FX5UC-32M□ CPU module

| Used input number | Maximum input frequency |
|---|---|
| X0 to X5 | 200 kHz |
| X6, X7 | 10 kHz |

FX5U-64M□/FX5U-80M□ CPU module

| Used input number | Maximum input frequency |
|---|---|
| X0 to X7 | 200 kHz |
| X10 to X17 | 10 kHz |

- When the DSPD instruction is used, the UP/DOWN pulse input, preset input and enable input operate in accordance with the contents set by the parameters of the high-speed counter.
- When the measurement time is changed while the DSPD instruction is executed, the changed time is applied every time the measurement time ends.
- When the DSPD instruction is started, the high-speed counter and pulse density measurement are started simultaneously. When the DSPD instruction is stopped, only the pulse density measurement is stopped and the high-speed counter is not stopped.
- When the current value of the high-speed counter is overwritten, a preset input is executed, or the high-speed counter is reset by the DHCMOV instruction while the SPD instruction is executed, the operation continues, but the pulse density cannot be measured normally.
- When the DSPD instruction is used, pulses per unit time which exceeds the ring length of the high-speed counter cannot be input.
- The measurement time specified by (s2) overwrites the value stored in the SD device specified for the measurement unit time.
- When the measurement time specified by (s2) is outside the range from 1 to 2147483647, the specified measurement time is rounded into “1” with the sign.

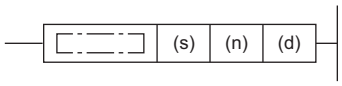
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3600H | The channel number or device number in which parameters are not set in (s1) is specified. |
| 3405H | An unavailable bit device is set in (s1). |
| | A channel number other than 1 to 8 is specified in (s1). |
| | When FX3 compatible function of a high-speed counter is valid and a channel number is specified to (s1). |
| 1810H | The input specified in (s1) is already used by another instruction. |

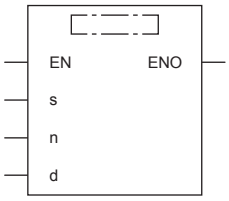
# 16 bit binary pulse output

## PLSY [For the FX3 compatible operand specification]

This instruction outputs 16-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 16-bit pulses specified by the positioning address (n).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLSY(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed or word device number storing data | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (n) | Positioning address or word device number storing data | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (d) | Bit device number from which pulses are to be output | 0 to 3 | bit | ANY_ELEMENTARY (BOOL) |

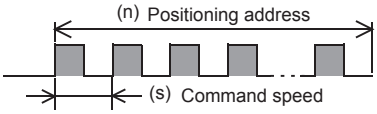
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Y0 to Y3 can be used.

### Processing details

- This instruction outputs 16-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 16-bit pulses specified by the positioning address (n).




- Set the value from 0 to 65535 (in user unit) to the command speed (s), so that the command speed is 200 kpps or less when the command speed is converted to frequency.
- Set the value from 0 to 65535 (in user unit) to the positioning address (n), so that the positioning address is within the range from 0 to 2147483647 when the positioning address is converted to number of pulses.
- Specify the Y device number (Y0 to Y3) in (d).

• The following tables show the special relays and special registers related to the PLSY instruction.

[Special relays]

| Axis number | | | | Name | Descriptions |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | |
| SM5500 | SM5501 | SM5502 | SM5503 | Positioning instruction activation | ON: During activation, OFF: Not activated |
| SM5516 | SM5517 | SM5518 | SM5519 | Pulse output monitor | ON: During output, OFF: During stop |
| SM5532 | SM5533 | SM5534 | SM5535 | Positioning error occurrence | On: Error occurred, OFF: Error not occurred |
| SM5628 | SM5629 | SM5630 | SM5631 | Pulse output stop command | ON: Stop command is on, OFF: Stop command is off |
| SM5644 | SM5645 | SM5646 | SM5647 | Pulse deceleration stop command*1 | ON: Deceleration stop command is on, OFF: Deceleration stop command is off |
| SM5660 | SM5661 | SM5662 | SM5663 | Forward limit | ON: Forward limit is on, OFF: Forward limit is off |
| SM5676 | SM5677 | SM5678 | SM5679 | Reverse limit | ON: Reverse limit is on, OFF: Reverse limit is off |

*1 Because the PLSY instruction does not have the acceleration/deceleration function, the operation is stopped immediately even though the pulse deceleration stop command is turned on.

[Special registers]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD5500<br>SD5501 | SD5540<br>SD5541 | SD5580<br>SD5581 | SD5620<br>SD5621 | Current address (in user unit) |
| SD5502<br>SD5503 | SD5542<br>SD5543 | SD5582<br>SD5583 | SD5622<br>SD5623 | Current address (in pulse unit) |
| SD5504<br>SD5505 | SD5544<br>SD5545 | SD5584<br>SD5585 | SD5624<br>SD5625 | Current speed (in user unit) |
| SD5510 | SD5550 | SD5590 | SD5630 | Positioning error error code |

[Special relays (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SM8029 | | | | Instruction execution complete flag |
| SM8329 | | | | Instruction execution abnormal end flag |
| SM8340 | SM8350 | SM8360 | SM8370 | Pulse output monitoring |
| SM8348 | SM8358 | SM8368 | SM8378 | Positioning instruction activation |

[Special registers (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD8136<br>SD8137 | | — | — | Total number of outputs for axis 1 and 2 of PLSY instruction |
| SD8140<br>SD8141 | SD8142<br>SD8143 | — | — | Total number of output pulses of PLSY instruction |
| SD8340<br>SD8341 | SD8350<br>SD8351 | SD8360<br>SD8361 | SD8370<br>SD8371 | Current address (in user unit) |

## Precautions

- The operation cannot be performed normally in an environment such as user program where the instruction cannot be executed at each scan or if the instruction is jumped by the CJ(P) instruction. However, the pulse output is continued.
- The same devices as the ones of position instruction, PWM output or general-purpose output cannot be used for the output in the PLSY instruction.
- The following table shows how to stop the pulse output. The operation is stopped immediately in any stopping method by the PLSY instruction. Note that the motor is stopped without deceleration and this may damage the system.

| Operation | Whether to decelerate or not | Abnormal end flag |
|---|---|---|
| Turn off the drive contact. | Stops immediately. | OFF |
| All outputs disable (Turn on the special relay.) | | ON |
| Pulse output stop command (Turn on the special relay.) | | ON |
| Pulse deceleration stop command (Turn on the special relay.) | | ON |
| Forward limit (Turn on the special relay.) | | ON |
| Reverse limit (Turn on the special relay.) | | ON |
| Set 0 for the command speed specified by (s2). | | OFF |

- If the positioning address is 0 when the PLSY instruction is activated, pulses are output without limitation.
- Overwrite the positioning address during the pulse output to change the positioning address in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten. The positioning address becomes invalid if it is changed from 0 to a value other than 0 or from a value other than 0 to 0 during positioning operation.
- When the positioning address is changed during the pulse output, the operation is stopped immediately if the changed value is the number of pulses which have already been output or less.
- Overwrite the command speed during the pulse output to change the command speed in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten.
- When the numbers of pulses (by the pulses conversion) of the command speed and positioning address exceed the 32-bit range, an error occurs and the operation cannot be performed.
- The PLSY instruction always increases the current address because the setting of rotation direction is disabled due to the absence of direction.
- When the output mode is CW/CCW mode, output is always performed from the device set to CW.
- If reverse limit is used, it operates as forward limit.
- Do not set the value of 200 kpps or more by the frequency conversion when changing the command speed during the pulse output.
- If the command speed is set to 0 when the PLSY instruction is activated, the operation ends with an error and stops pulse output.
- If the command speed is changed to 0 during pulse output, the operation immediately stops without abnormal end flag. However, if the drive contact is not turned off, pulse output will restart when the command speed is changed.
- The command speed is changed to negative value during pulse output, it is the operation ends with an error.
- The following table shows the operation timing of the complete flag and abnormal end flag of the PLSY instruction.

| | Complete flag (SM8029) | Abnormal end flag (SM8329) |
|---|---|---|
| ON condition | From when the output of the specified positioning address is completed until the drive contact is turned off<br>• Pulse deceleration stop command (when unlimited pulses are being output) | From the following stops until the drive contact is turned off<br>• The specified axis is already used*1<br>• Pulse output stop command<br>• Pulse deceleration stop command (when unlimited pulses are not being output)<br>• Forward limit<br>• Reverse limit<br>• All outputs disabled<br>• Positioning address error<br>• Command speed 0 (when the PLSY instruction is activated) |
| ON→OFF condition | • When the drive contact is turned off | • When the drive contact is turned off |

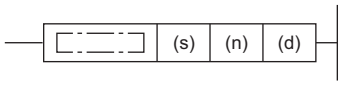
*1 The flag turns on only during one scan time when the activation contact of the instruction turns off and on.

## Operation error

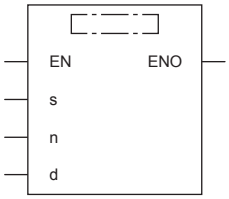
| Error code (SD0/SD8067) | | | | Description |
|---|---|---|---|---|
| Axis 1 | Axis 2 | Axis 3 | Axis 4 | |
| SD5510 | SD5550 | SD5590 | SD5630 | |
| 1810H | | | | The axis number specified by (d) is used by another instruction. |
| 2820H | | | | The value specified by (s) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (n) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (d) is outside the following range.<br>0 to 3 |
| 3600H | | | | The axis number specified by (d) is not set by parameters.<br>A function which is set to be not used by parameters (such as interrupt input signal 1 and zero return relations) is used. |
| 3631H | 3632H | 3633H | 3634H | The numbers of pulses (by the pulses conversion) of the positioning address specified by (n) exceed the 32-bit range. |
| 3641H | 3642H | 3643H | 3644H | The numbers of pulses (by the pulses conversion) of the command speed specified by (s) exceed the 32-bit range. |
| 3651H | 3652H | 3653H | 3654H | The operation decelerates and stops by the forward limit or reverse limit during the pulse output or at the activating of the positioning. |
| 3661H | 3662H | 3663H | 3664H | The operation decelerates and stops by the pulse output stop command or special relay whose all outputs are disabled during the pulse output or at the activating of the positioning. |

## PLSY [For the FX5 operand specification]

This instruction outputs 16-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 16-bit pulses specified by the positioning address (n).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLSY(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

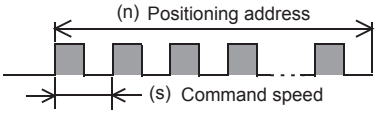
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed or word device number storing data | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (n) | Positioning address or word device number storing data | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (d) | Axis number from which pulses are to be output | 1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction outputs 16-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 16-bit pulses specified by the positioning address (n).




- Set the value from 0 to 65535 (in user unit) in the command speed (s), so that the command speed is 200 kpps or less when the command speed is converted to frequency.
- Set the value from 0 to 65535 (in user unit) in the positioning address (n), so that the positioning address is within the range from 0 to 2147483647 when the positioning address is converted to number of pulses.
- Specify the axis number (K1 to K4) in which positioning parameters exist in (d).

• The following tables show the special relays and special registers related to the PLSY instruction.

[Special relays]

| Axis number | | | | Name | Descriptions |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | |
| SM5500 | SM5501 | SM5502 | SM5503 | Positioning instruction activation | ON: During activation, OFF: Not activated |
| SM5516 | SM5517 | SM5518 | SM5519 | Pulse output monitor | ON: During output, OFF: During stop |
| SM5532 | SM5533 | SM5534 | SM5535 | Positioning error occurrence | On: Error occurred, OFF: Error not occurred |
| SM5628 | SM5629 | SM5630 | SM5631 | Pulse output stop command | ON: Stop command is on, OFF: Stop command is off |
| SM5644 | SM5645 | SM5646 | SM5647 | Pulse deceleration stop command*1 | ON: Deceleration stop command is on, OFF: Deceleration stop command is off |
| SM5660 | SM5661 | SM5662 | SM5663 | Forward limit | ON: Forward limit is on, OFF: Forward limit is off |
| SM5676 | SM5677 | SM5678 | SM5679 | Reverse limit | ON: Reverse limit is on, OFF: Reverse limit is off |

*1 Because the PLSY instruction does not have the acceleration/deceleration function, the operation is stopped immediately even though the pulse deceleration stop command is turned on.

[Special registers]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD5500<br>SD5501 | SD5540<br>SD5541 | SD5580<br>SD5581 | SD5620<br>SD5621 | Current address (in user unit) |
| SD5502<br>SD5503 | SD5542<br>SD5543 | SD5582<br>SD5583 | SD5622<br>SD5623 | Current address (in pulse unit) |
| SD5504<br>SD5505 | SD5544<br>SD5545 | SD5584<br>SD5585 | SD5624<br>SD5625 | Current speed (in user unit) |
| SD5510 | SD5550 | SD5590 | SD5630 | Positioning error error code |

[Special relays (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SM8029 | | | | Instruction execution complete flag |
| SM8329 | | | | Instruction execution abnormal end flag |
| SM8340 | SM8350 | SM8360 | SM8370 | Pulse output monitoring |
| SM8348 | SM8358 | SM8368 | SM8378 | Positioning instruction activation |

[Special registers (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD8136<br>SD8137 | | — | — | Total number of outputs for axis 1 and 2 of PLSY instruction |
| SD8140<br>SD8141 | SD8142<br>SD8143 | — | — | Total number of output pulses of PLSY instruction |
| SD8340<br>SD8341 | SD8350<br>SD8351 | SD8360<br>SD8361 | SD8370<br>SD8371 | Current address (in user unit) |

## Precautions

- The operation cannot be performed normally in an environment such as user program where the instruction cannot be executed at each scan or if the instruction is jumped by the CJ(P) instruction. However, the pulse output is continued.
- The same devices as the ones of position instruction, PWM output or general-purpose output cannot be used for the output in the PLSY instruction.
- The following table shows how to stop the pulse output. The operation is stopped immediately in any stopping method by the PLSY instruction. Note that the motor is stopped without deceleration and this may damage the system.

| Operation | Whether to decelerate or not | Abnormal end flag |
|---|---|---|
| Turn off the drive contact. | Stops immediately. | OFF |
| All outputs disable (Turn on the special relay.) | | ON |
| Pulse output stop command (Turn on the special relay.) | | ON |
| Pulse deceleration stop command (Turn on the special relay.) | | ON |
| Forward limit (Turn on the special relay.) | | ON |
| Reverse limit (Turn on the special relay.) | | ON |
| Set 0 for the command speed specified by (s2). | | OFF |

- If the positioning address is 0 when the PLSY instruction is activated, pulses are output without limitation.
- Overwrite the positioning address during the pulse output to change the positioning address in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten. The positioning address becomes invalid if it is changed from 0 to a value other than 0 or from a value other than 0 to 0 during positioning operation.
- When the positioning address is changed during the pulse output, the operation is stopped immediately if the changed value is the number of pulses which have already been output or less.
- Overwrite the command speed during the pulse output to change the command speed in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten.
- When the numbers of pulses (by the pulses conversion) of the command speed and positioning address exceed the 32-bit range, an error occurs and the operation cannot be performed.
- The PLSY instruction always increases the current address because the setting of rotation direction is disabled due to the absence of direction.
- When the output mode is CW/CCW mode, output is always performed from the device set to CW.
- If reverse limit is used, it operates as forward limit.
- Do not set the value of 200 kpps or more by the frequency conversion when changing the command speed during the pulse output.
- If the command speed is set to 0 when the PLSY instruction is activated, the operation ends with an error and stops pulse output.
- If the command speed is changed to 0 during pulse output, the operation immediately stops without abnormal end flag. However, if the drive contact is not turned off, pulse output will restart when the command speed is changed.
- The command speed is changed to negative value during pulse output, it is the operation ends with an error.
- The following table shows the operation timing of the complete flag and abnormal end flag of the PLSY instruction.

| | Complete flag (SM8029) | Abnormal end flag (SM8329) |
|---|---|---|
| ON condition | From when the output of the specified positioning address is completed until the drive contact is turned off<br>• Pulse decelerate and stop command (when unlimited pulses are being output) | From the following stops until the drive contact is turned off<br>• The specified axis is already used*1<br>• Pulse output stop command<br>• Pulse deceleration stop command (when unlimited pulses are not being output)<br>• Forward limit<br>• Reverse limit<br>• All outputs disabled<br>• Positioning address error<br>• Command speed 0 (when the PLSY instruction is activated) |
| ON→OFF condition | • When the drive contact is turned off | • When the drive contact is turned off |

*1 The flag turns on only during one scan time when the activation contact of the instruction turns off and on.

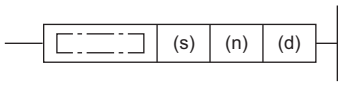
## Operation error

| Error code (SD0/SD8067) | | | | Description |
|---|---|---|---|---|
| Axis 1 | Axis 2 | Axis 3 | Axis 4 | |
| SD5510 | SD5550 | SD5590 | SD5630 | |
| 1810H | | | | The axis number specified by (d) is used by another instruction. |
| 2820H | | | | The value specified by (s) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (n) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (d) is outside the following range.<br>0 to 3 |
| 3600H | | | | The axis number specified by (d) is not set by parameters.<br>A function which is set to be not used by parameters (such as interrupt input signal 1 and zero return relations) is used. |
| 3631H | 3632H | 3633H | 3634H | The numbers of pulses (by the pulses conversion) of the positioning address specified by (n) exceed the 32-bit range. |
| 3641H | 3642H | 3643H | 3644H | The numbers of pulses (by the pulses conversion) of the command speed specified by (s) exceed the 32-bit range. |
| 3651H | 3652H | 3653H | 3654H | The operation decelerates and stops by the forward limit or reverse limit during the pulse output or at the activating of the positioning. |
| 3661H | 3662H | 3663H | 3664H | The operation decelerates and stops by the pulse output stop command or special relay whose all outputs are disabled during the pulse output or at the activating of the positioning. |

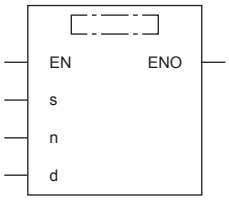
# 32 bit binary pulse output

## DPLSY [For the FX3 compatible operand specification]

This instruction outputs 32-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 32-bit pulses specified by the positioning address (n).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPLSY(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed or word device number storing data | 0 to 2147483647 | 32-bit unsigned binary | ANY32 |
| (n) | Positioning address or word device number storing data | 0 to 2147483647 | 32-bit unsigned binary | ANY32 |
| (d) | Bit device number from which pulses are to be output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |

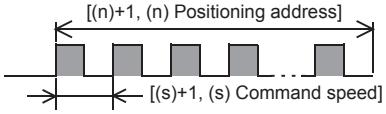
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Y0 to Y3 can be used.

### Processing details

- This instruction outputs 32-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 32-bit pulses specified by the positioning address (n).




- Set the value from 0 to 2147483647 (in user unit) to the command speed (s), so that the command speed is 200 kpps or less when the command speed is converted to frequency.
- Set the value from 0 to 2147483647 (in user unit) to the positioning address (n), so that the positioning address is within the range from 0 to 2147483647 when the positioning address is converted to number of pulses.
- Specify the Y device number (Y0 to Y3) in (d).

• The following tables show the special relays and special registers related to the DPLSY instruction.

[Special relays]

| Axis number | | | | Name | Descriptions |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | |
| SM5500 | SM5501 | SM5502 | SM5503 | Positioning instruction activation | ON: During activation, OFF: Not activated |
| SM5516 | SM5517 | SM5518 | SM5519 | Pulse output monitor | ON: During output, OFF: During stop |
| SM5532 | SM5533 | SM5534 | SM5535 | Positioning error occurrence | On: Error occurred, OFF: Error not occurred |
| SM5628 | SM5629 | SM5630 | SM5631 | Pulse output stop command | ON: Stop command is on, OFF: Stop command is off |
| SM5644 | SM5645 | SM5646 | SM5647 | Pulse deceleration stop command*1 | ON: Deceleration stop command is on, OFF: Deceleration stop command is off |
| SM5660 | SM5661 | SM5662 | SM5663 | Forward limit | ON: Forward limit is on, OFF: Forward limit is off |
| SM5676 | SM5677 | SM5678 | SM5679 | Reverse limit | ON: Reverse limit is on, OFF: Reverse limit is off |

*1 Because the DPLSY instruction does not have the acceleration/deceleration function, the operation is stopped immediately even though the pulse deceleration stop command is turned on.

[Special registers]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD5500<br>SD5501 | SD5540<br>SD5541 | SD5580<br>SD5581 | SD5620<br>SD5621 | Current address (in user unit) |
| SD5502<br>SD5503 | SD5542<br>SD5543 | SD5582<br>SD5583 | SD5622<br>SD5623 | Current address (in pulse unit) |
| SD5504<br>SD5505 | SD5544<br>SD5545 | SD5584<br>SD5585 | SD5624<br>SD5625 | Current speed (in user unit) |
| SD5510 | SD5550 | SD5590 | SD5630 | Positioning error error code |

[Special relays (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SM8029 | | | | Instruction execution complete flag |
| SM8329 | | | | Instruction execution abnormal end flag |
| SM8340 | SM8350 | SM8360 | SM8370 | Pulse output monitoring |
| SM8348 | SM8358 | SM8368 | SM8378 | Positioning instruction activation |

[Special registers (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD8136<br>SD8137 | | — | — | Total number of outputs for axis 1 and 2 of PLSY instruction |
| SD8140<br>SD8141 | SD8142<br>SD8143 | — | — | Total number of output pulses of PLSY instruction |
| SD8340<br>SD8341 | SD8350<br>SD8351 | SD8360<br>SD8361 | SD8370<br>SD8371 | Current address (in user unit) |

## Precautions

- The operation cannot be performed normally in an environment such as user program where the instruction cannot be executed at each scan or if the instruction is jumped by the CJ(P) instruction. However, the pulse output is continued.
- The same devices as the ones of position instruction, PWM output or general-purpose output cannot be used for the output in the DPLSY instruction.
- The following table shows how to stop the pulse output. The operation is stopped immediately in any stopping method by the DPLSY instruction. Note that the motor is stopped without deceleration and this may damage the system.

| Operation | Whether to decelerate or not | Abnormal end flag |
|---|---|---|
| Turn off the drive contact. | Stops immediately. | OFF |
| All outputs disable (Turn on the special relay.) | | ON |
| Pulse output stop command (Turn on the special relay.) | | ON |
| Pulse deceleration stop command (Turn on the special relay.) | | ON |
| Forward limit (Turn on the special relay.) | | ON |
| Reverse limit (Turn on the special relay.) | | ON |
| Set 0 for the command speed specified by (s2). | | OFF |

- If the positioning address is 0 when the DPLSY instruction is activated, pulses are output without limitation.
- Overwrite the positioning address during the pulse output to change the positioning address in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten. The positioning address becomes invalid if it is changed from 0 to a value other than 0 or from a value other than 0 to 0 during positioning operation.
- When the positioning address is changed during the pulse output, the operation is stopped immediately if the changed value is the number of pulses which have already been output or less.
- Overwrite the command speed during the pulse output to change the command speed in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten.
- When the numbers of pulses (by the pulses conversion) of the command speed and positioning address exceed the 32-bit range, an error occurs and the operation cannot be performed.
- The DPLSY instruction always increases the current address because the setting of rotation direction is disabled due to the absence of direction.
- When the output mode is CW/CCW mode, output is always performed from the device set to CW.
- If reverse limit is used, it operates as forward limit.
- Do not set the value of 200 kpps or more by the frequency conversion when changing the command speed during the pulse output.
- If the command speed is set to 0 when the DPLSY instruction is activated, the operation ends with an error and stops pulse output.
- If the command speed is changed to 0 during pulse output, the operation immediately stops without abnormal end flag. However, if the drive contact is not turned off, pulse output will restart when the command speed is changed.
- The command speed is changed to negative value during pulse output, it is the operation ends with an error.
- The following table shows the operation timing of the complete flag and abnormal end flag of the DPLSY instruction.

| | Complete flag (SM8029) | Abnormal end flag (SM8329) |
|---|---|---|
| ON condition | From when the output of the specified positioning address is completed until the drive contact is turned off<br>• Pulse decelerate and stop command (when unlimited pulses are being output) | From the following stops until the drive contact is turned off<br>• The specified axis is already used*1<br>• Pulse output stop command<br>• Pulse deceleration stop command (when unlimited pulses are not being output)<br>• Forward limit<br>• Reverse limit<br>• All outputs disabled<br>• Positioning address error<br>• Command speed 0 (when the DPLSY instruction is activated) |
| ON→OFF condition | • When the drive contact is turned off | • When the drive contact is turned off |

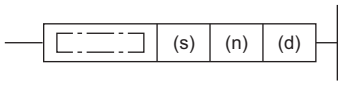
*1 The flag turns on only during one scan time when the activation contact of the instruction turns off and on.

## Operation error

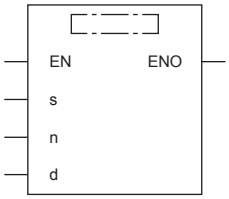
| Error code (SD0/SD8067) | | | | Description |
|---|---|---|---|---|
| Axis 1 | Axis 2 | Axis 3 | Axis 4 | |
| SD5510 | SD5550 | SD5590 | SD5630 | |
| 1810H | | | | The axis number specified by (d) is used by another instruction. |
| 2820H | | | | The value specified by (s) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (n) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (d) is outside the following range.<br>0 to 3 |
| 3600H | | | | The axis number specified by (d) is not set by parameters.<br>A function which is set to be not used by parameters (such as interrupt input signal 1 and zero return relations) is used. |
| 3631H | 3632H | 3633H | 3634H | The numbers of pulses (by the pulses conversion) of the positioning address specified by (n) exceed the 32-bit range. |
| 3641H | 3642H | 3643H | 3644H | The numbers of pulses (by the pulses conversion) of the command speed specified by (s) exceed the 32-bit range. |
| 3651H | 3652H | 3653H | 3654H | The operation decelerates and stops by the forward limit or reverse limit during the pulse output or at the activating of the positioning. |
| 3661H | 3662H | 3663H | 3664H | The operation decelerates and stops by the pulse output stop command or special relay whose all outputs are disabled during the pulse output or at the activating of the positioning. |

## DPLSY [For the FX5 operand specification]

This instruction outputs 32-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 32-bit pulses specified by the positioning address (n).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPLSY(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

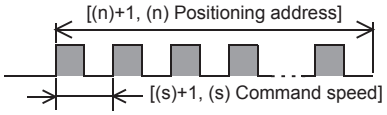
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed or word device number storing data | 0 to 2147483647 | 32-bit signed binary | ANY32 |
| (n) | Positioning address or word device number storing data | 0 to 2147483647 | 32-bit signed binary | ANY32 |
| (d) | Axis number from which pulses are to be output | 1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- This instruction outputs 32-bit pulse trains specified by the command speed (s) from the device specified by the output (d) for the amount of 32-bit pulses specified by the positioning address (n).




- Set the value from 0 to 2147483647 (in user unit) to the command speed (s), so that the command speed is 200 kpps or less when the command speed is converted to frequency.
- Set the value from 0 to 2147483647 (in user unit) to the positioning address (n), so that the positioning address is within the range from 0 to 2147483647 when the positioning address is converted to number of pulses.
- Specify the axis number (K1 to K4) in which positioning parameters exist in (d).

• The following tables show the special relays and special registers related to the DPLSY instruction.

[Special relays]

| Axis number | | | | Name | Descriptions |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | |
| SM5500 | SM5501 | SM5502 | SM5503 | Positioning instruction activation | ON: During activation, OFF: Not activated |
| SM5516 | SM5517 | SM5518 | SM5519 | Pulse output monitor | ON: During output, OFF: During stop |
| SM5532 | SM5533 | SM5534 | SM5535 | Positioning error occurrence | On: Error occurred, OFF: Error not occurred |
| SM5628 | SM5629 | SM5630 | SM5631 | Pulse output stop command | ON: Stop command is on, OFF: Stop command is off |
| SM5644 | SM5645 | SM5646 | SM5647 | Pulse deceleration stop command*1 | ON: Deceleration stop command is on, OFF: Deceleration stop command is off |
| SM5660 | SM5661 | SM5662 | SM5663 | Forward limit | ON: Forward limit is on, OFF: Forward limit is off |
| SM5676 | SM5677 | SM5678 | SM5679 | Reverse limit | ON: Reverse limit is on, OFF: Reverse limit is off |

*1 Because the DPLSY instruction does not have the acceleration/deceleration function, the operation is stopped immediately even though the pulse deceleration stop command is turned on.

[Special registers]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD5500<br>SD5501 | SD5540<br>SD5541 | SD5580<br>SD5581 | SD5620<br>SD5621 | Current address (in user unit) |
| SD5502<br>SD5503 | SD5542<br>SD5543 | SD5582<br>SD5583 | SD5622<br>SD5623 | Current address (in pulse unit) |
| SD5504<br>SD5505 | SD5544<br>SD5545 | SD5584<br>SD5585 | SD5624<br>SD5625 | Current speed (in user unit) |
| SD5510 | SD5550 | SD5590 | SD5630 | Positioning error error code |

[Special relays (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SM8029 | | | | Instruction execution complete flag |
| SM8329 | | | | Instruction execution abnormal end flag |
| SM8340 | SM8350 | SM8360 | SM8370 | Pulse output monitoring |
| SM8348 | SM8358 | SM8368 | SM8378 | Positioning instruction activation |

[Special registers (FX3 compatible area)]

| Axis number | | | | Name |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| SD8136<br>SD8137 | | — | — | Total number of outputs for axis 1 and 2 of DPLSY instruction |
| SD8140<br>SD8141 | SD8142<br>SD8143 | — | — | Total number of output pulses of DPLSY instruction |
| SD8340<br>SD8341 | SD8350<br>SD8351 | SD8360<br>SD8361 | SD8370<br>SD8371 | Current address (in user unit) |

## Precautions

- The operation cannot be performed normally in an environment such as user program where the instruction cannot be executed at each scan or if the instruction is jumped by the CJ(P) instruction. However, the pulse output is continued.
- The same devices as the ones of position instruction, PWM output or general-purpose output cannot be used for the output in the DPLSY instruction.
- The following table shows how to stop the pulse output. The operation is stopped immediately in any stopping method by the DPLSY instruction. Note that the motor is stopped without deceleration and this may damage the system.

| Operation | Whether to decelerate or not | Abnormal end flag |
|---|---|---|
| Turn off the drive contact. | Stops immediately. | OFF |
| All outputs disable (Turn on the special relay.) | | ON |
| Pulse output stop command (Turn on the special relay.) | | ON |
| Pulse deceleration stop command (Turn on the special relay.) | | ON |
| Forward limit (Turn on the special relay.) | | ON |
| Reverse limit (Turn on the special relay.) | | ON |
| Set 0 for the command speed specified by (s2). | | OFF |

- If the positioning address is 0 when the DPLSY instruction is activated, pulses are output without limitation.
- Overwrite the positioning address during the pulse output to change the positioning address in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten. The positioning address becomes invalid if it is changed from 0 to a value other than 0 or from a value other than 0 to 0 during positioning operation.
- When the positioning address is changed during the pulse output, the operation is stopped immediately if the changed value is the number of pulses which have already been output or less.
- Overwrite the command speed during the pulse output to change the command speed in operation. The written value is reflected at the first time that the instruction is executed after the device is overwritten.
- When the numbers of pulses (by the pulses conversion) of the command speed and positioning address exceed the 32-bit range, an error occurs and the operation cannot be performed.
- The PLSY instruction always increases the current address because the setting of rotation direction is disabled due to the absence of direction.
- When the output mode is CW/CCW mode, output is always performed from the device set to CW.
- If reverse limit is used, it operates as forward limit.
- Do not set the value of 200 kpps or more by the frequency conversion when changing the command speed during the pulse output.
- If the command speed is set to 0 when the PLSY instruction is activated, the operation ends with an error and stops pulse output.
- If the command speed is changed to 0 during pulse output, the operation immediately stops without abnormal end flag. However, if the drive contact is not turned off, pulse output will restart when the command speed is changed.
- The command speed is changed to negative value during pulse output, it is the operation ends with an error.
- The following table shows the operation timing of the complete flag and abnormal end flag of the DPLSY instruction.

| | Complete flag (SM8029) | Abnormal end flag (SM8329) |
|---|---|---|
| ON condition | From when the output of the specified positioning address is completed until the drive contact is turned off<br>• Pulse decelerate and stop command (when unlimited pulses are being output) | From the following stops until the drive contact is turned off<br>• The specified axis is already used*1<br>• Pulse output stop command<br>• Pulse deceleration stop command (when unlimited pulses are not being output)<br>• Forward limit<br>• Reverse limit<br>• All outputs disabled<br>• Positioning address error<br>• Command speed 0 (when the DPLSY instruction is activated) |
| ON→OFF condition | • When the drive contact is turned off | • When the drive contact is turned off |

*1 The flag turns on only during one scan time when the activation contact of the instruction turns off and on.

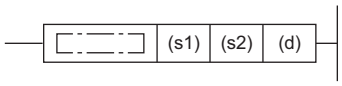
## Operation error

| Error code (SD0/SD8067) | | | | Description |
|---|---|---|---|---|
| Axis 1 | Axis 2 | Axis 3 | Axis 4 | |
| SD5510 | SD5550 | SD5590 | SD5630 | |
| 1810H | | | | The axis number specified by (d) is used by another instruction. |
| 2820H | | | | The value specified by (s) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (n) is outside the following range.<br>0 to 65535 |
| | | | | The value specified by (d) is outside the following range.<br>0 to 3 |
| 3600H | | | | The axis number specified by (d) is not set by parameters.<br>A function which is set to be not used by parameters (such as interrupt input signal 1 and zero return relations) is used. |
| 3631H | 3632H | 3633H | 3634H | The numbers of pulses (by the pulses conversion) of the positioning address specified by (n) exceed the 32-bit range. |
| 3641H | 3642H | 3643H | 3644H | The numbers of pulses (by the pulses conversion) of the command speed specified by (s) exceed the 32-bit range. |
| 3651H | 3652H | 3653H | 3654H | The operation decelerates and stops by the forward limit or reverse limit during the pulse output or at the activating of the positioning. |
| 3661H | 3662H | 3663H | 3664H | The operation decelerates and stops by the pulse output stop command or special relay whose all outputs are disabled during the pulse output or at the activating of the positioning. |

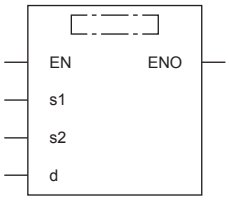
# 16 bit binary pulse width modulation

## PWM

This instruction outputs the pulse (in 16-bit data units) of the ON time (in 16-bit data units) specified by (s1) and the period specified by (s2) to the output destination specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PWM(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | ON time or the device number storing the ON time | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (s2) | Period or the device number storing the period | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d) | Channel number or device number from which pulses are to be output | — | Bit/Word | ANY_ELEMENTARY |

#### ■Applicable devices

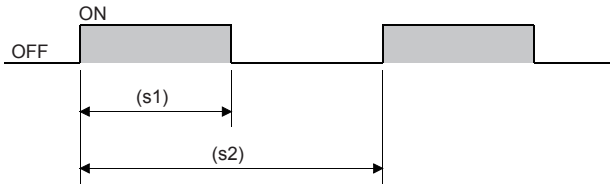
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | ○*2 | ○*2 | ○*2 | — | — | ○ | ○*2 | — | — | — |

*1 When a bit device is specified, specify one of Y0 to Y7.
Only Y can be used for a bit device.
If Y is specified, outputs are enabled when there is an unused channel number in the parameter setting and the specified Y number is not used.
The nibble of a bit device cannot be specified.

*2 When a word device or constant is specified, specify one of the channel numbers.

### Processing details

- This instruction outputs the pulse of the ON time specified by (s1) and the period specified by (s2) to the output destination specified by (d).




- Time with a unit selected on the parameter setting screen (μs or ms) can be specified by (s1) and (s2).
- The pulse output destination channel number selected on the parameter setting screen can be specified by (d).

- This instruction store the number of pulses, pulse width, and period output from each channel to an SD device. The pulse width and period are stored in the units set by the parameters. When 0 is specified in the pulse output, pulses are output without any limitation.

| Pulse output destination channel | Number of output pulses | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5301, SD5300 | R/W | 0 | • When the DHCMOV instruction is executed*1<br>• When the PWM instruction is executed<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5317, SD5316 | | | | |
| CH3 | SD5333, SD5332 | | | | |
| CH4 | SD5349, SD5348 | | | | |

| Pulse output destination channel | ON time | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5303, SD5302 | R/W | 0*2 | • When the DHCMOV instruction is executed*1<br>• When this instruction is executed*3<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5319, SD5318 | | | | |
| CH3 | SD5335, SD5334 | | | | |
| CH4 | SD5351, SD5350 | | | | |

| Pulse output destination channel | Period | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5305, SD5304 | R/W | 0*2 | • When the DHCMOV instruction is executed*1<br>• When this instruction is executed*3<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5321, SD5320 | | | | |
| CH3 | SD5337, SD5336 | | | | |
| CH4 | SD5353, SD5352 | | | | |

*1 When the DHCMOV instruction is used, the latest value can be read. A writable device can be updated immediately.
*2 Parameter setting values are set to an SD device at STOP to RUN.
*3 When this instruction is executed, the pulse width and period specified (s1) and (s2) are set to an SD device.

- After the pulse output is started from each channel, the pulse output monitor turns on.

| Pulse output destination channel | Pulse output monitor | R/W | Initial value | ON timing | OFF timing |
|---|---|---|---|---|---|
| CH1 | SM5300 | R | OFF | • When the HIOEN instruction is executed<br>• When this instruction is executed | • Power on<br>• Reset<br>• RUN→STOP/PAUSE<br>• When the specified number of pulses are output.<br>• The drive contact is turned off |
| CH2 | SM5301 | | | | |
| CH3 | SM5302 | | | | |
| CH4 | SM5303 | | | | |

- This instruction stores the number of pulses output from each channel.

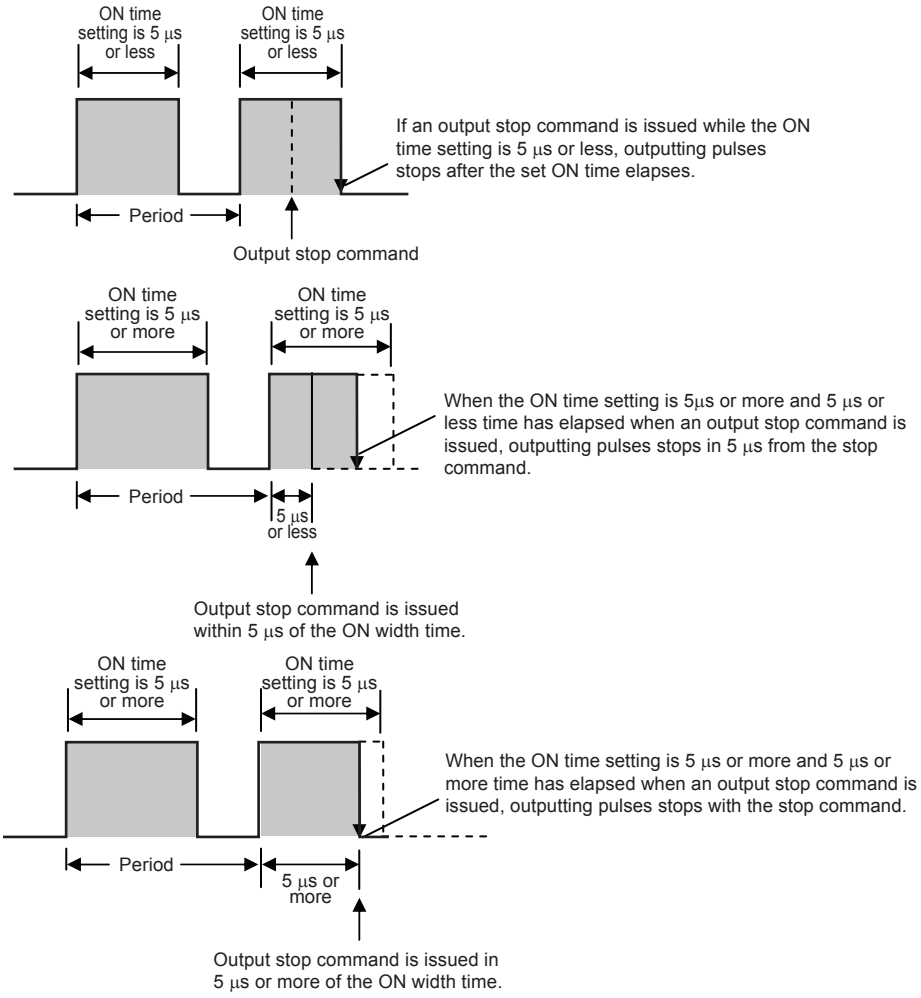
| Pulse output destination channel | Monitoring the current number of output pulses | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5307, SD5306 | R/W | 0 | • When the DHCMOV instruction is executed<br>→ An SD device is updated<br>• When the PWM instruction is executed<br>• END processing | • Power-on<br>• Reset<br>• STOP/PAUSE→RUN |
| CH2 | SD5323, SD5322 | | | | |
| CH3 | SD5339, SD5338 | | | | |
| CH4 | SD5355, SD5354 | | | | |

- The number of output pulses set to an SD device is valid for this instruction as well. The setting values are always read and updated.
- When the specified number of output pulses is equal to or less than the number of pulses which have already been output, pulse output stops after outputting pulses which are being output.
- When the specified number of output pulses is larger than the number of pulses which have already been output, pulse output stops after outputting set number of pulses.
- When the number of output pulses is set from the no limitation output setting (number of output pulses is 0), the number of output pulses is not updated (because outputting pulses continues or stops in the no limitation output).
- The maximum number of output pulses which can be output when the PWM instruction is executed once (= maximum value which can be set to an SD device) is “2147483647”.
- The ON time and period can be set during the pulse output. Setting values are always read and updated.
- When the number of output pulses is 0 (no limitation output setting), the monitor of the current number of output pulses is set to 0.

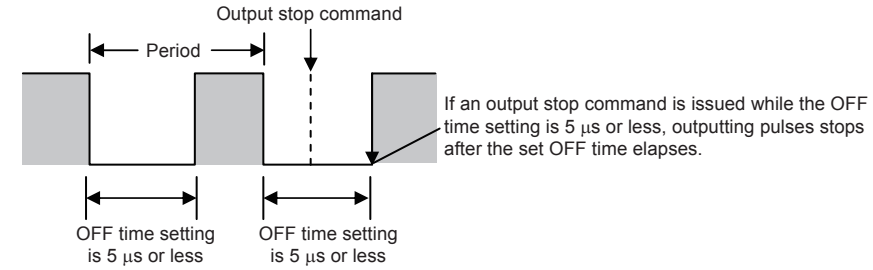
- When the number of output pulses is specified, the output pulses are monitored. When the PWM output is executed several times, the monitor of the number of output pulses is an integrated value.
- The monitor of the current number of output pulses can be changed during the pulse output.
- The monitor of the current number of output pulses is updated when the number of pulses is counted at the falling edge of pulses in the positive logic and at the rising edge of pulses in the negative logic.
- When the output always remains ON or OFF, the monitor of the current number of output pulses does not change.
- The maximum value of the monitor of the current number of output pulses is “FFFFFFFFH”. After the current number of output pulses reaches the maximum value, the monitor of the current number of output pulses starts to count again from “0”.

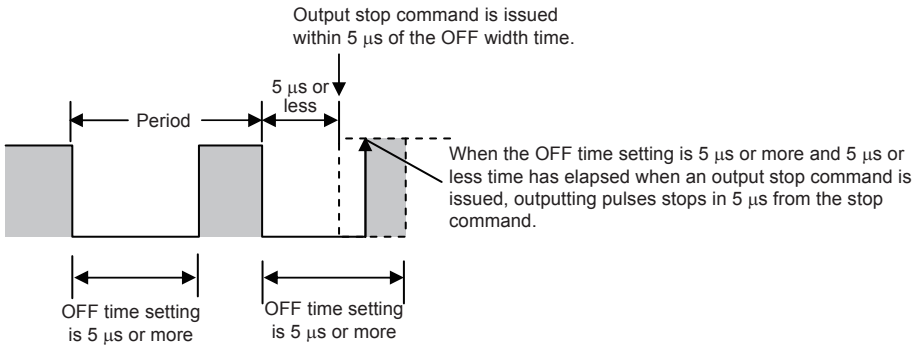
## Precautions

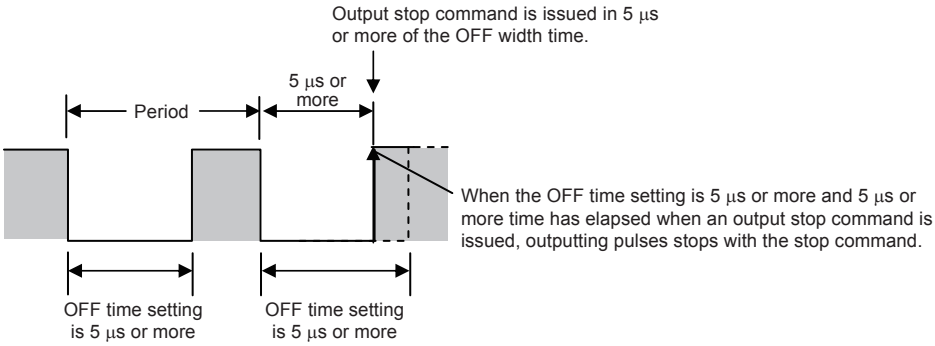
- Specify the ON time by (s1) and the period by (s2) so that [(s2)-(s1)] is equal to or larger than 3 μs.
- Specify 2 μs or more in Y0 to Y3 and 200 μs or more in Y4 to Y7 for the ON time specified by (s1), and specify 5 μs or more in Y0 to Y3 and 400 μs or more in Y4 to Y7 for the period specified by (s2).
- When a channel number that is not selected for the PWM output in the parameter setting is specified for (d), this instruction is not executed. An operation error occurs.
- Operations when the PWM output is stopped (while the output pulse is on)



- Operations when the PWM output is stopped (while the output pulse is off)




- The PWM output stops when SM8034 is on, and starts when SM8034 is off.
- PWM output does not stop, even if a pulse stop command for positioning is driven.
- When specifying the number of output pulses, executing the PWM instruction, and then outputting pulses again after the pulse output stops due to the completion of output of the specified number of pulses, turn OFF the contact which drove the PWM instruction. When driving PWM output by the HIOEN instruction, use the HIOEN instruction to turn PWM output off.
- When the period setting is equivalent to the ON time setting, the output always remains ON. The output ON state continues even after “Period x Number of output pulses” is finished in this condition.

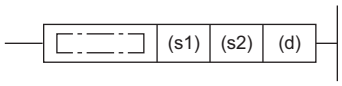
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | The output destination specified by (d) is already used by another instruction (positioning instruction).<br>(The PWM output is not executed.) |
| | A Y device is specified as the output destination specified by (d), and there is no unused channel number in the parameter setting. |
| 3405H | Y10 or later is specified as the output destination specified by (d).<br>(The PWM output stops.) |
| 3600H | A channel number that is not selected in the parameter setting are specified for the output destination specified by (d).<br>(The PWM output is not executed.) |
| 3611H(CH1)<br>3612H(CH2)<br>3613H(CH3)<br>3614H(CH4) | The ON time specified by (s1) is larger than the period specified by (s2).<br>(The PWM output stops.) |
| | The ON time or period is less than “1”. |
| | The SD device specified for the number of output pulses stores a value outside the available range (0 to 2147483647). |

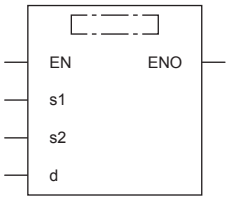
# 32 bit binary pulse width modulation

## DPWM

This instruction outputs the pulse (in 32-bit data units) of the ON time (in 32-bit data units) specified by (s1) and the period specified by (s2) to the output destination specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPWM(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | ON time or the device number storing the ON time | 1 to 2147483647 | 32-bit unsigned binary | ANY32 |
| (s2) | Period or the device number storing the period | 1 to 2147483647 | 32-bit unsigned binary | ANY32 |
| (d) | Channel number or device number from which pulses are to be output | — | Bit/Word | ANY_ELEMENTARY |

#### ■Applicable devices

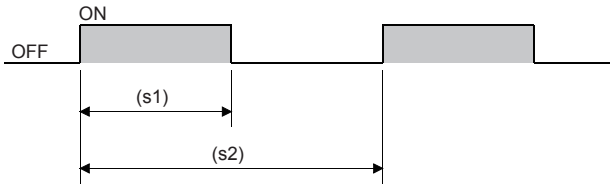
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | ○*2 | ○*2 | ○*2 | — | — | ○ | ○*2 | — | — | — |

*1 When a bit device is specified, specify one of Y0 to Y7.
Only Y can be used for a bit device.
If Y is specified, outputs are enabled when there is an unused channel number in the parameter setting and the specified Y number is not used.
The nibble of a bit device cannot be specified.

*2 When a word device or constant is specified, specify one of the CH numbers.

### Processing details

- This instruction outputs the pulse of the ON time specified by (s1) and the period specified by (s2) to the output destination specified by (d).




- Time with a unit selected on the parameter setting screen (μs or ms) can be specified by (s1) and (s2).
- The pulse output destination channel number selected on the parameter setting screen can be specified by (d).

• This instruction stores the number of pulses, pulse width, and period output from each channel to an SD device. The pulse width and period are stored in the units set by the parameters. When 0 is specified in the pulse output, pulses are output without any limitation.

| Pulse output destination channel | Number of output pulses | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5301, SD5300 | R/W | 0 | • When the DHCMOV instruction is executed*1<br>• When the DPWM instruction is executed<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5317, SD5316 | | | | |
| CH3 | SD5333, SD5332 | | | | |
| CH4 | SD5349, SD5348 | | | | |

| Pulse output destination channel | ON time | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5303, SD5302 | R/W | 0*2 | • When the DHCMOV instruction is executed*1<br>• When the DPWM instruction is executed*3<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5319, SD5318 | | | | |
| CH3 | SD5335, SD5334 | | | | |
| CH4 | SD5351, SD5350 | | | | |

| Pulse output destination channel | Period | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5305, SD5304 | R/W | 0*2 | • When the DHCMOV instruction is executed*1<br>• When the DPWM instruction is executed*3<br>• END processing | STOP/PAUSE→RUN |
| CH2 | SD5321, SD5320 | | | | |
| CH3 | SD5337, SD5336 | | | | |
| CH4 | SD5353, SD5352 | | | | |

*1 When the DHCMOV instruction is used, the latest value can be read. A writable device can be updated immediately.
*2 Parameter setting values are set to an SD device at STOP to RUN.
*3 When this instruction is executed, the pulse width and period specified (s1) and (s2) are set to an SD device.

• After the pulse output is started from each channel, the pulse output monitor turns on.

| Pulse output destination channel | Pulse output monitor | R/W | Initial value | ON timing | OFF timing |
|---|---|---|---|---|---|
| CH1 | SM5300 | R | OFF | • When the DHIOEN instruction is executed<br>• When the DPWM instruction is executed | • Power on<br>• Reset<br>• RUN→STOP/PAUSE<br>• When the specified pulse number output is terminated<br>• The drive contact is turned off |
| CH2 | SM5301 | | | | |
| CH3 | SM5302 | | | | |
| CH4 | SM5303 | | | | |

• This instruction stores the number of pulses output from each channel.

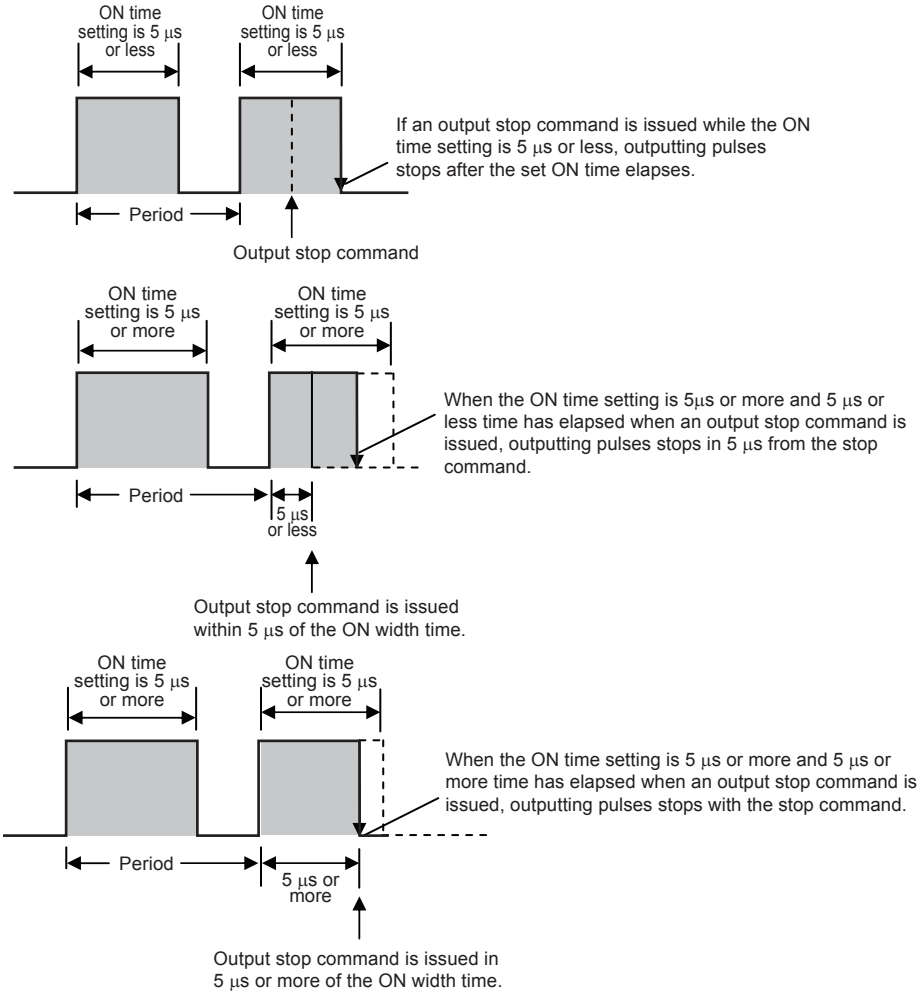
| Pulse output destination channel | Monitoring the current number of output pulses | R/W | Initial value | Timing of reflection on operation | Timing of clearing to initial value |
|---|---|---|---|---|---|
| CH1 | SD5307, SD5306 | R/W | 0 | • When the DHCMOV instruction is executed<br>→ An SD device is updated<br>• When the DPWM instruction is executed<br>• END processing | • Power-on<br>• Reset<br>• STOP/PAUSE→RUN |
| CH2 | SD5323, SD5322 | | | | |
| CH3 | SD5339, SD5338 | | | | |
| CH4 | SD5355, SD5354 | | | | |

• The number of output pulses set to an SD device is valid for this instruction as well. The setting values are always read and updated.
• When the specified number of output pulses is equal to or less than the number of pulses which have already been output, pulse output stops after outputting pulses which are being output.
• When the specified number of output pulses is larger than the number of pulses which have already been output, pulse output stops after outputting set number of pulses.
• When the number of output pulses is set from the no limitation output setting (number of output pulses is 0), the number of output pulses is not updated (because outputting pulses continues or stops in the no limitation output).
• The maximum number of output pulses which can be output when the DPWM instruction is executed once (= maximum value which can be set to an SD device) is “2147483647”
• The ON time and period can be set during the pulse output. Setting values are always read and updated.
• When the number of output pulses is 0 (no limitation output setting), the monitor of the current number of output pulses is set to 0.

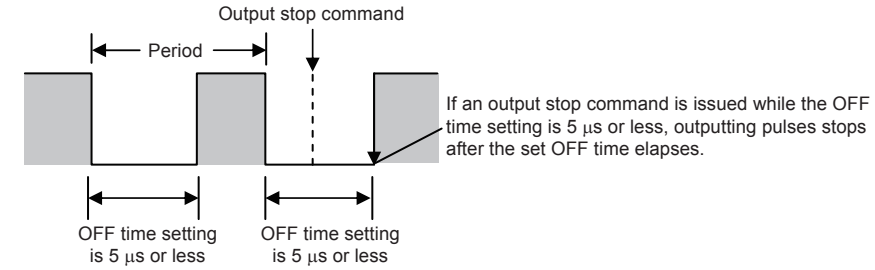
- When the number of output pulses is specified, the output pulses are monitored. When the DPWM output is executed several times, the monitor of the number of output pulses is an integrated value.
- The monitor of the current number of output pulses can be changed during the pulse output.
- The monitor of the current number of output pulses is updated when the number of pulses is counted at the falling edge of pulses in the positive logic and at the rising edge of pulses in the negative logic.
- When the output always remains ON or OFF, the monitor of the current number of output pulses does not change.
- The maximum value of the monitor of the current number of output pulses is “FFFFFFFFH”. After the current number of output pulses reaches the maximum value, the monitor of the current number of output pulses starts to count again from “0”.

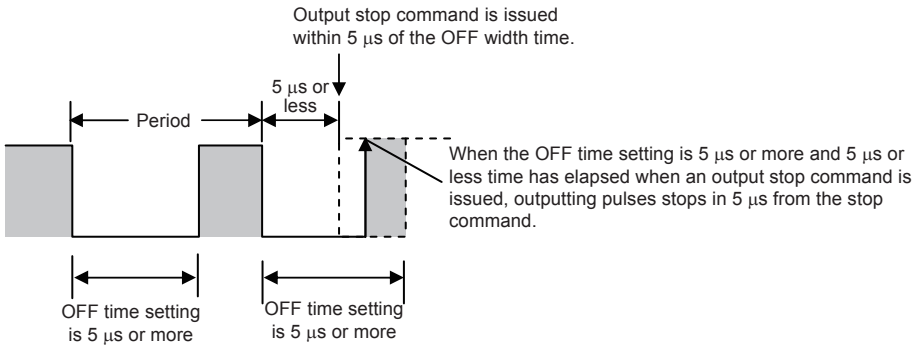
## Precautions

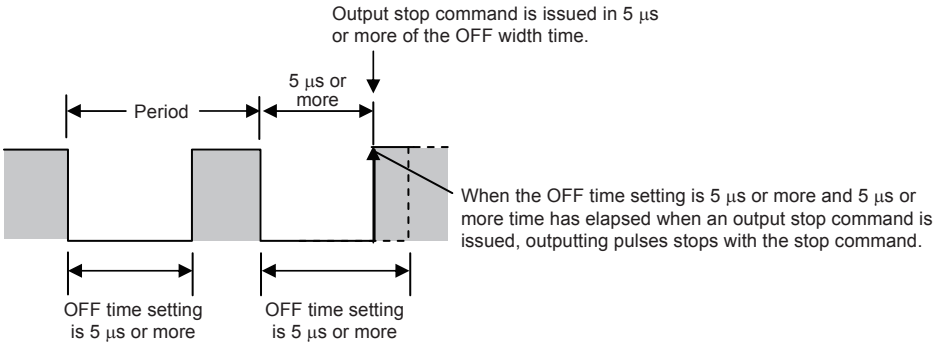
- Specify the ON time by (s1) and the period by (s2) so that [(s2)-(s1)] is equal to or larger than 3 μs.
- When a negative value is specified for the ON time by (s1) and the period by (s2), an operation error occurs. (In 16-bit instruction PWM, no error occurs.)
- Specify 2 μs or more in Y0 to Y3 and 200 μs or more in Y4 to Y7 for the ON time specified by (s1), and specify 5 μs or more in Y0 to Y3 and 400 μs or more in Y4 to Y7 for the period specified by (s2).
- When a channel number that is not selected for the PWM output in the parameter setting is specified for (d), this instruction is not executed. An operation error occurs.
- Operations when the PWM output is stopped (while the output pulse is on)



- Operations when the PWM output is stopped (while the output pulse is off)




- The PWM output stops when SM8034 is on, and starts when SM8034 is off.
- PWM output does not stop, even if a pulse stop command for positioning is driven.
- When specifying the number of output pulses, executing the PWM instruction, and then outputting pulses again after the pulse output stops due to the completion of output of the specified number of pulses, turn OFF the contact which drove the PWM instruction. When driving PWM output by the DHIOEN instruction, use the DHIOEN instruction to turn PWM output off.
- When the period setting is equivalent to the ON time setting, the output always remains ON. The output ON state continues even after “Period x Number of output pulses” is finished in this condition.

## Operation error

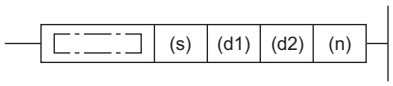
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | The output destination specified by (d) is already used by another instruction (positioning instruction).<br>(The PWM output is not executed.) |
| | A Y device is specified as the output destination specified by (d), and there is no unused channel number in the parameter setting |
| 3405H | Y10 or later is specified as the output destination specified by (d).<br>(The PWM output stops.) |
| 3600H | A channel number that is not selected in the parameter setting are specified for the output destination specified by (d).<br>(The PWM output is not executed.) |
| 3611H(CH1)<br>3612H(CH2)<br>3613H(CH3)<br>3614H(CH4) | The ON time specified by (s1) is larger than the period specified by (s2).<br>(The PWM output stops.) |
| | In (s1) and (s2), a negative value is specified.<br>(The PWM output stops.) |
| | Values of an SD device for setting pulse width and period of this instruction are incorrect.<br>(The PWM output stops.) |
| | The ON time or period is less than “1”. |
| | The SD device specified for the number of output pulses stores a value outside the available range (0 to 2147483647). |

# 8.16 Input Matrix Instruction

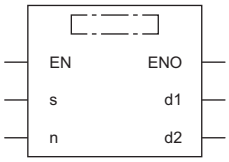
## Input matrix

### MTR

Reads matrix input as 8-point input × "n"-point output (transistor) in the time division method.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=MTR(EN, s, n, d1, d2); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Input device (X) number of matrix signal input<br>X0, X10, X20 … final input device number (Only "0" is allowed in the least significant digit of device numbers.) | — | Bit | ANYBIT_ARRAY<br>(Number of elements: 8) |
| (d1) | Head device (Y) number of matrix signal output<br>Y0, Y10, Y20 … final output device number (Only "0" is allowed in the least significant digit of device numbers.)) | — | Bit | ANY_BOOL |
| (d2) | Head bit device (Y, M or S) number of ON output destination<br>Y0, Y10, Y20 … final Y number, M0, M10, M20 … final M number or S0,S10, S20 … final S number (Only "0" is allowed in the least significant digit of device numbers.) | — | Bit | ANY_BOOL |
| (n) | Number of columns in matrix input | 2 to 8 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

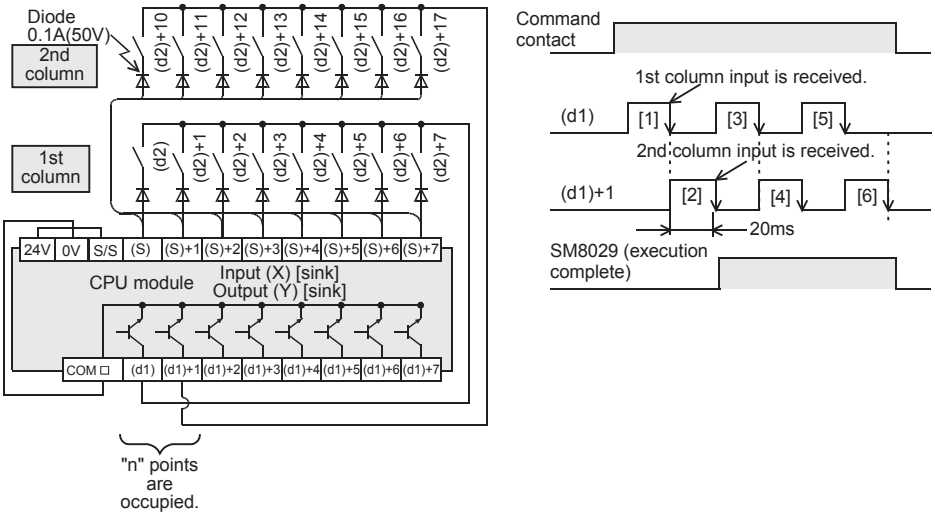
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d1) | ○*2 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*3 | — | — | — | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only X can be used.
*2 Only Y can be used.
*3 X cannot be used.

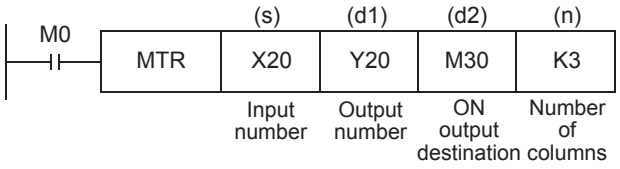
#### Processing details

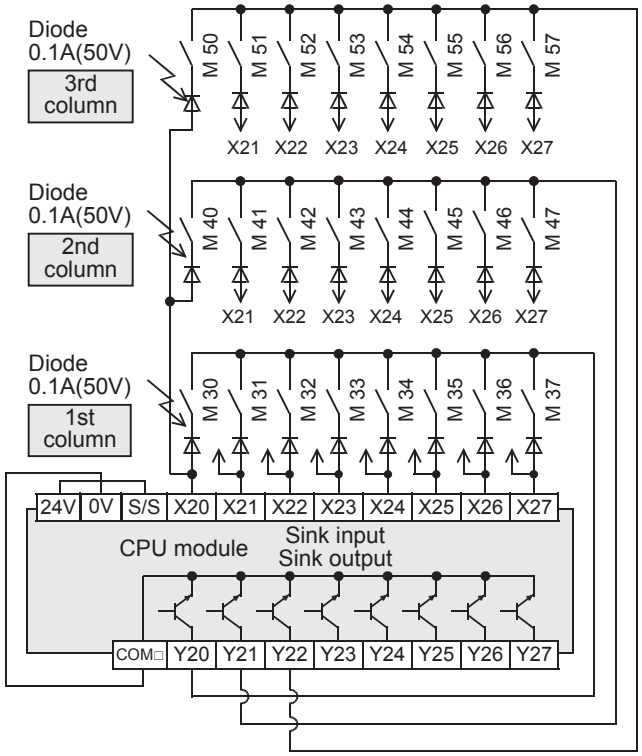
- An input signal of 8 points x "n" columns is controlled in the time division method using 8 inputs specified in (s) and transistor outputs specified in (d1). Each column is read in turn, and then output to devices specified in (d2).
- For each output, the I/O processing is executed immediately in turn in interrupt at every 10 ms or 20 ms.

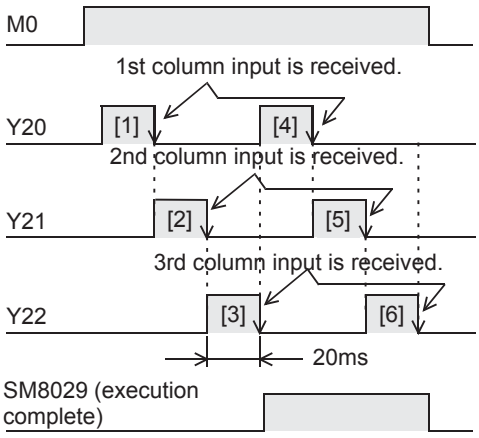
• For wiring details, refer to the hardware manual of each CPU module.




• In this program example, n=Three outputs (Y20, Y21 and Y22) are set to ON in turn repeatedly. Every time an output is set to ON, eight inputs in the 1st, 2nd and 3rd columns are received in turn repeatedly, and stored to M30 to M37, M40 to M47 and M50 to M57 respectively. For wiring details, refer to the hardware manual of each CPU module.




• The table below shows the related devices.

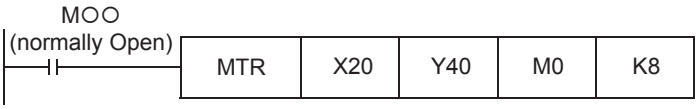
| Device | Name | Description |
|---|---|---|
| SM8029 | Instruction execution complete | ON: Turns ON after the matrix in the nth (last) column is input.<br>OFF: Remains OFF while the matrix in the 1st to nth (last) columns is being input. |

## Precautions

• Eight devices are occupied from the device specified in (s).

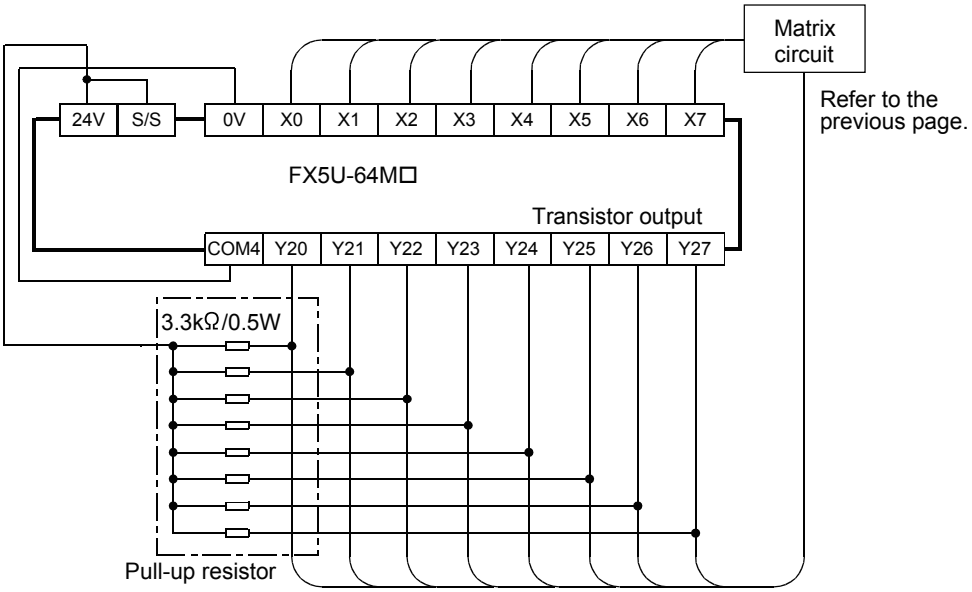
• When specifying the output in( d2) , make sure that "n" output numbers specified in (d1) does not overlap the output specified in (d2).

- The MTR instruction can only be used once in a program.
- One diode of 0.1 A/50 V is required for each switch.
- Use the transistor output format.
- If write during RUN is executed while the MTR instruction is being executed, the control right is released by the END processing. The MTR instruction executed first in the next scan will acquire the control right next.
- For the MTR instruction, set the command input to normally Open.

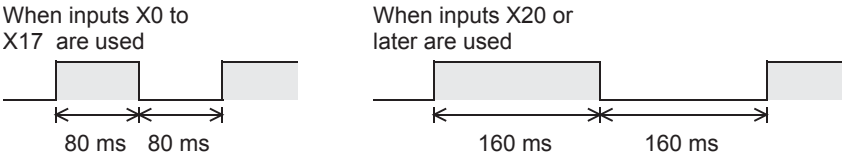



- Inputs available in MTR instruction , use inputs X20 and later under normal conditions.
- When using the inputs X0 to X17,the receiving speed is higher. Because the output transistor recovery time is long and the input sensitivity is high, however, erroneous input pulses may be counted. Change the input response time of unit parameter setting to “5 ms” to input at intervals of 10 ms. To prevent erroneous input pulses, connect pull-up resistors (3.3 kΩ/0.5 W) to transistor outputs used in MTR instruction. For pull-up resistors, use the power supply shown in the table below. The figure below shows an example of the FX5 CPU module (sink input/sink output).

| | **Power supply used for pull-up resistors transistor output** |
|---|---|
| AC power type CPU module. | Service power supply |




- Because 64 input points (8 rows × 8 columns) are received in a cycle of 80 or 160 ms, the ON/OFF duration of each input signal should be greater than or equal to the value shown below:




## Operation error

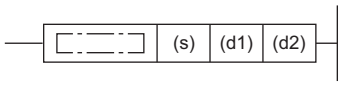
| **Error code (SD0/SD8067)** | **Description** |
|---|---|
| 3405H | The value specified by (n) is outside the following range.<br>2 to 8 |
| 2820H | The device range specified by (s) exceeds the corresponding device range. |
| | The device range specified by (d1) exceeds the corresponding device range. |
| | The device range specified by (d2) exceeds the corresponding device range. |
| 1811H | The number of times the MTR instruction is simultaneously driven exceeds 1. |
| 3582H | The MTR instruction is used in the interrupt program. |

# 8.17 Initial State

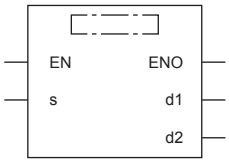
## Initial State

### IST

Automatically controls the initial state and special relays in a step ladder program.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IST(EN, s, d1, d2); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head bit device number of the selector switch in the operation mode | — | Bit | ANYBIT_ARRAY<br>(Number of elements: 8) |
| (d1) | Smallest state relay number of practical state relays in the automatic mode<br>((d1) < (d2)) | — | Bit | ANY_BOOL |
| (d2) | Largest state relay number of practical state relays in the automatic mode<br>((d1) < (d2)) | — | Bit | ANY_BOOL |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |
| (d1) | ○*2 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 S cannot be used.
*2 Only S can be used.
*3 T, ST, C cannot be used.

#### Processing details

- Specify the head input in the operation mode in (s).
- Selector switches in the operation mode occupy eight devices from the head device.

- The switch functions shown in the table below are assigned to each of the devices specified for selector switches in the operation mode. When X20 is assigned, it is necessary to set X20 to X24 as rotary switches so that they do not turn ON at the same time. It is not necessary to wire unused switches, but they cannot be used for any other purpose because they are occupied by IST instruction.

| Source | Device number (example) | Switch function | Descriptions |
|---|---|---|---|
| (s) | X20 | Individual operation | Each load is turned ON and OFF by an individual pushbutton switch. |
| (s)+1 | X21 | Return to zero point | When the pushbutton switch for zero return is pressed, the machine automatically returns to the zero point. |
| (s)+2 | X22 | Stepping | Every time the start button is pressed, the machine performs one process. |
| (s)+3 | X23 | Cycle operation | When the start button is pressed while the machine is located at the zero point, the machine performs one cycle of automatic operation and stops at the zero point.<br>If the stop button is pressed in the middle of one cycle, the machine stops immediately. When the start button is pressed after that, the machine performs the continuous operation from the last position, and automatically stops at the zero point. |
| (s)+4 | X24 | Continuous operation | When the start button is pressed while the machine is located at the zero point, the machine starts continuous operation.<br>When the stop button is pressed, the machine finishes the current cycle until the zero point, and then stops at the zero point. |
| (s)+5 | X25 | Zero return start | Starts return to the zero point. |
| (s)+6 | X26 | Automatic start | Starts the stepping operation, cycle operation or continuous operation. |
| (s)+7 | X27 | Stop | Stops each operation. |

- Specify the smallest device number of practical state relays in (d1) (for the automatic mode).
- Specify the largest device number of practical state relays in (d2) (for the automatic mode).

• While the command input is ON, the following devices are automatically switched and controlled. While the command input is OFF, the devices are not switched.

| Device number | Descriptions | ON/OFF condition | |
|---|---|---|---|
| SM8040 | STL transfer disable | ON condition | Always remains ON in the individual operation.<br>Always remains ON in the stepping operation except when the [START] button is pressed.<br>Turns ON when the [STOP] button is pressed during return to the zero point and in the cycle operation. |
| | | OFF condition | Turns OFF when the [START] button is pressed in the stepping operation.<br>Turns OFF after the [STOP] button is pressed during return to the zero point and in the cycle operation. |
| SM8041 | Transfer start | ON condition | Turns ON when the [START] button is pressed in the stepping operation and cycle operation.<br>Turns ON after the [START] button is pressed in the cycle operation. |
| | | OFF condition | Turns OFF when the operation mode is changed from RUN to STOP.<br>Always remains OFF in the individual operation and during return to the zero point.<br>Turns OFF after the [STOP] button is pressed in the continuous operation. |
| SM8042 | Start pulse | ON condition | Instantaneously turns ON when the [START] button is pressed. |
| | | OFF condition | Remains OFF except in the ON condition. |
| SM8043 | Zero return complete | ON condition | Turns ON when return to the zero point is completed (in the user program). |
| | | OFF condition | Turns OFF when the operation mode is changed from RUN to STOP.<br>Remains OFF while return to the zero point is not completed. |
| SM8044 | Zero return condition | ON condition | Turns ON when the zero point condition is established (in the user program). |
| | | OFF condition | Turns OFF when the operation mode is changed from RUN to STOP.<br>Remains OFF while return to the zero point is not completed. |
| SM8045 | All output reset disable | ON condition | Turns ON when all-output reset is not executed (in the user program). |
| | | OFF condition | Turns OFF when all-output reset is executed (in the user program). |
| SM8046 | STL state ON | ON condition | Turns ON when SM8047 (Enable STL monitoring) is ON and either step relay (device S) is ON. |
| | | OFF condition | Turns OFF when SM8047 (Enable STL monitoring) is OFF or when SM8047 (STL monitor enable) is ON and all step relays (device S) are OFF. |
| SM8047 | Enable STL monitoring | ON condition | Turns ON when the IST instruction command is given. |
| | | OFF condition | Turns OFF when the step ladder is finished (in the user program). |

| Device number | Descriptions | ON/OFF condition | |
|---|---|---|---|
| S0 | Individual operation initial state | ON condition | when a individual operation mode is selected. |
| | | OFF condition | Except a individual operation mode. |
| S1 | Zero return initial state | ON condition | when a zero return operation mode is selected. |
| | | OFF condition | Except a zero return operation mode. |
| S2 | Automatic operation initial state | ON condition | when a automatic operation mode is selected. |
| | | OFF condition | Except a automatic operation mode. |

• Do not program the following state relays as general state relays;

| Device number | Descriptions | ON/OFF condition | |
|---|---|---|---|
| S0 to S9 | Occupied for the initial state<br>• S0 to S2 are used for individual operation, zero return and automatic operation as shown above.<br>• S3 to S9 can be used arbitrarily. | ON condition | Turns ON when a step relay (device S) is selected for the initial state. |
| | | OFF condition | Turns OFF when no step relay (device S) is selected. |
| S10 to S19 | Occupied for zero return | ON condition | Turns ON when a step relay (device S) is selected for return to the zero point. |
| | | OFF condition | Turns OFF when no step relay (device S) is selected. |

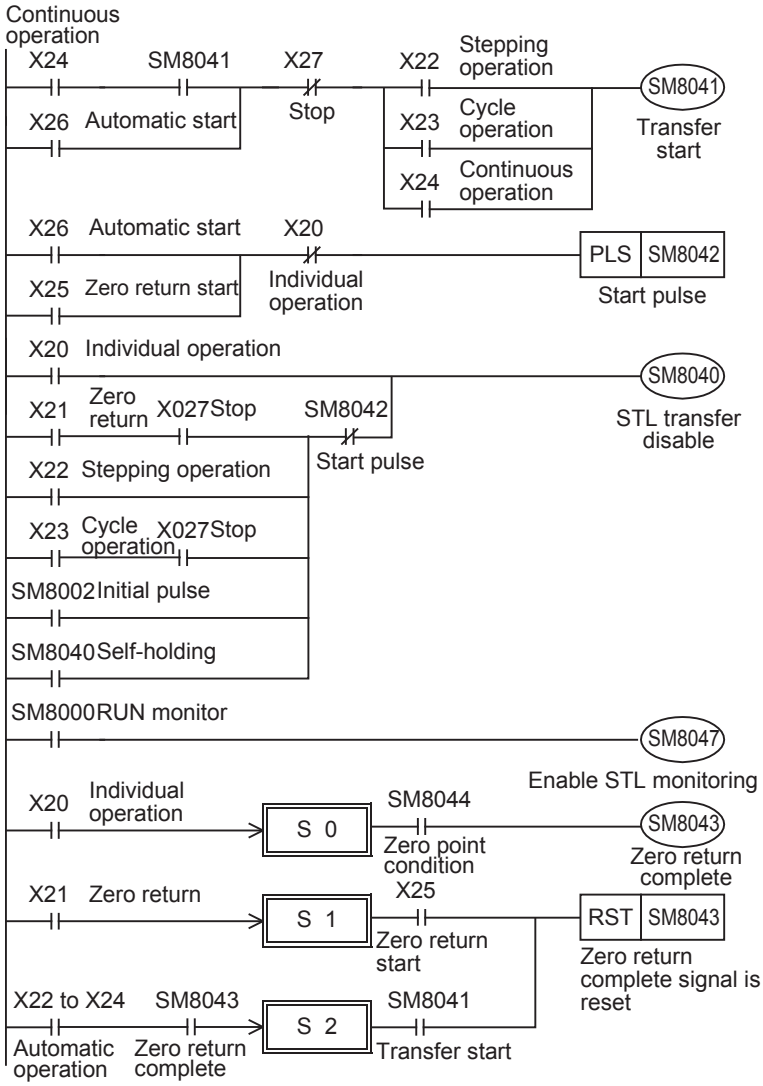
• If the devices are switched among individual operation (X20), zero return (X21) and automatic operation (X22, 23 and X24) while the zero return complete device (SM8043) is OFF, all outputs are set to OFF. Automatic operation can be started again after zero return is completed.

## Precautions

- It is not necessary to use all switches for mode selection. When some switches are not used, leave the corresponding numbers in the unused status. Such numbers cannot be used for any other purpose.
- The IST instruction should be programmed earlier than a series of STL circuit such as state relays S0 to S2.
- Use the state relays S10 to S19 for the zero return operation. In the final state in the zero return operation, set SM8043 to ON, and then let it be reset to OFF by itself.
- The IST instruction can only be used once in a program.

### ■IST instruction equivalent circuit

- The details on special relays (M) and initial state relays (S0 to S9) which are automatically controlled by the IST instruction is as shown in the equivalent circuit below. (Refer to the equivalent circuit below for reference.) This equivalent circuit cannot be programmed.

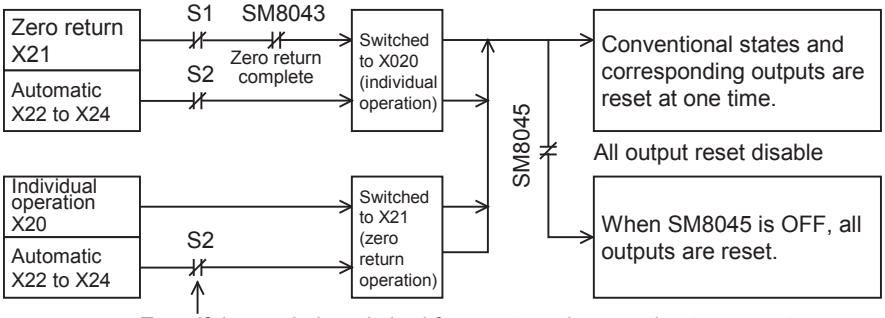



SM8041 is set to ON when the start button is pressed in the automatic mode.
Especially in the continuous mode, SM8041 holds its status by itself, and is reset when the stop button is pressed.

SM8040 is set to ON in the stepping mode, and set to OFF every time the start button is pressed.
In the zero return operation or cycle operation, SM8040 holds its status by itself when the stop button is pressed, and is reset when the start button is pressed.

The initial state is switched according to each mode input, and SM8043 is controlled at the same time.
However, it is necessary to control SM8044 and SM8043 in user programs also.

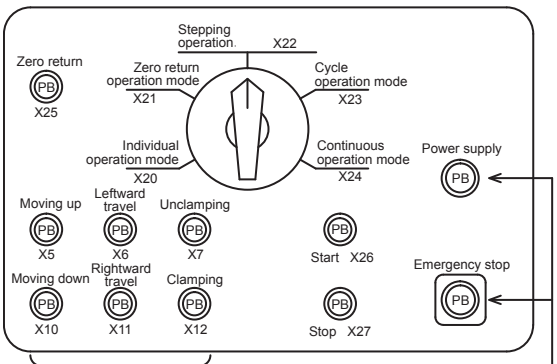
• When the operation mode is switched among the individual operation, zero return operation and automatic operation,all outputs and conventional states are reset at one time unless the machine is located in the zero point. (Reset of all outputs is not executed when SM8045 is driven.)




Even if the mode is switched from automatic operation to zero return operation while S2 is ON, state relays (except initial state relays) and outputs are not reset.

## ■Example of IST instruction introduction (example of workpiece transfer mechanism)

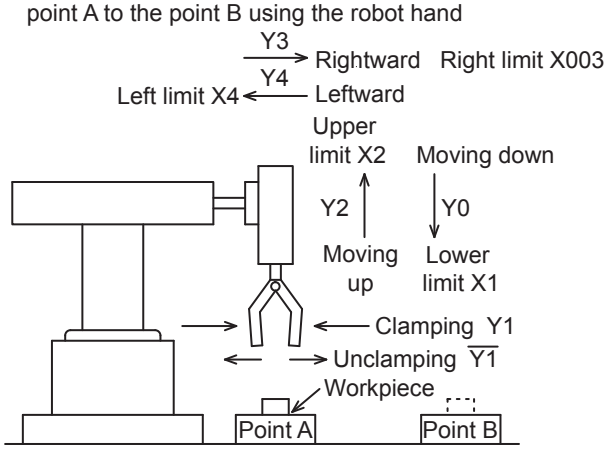
• Operation mode




Pushbutton switches for individual operations of the robot hand shown in the figure on the right

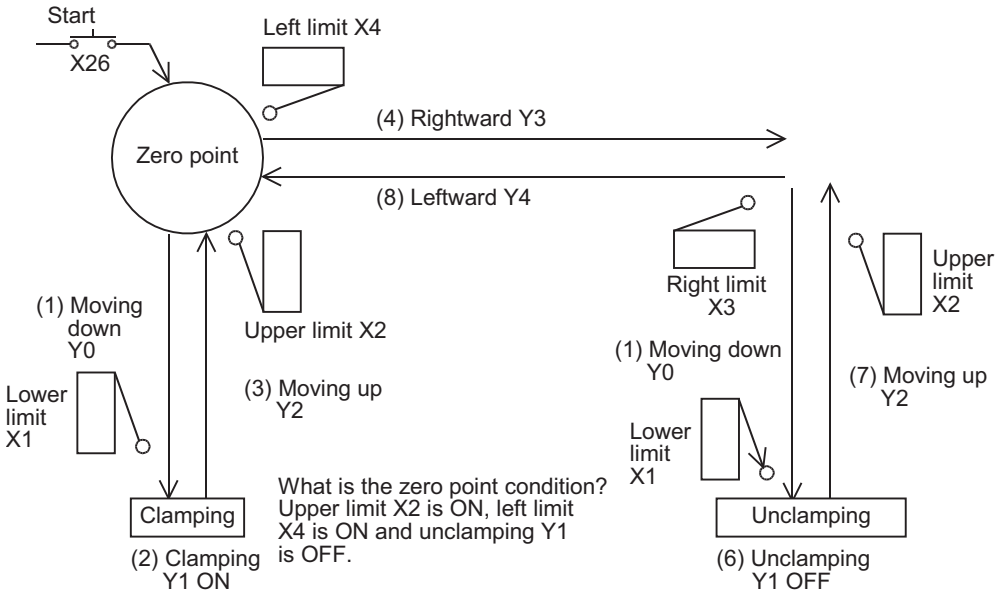
Pushbutton switches for the external circuit to turn ON and OFF the load power supply

Mechanism for transferring a workpiece from the point A to the point B using the robot hand




| Operation mode | | Contents of operation |
|---|---|---|
| Manual mode | Individual operation mode: | Each load is turned ON and OFF by an individual pushbutton switch. |
| | Zero return operation mode: | When the pushbutton switch for zero return is pressed, the machine automatically returns to the zero point. |
| Automatic mode | Stepping operation mode: | Every time the start button is pressed, the machine performs one process. |
| | Cycle operation mode | When the start button is pressed while the machine is located at the zero point, the machine performs one cycle of automatic operation and stops at the zero point.<br>If the stop button is pressed in the middle of one cycle, the machine stops immediately. When the start button is pressed after that, the machine performs the continuous operation from the last position, and automatically stops at the zero point. |
| | Continuous operation mode | When the start button is pressed while the machine is located at the zero point, the machine starts continuous operation.<br>When the stop button is pressed, the machine finishes the current cycle until the zero point, and then stops at the zero point. |

• Transfer mechanism

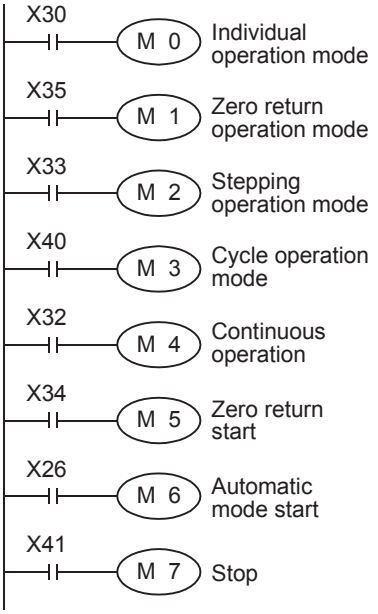



• For using IST instruction, it is necessary to assign inputs having consecutive device numbers as shown below for mode inputs. When using non-consecutive inputs or omitting some modes, change the layout by using an auxiliary relay as the head input for mode specification as shown in the figure below.

| Input device | Assignment |
|---|---|
| X20 | Individual operation mode |
| X21 | Zero return operation mode |
| X22 | Stepping operation mode |
| X23 | Cycle operation mode |
| X24 | Continuous operation mode |
| X25 | Zero return start |
| X26 | Automatic mode start |
| X27 | Stop |

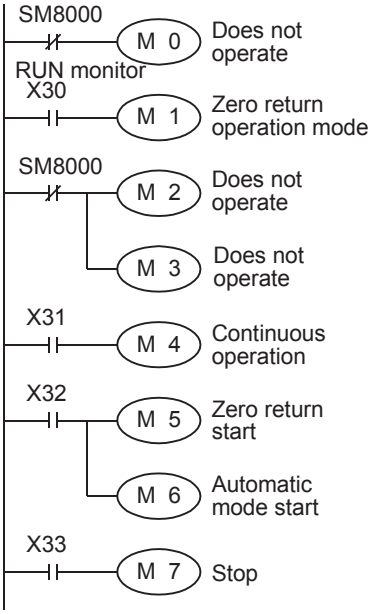
When inputs do not have consecutive device numbers

Example:
X30: Individual operation mode
X35: Zero return operation mode
X33: Stepping operation mode
X40: Cycle operation mode
X32: Continuous operation mode
X34: Zero return start
X26: Automatic mode start
X41: Stop



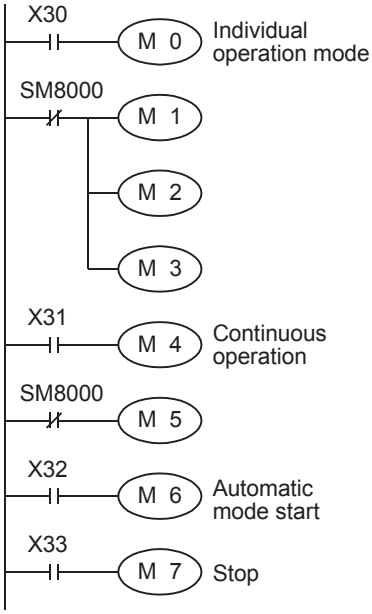

When only the continuous operation mode and zero return operation mode are used

Example:
X30: Zero return operation mode
X31: Continuous operation mode
X32: Automatic mode start zero return start
X33: Stop

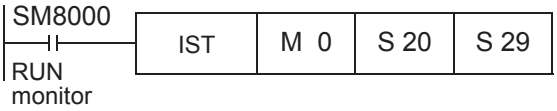



When only the continuous operation mode and individual operation mode are used

Example:
X30: Individual operation mode
X31: Continuous operation mode
X32: Automatic mode start
X33: Stop




In this example, M0 is used as the head input for mode specification.

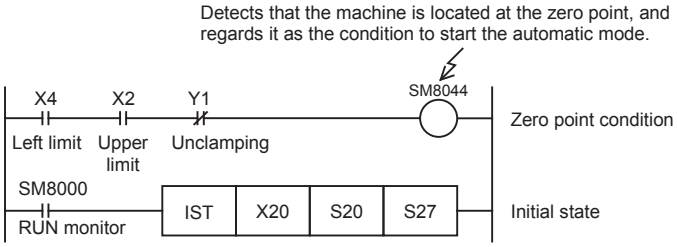


- Special relay (SM) used in the IST instruction are classified into two types. Some special relays are automatically controlled by the IST instruction itself according to the situation. Other special relays should be controlled by a program for preparation of operation or for purpose of control.

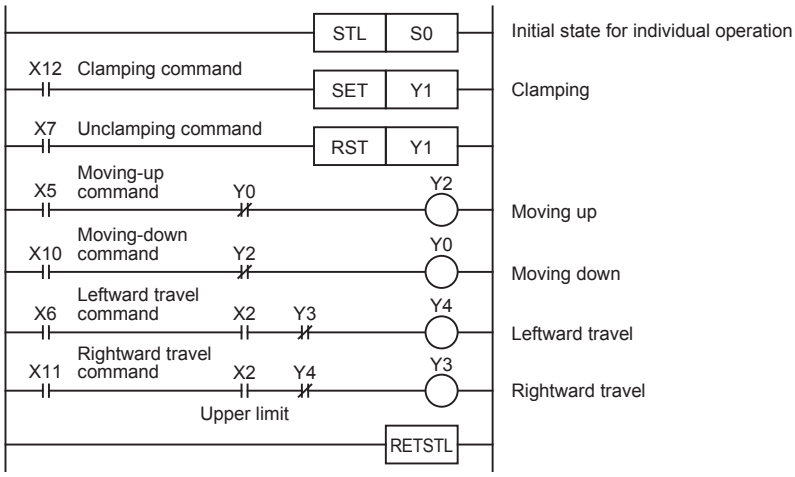
| Special relay | Description | Remarks |
|---|---|---|
| SM8040<br>(STL transfer disable) | When this special relay turns ON, transfer of every state is disabled.<br>Individual operation mode: SM8040 is always effective.<br>Zero return operation mode and cycle operation mode: When the stop button is pressed, the operation is held until the start button is pressed.<br>Stepping operation mode: SM8040 is always effective except when the start button is pressed. When the start button is pressed, SM8040 is not effective and transfer of states is allowed.<br>Others: The operation is latched when the PLC mode switches from STOP to RUN, and reset when the start button is pressed. Even in the transfer disabled status, the operation is held for outputs in the states. | Special relays automatically controlled by the IST instruction |
| SM8041<br>(Transfer start) | This special relay allows transfer from the initial state S2 to the next state.<br>Individual operation mode and zero return operation mode: SM8041 is not effective.<br>Stepping operation mode and cycle operation mode: SM8041 is effective only while the start button is pressed and held.<br>Continuous operation mode: The operation is latched when the start button is pressed, and cleared when the stop button is pressed. | |
| SM8042<br>(Start pulse) | SM8042 is activated instantaneously only when the start button is pressed. | |
| SM8047<br>(Enable STL monitoring) | When IST instruction is executed, SM8047 is set to ON.<br>When the SM8047 turns ON, STL monitoring becomes valid, and state relay numbers (S0 to S899) in the ON status are stored in turn in the ascending order of device number to the special register SD8040 to SD8047.<br>Up to eight state relay numbers in the ON status can be monitored.<br>If either state relay is ON, the special auxiliary relay SM8046 is set to ON. | |
| SM8043<br>(Zero return complete) | Set this special relay (SM) to ON by a user program when the machine returns to the zero point in the zero return operation mode. | Special relays controlled by a sequence program |
| SM8044<br>(Zero point condition) | Detect the zero point condition of the machine, and drive this special relay. This signal is effective in every mode. | |
| SM8045<br>(All output reset disable) | When the mode is switched among individual operation mode, zero return operation mode and automatic mode, all outputs and operation state relays are reset if the machine is not located at the zero point. If SM8045 has been set to ON in advance, however, only operation state relays are reset. | |

## Program example

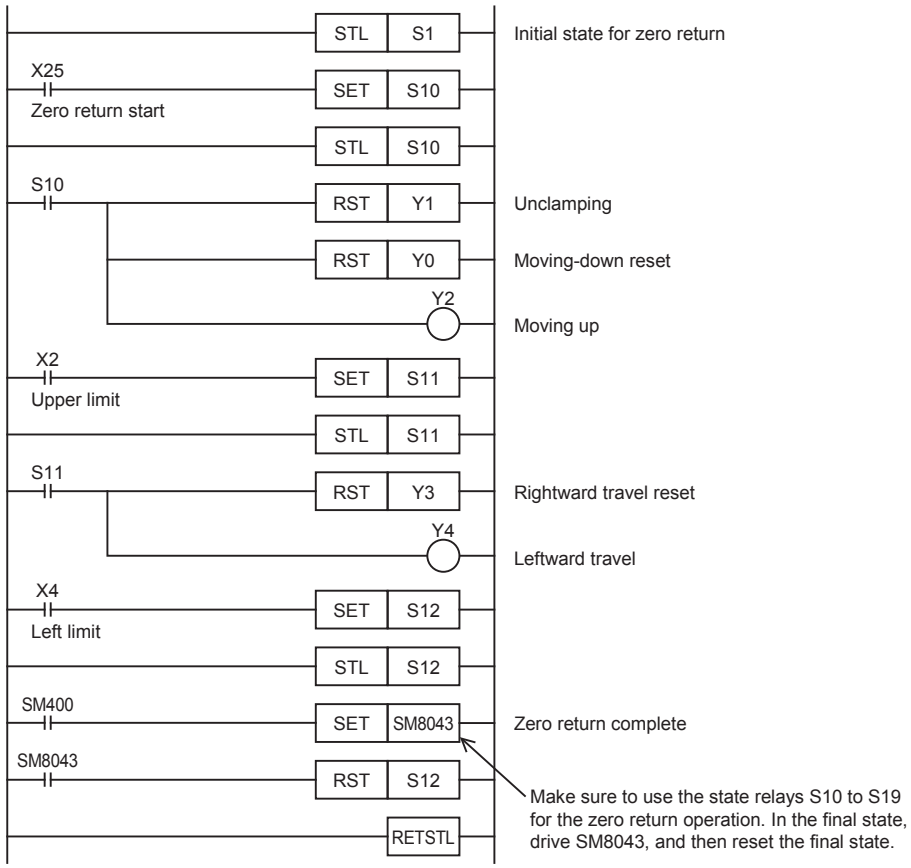
- While the machine is operating, the operation mode can be switched arbitrarily (among stepping operation, cycle operation and continuous operation) in the automatic mode. When the operation mode is switched between the individual operation mode, zero return operation mode and automatic mode while the machine is operating, all outputs are reset once to assure safety, after which the following mode becomes valid. (While SM8045 (All output reset disable) is ON, outputs are not reset at all.)



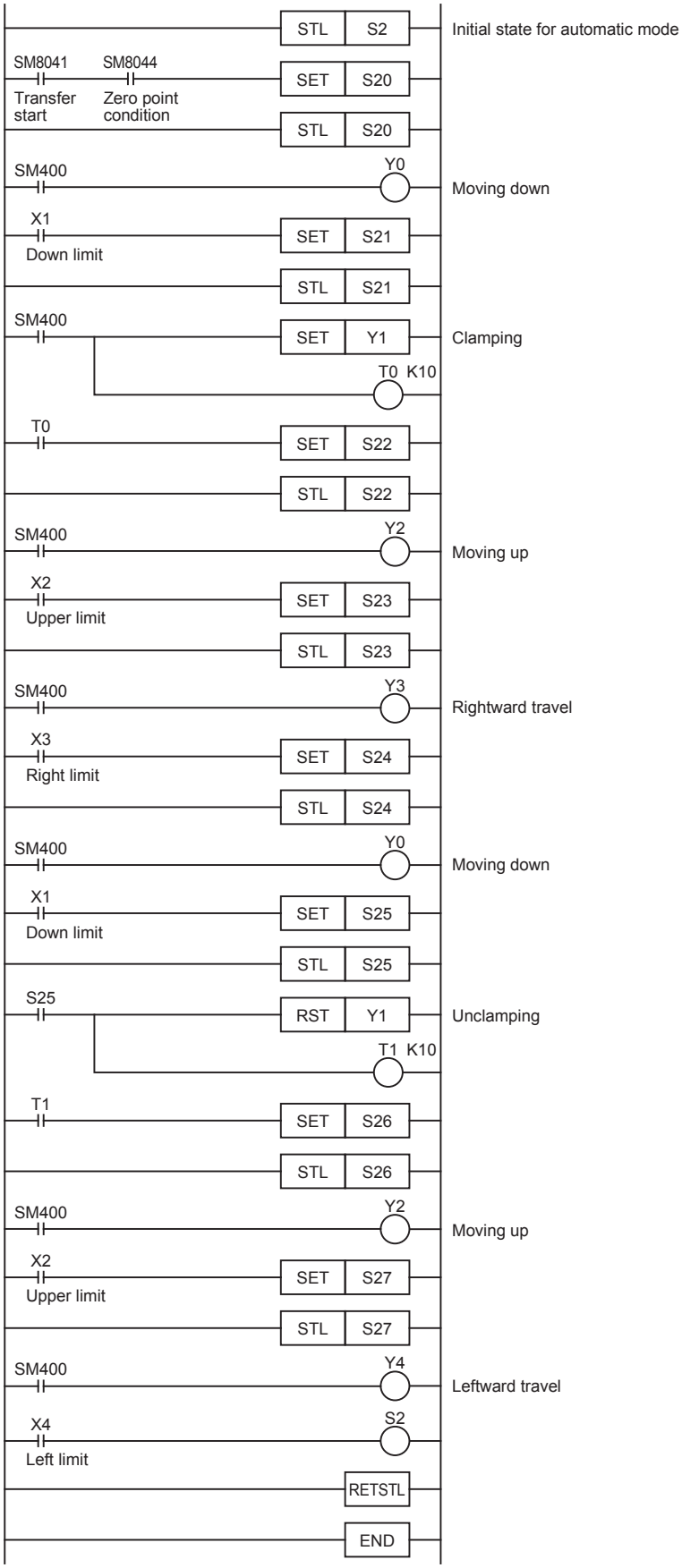
- Programming is not required when the individual operation mode is not provided.




- Programming is not required when the zero return operation mode is not provided. It is necessary to set SM8043 (zero return complete) to ON before starting the automatic mode.



• Automatic mode (stepping operation mode, cycle operation mode or continuous operation mode)



## Operation error

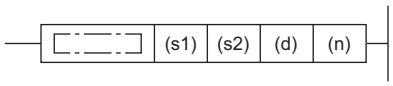
| Error code (SD0/SD8067) | Description |
|---|---|
| 1811H | The number of times the IST instruction is simultaneously driven exceeds 1. |
| 2820H | The device numbers specified in (d1) and (d2) show the following relationship:<br>(d1) ≥ (d2) |
| | Eight points are not secured from the device specified in (s). |
| | An unavailable device is set in (s). |
| | An unavailable device is set in (d1). |
| | An unavailable device is set in (d2). |

# 8.18 Drum Sequence

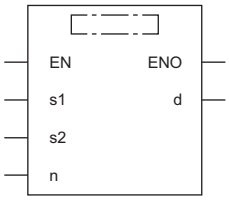
## 16-bit binary data absolute method

### ABSD

This instruction creates many output patterns corresponding to the current value (16-bit binary data) of a counter.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ABSD(EN,s1,s2,n,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number storing the data table (with rising and falling point data) | — | 16-bit signed binary | ANY16 |
| (s2) | Counter number for monitoring the current value compared with the data table | — | 16-bit signed binary | ANY16 |
| (d) | Head bit device number to be output | — | Bit | ANY_BOOL |
| (n) | Number of lines in the table and the number of output bit devices | 1 to 64 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

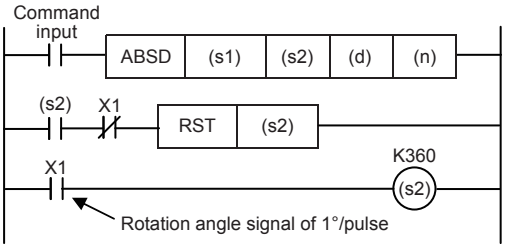
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*2 | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only C can be used.
*2 T, ST, C cannot be used.

#### Processing details

- In this example, outputs are controlled to on or off by one table rotation (0 to 360° using the rotation angle signal of 1°/pulse).

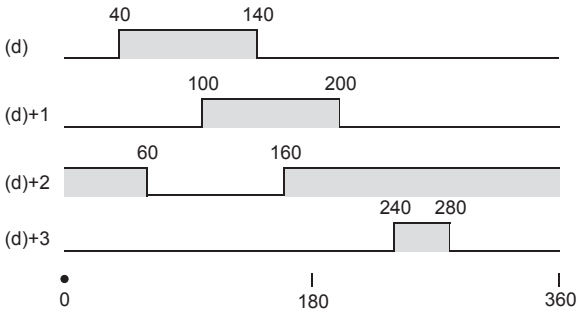
- The current value (s2) of the counter is compared with the data table with "n" lines starting from (s1) (which occupies "n" lines × 2 devices), and consecutive "n" outputs starting from (d) are controlled to on or off during one rotation.




- Write the following data to (s1) to (s1)+2(n)-1 in advance by a transfer instruction: For example, store 16-bit rising point data in even-numbered devices and 16-bit falling point data in odd-numbered devices.

| Rising point | | Falling point | | Target output |
|---|---|---|---|---|
| — | Data value (example) | — | Data value (example) | |
| (s1) | 40 | (s1)+1 | 140 | (d) |
| (s1)+2 | 100 | (s1)+3 | 200 | (d)+1 |
| (s1)+4 | 160 | (s1)+5 | 60 | (d)+2 |
| (s1)+6 | 240 | (s1)+7 | 280 | (d)+3 |
| ⋮ | — | ⋮ | — | ⋮ |
| (s1)+2(n)-2 | | (s1)+2(n)-1 | | (d)+n-1 |

- The following figure shows the output patterns for device points (n) starting from (d) when the command input is set to on. Each rising point/falling point can be changed by overwriting the data in (s1) to (s1)+2(n)-1.




## Precautions

- When specifying the nibble of a bit device to (s1), specify a multiple of 16 (0, 16, 32, 64 ...) as a device number and always specify K4 for the number of digits.
- The value of (n) determines the number of target outputs ($1 \leq (n) \leq 64$).
- Even if the command input is set to OFF, the ON/OFF status of outputs does not change.

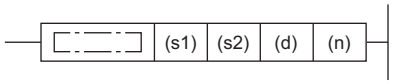
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of device points specified by (s1) or (d) is insufficient. |
| 3405H | The value specified by (n) is outside the following range.<br>1 to 64 |

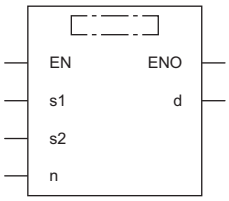
# 32-bit binary data absolute method

## DABSD

This instruction creates many output patterns corresponding to the current value (32-bit binary data) of a counter.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DABSD(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number storing the data table (with rising and falling point data) | — | 32-bit signed binary | ANY32 |
| (s2) | Counter number for monitoring the current value compared with the data table | — | 32-bit signed binary | ANY32 |
| (d) | Head bit device number to be output | — | Bit | ANY_BOOL |
| (n) | Number of lines in the table and the number of output bit devices | 1 to 64 | 16-bit unsigned binary | ANY16_U |

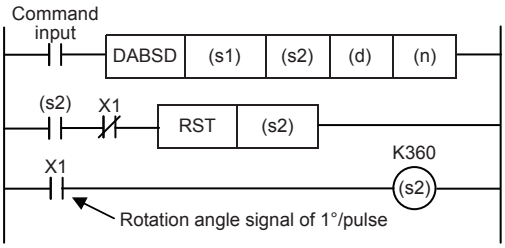
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (s2) | — | — | — | ○*1 | — | — | ○ | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*2 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only C (32 bits) can be used.
*2 T, ST, C cannot be used.

### Processing details

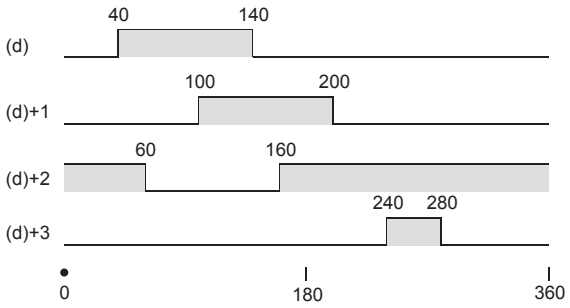
- In this example, outputs are controlled to on or off by one table rotation (0 to 360° using the rotation angle signal of 1°/pulse).
- The current value (s2) of the counter is compared with the data table with "n" lines starting from (s1) (which occupies "n" lines × 4 devices), and consecutive "n" outputs starting from (d) are controlled to on or off during one rotation.



- Write the following data to (s1), (s1)+1 to (s1)+4(n)-2, and (s1)+4(n)-1 in advance by a transfer instruction: For example, store 32-bit rising point data in even-numbered devices and 32-bit falling point data in odd-numbered devices.

| Rising point | | Falling point | | Target output |
|---|---|---|---|---|
| — | Data value (example) | — | Data value (example) | |
| (s1)+1, (s1) | 40 | (s1)+3, (s1)+2 | 140 | (d) |
| (s1)+5, (s1)+4 | 100 | (s1)+7, (s1)+6 | 200 | (d)+1 |
| (s1)+9, (s1)+8 | 160 | (s1)+11, (s1)+10 | 60 | (d)+2 |
| (s1)+13, (s1)+12 | 240 | (s1)+15, (s1)+14 | 280 | (d)+3 |
| ⋮ | — | ⋮ | — | ⋮ |
| (s1)+4(n)-3, (s1)+4(n)-4 | | (s1)+4(n)-1, (s1)+4(n)-2 | | (d)+n-1 |

- The following figure shows the output patterns for device points (n) starting from (d) when the command input is set to on. Each rising point/falling point can be changed by overwriting the data in (s1) to (s1)+2(n)-1.




## Precautions

- The DABSD instruction can specify a high-speed counter. When the high-speed counter is specified, the output pattern contains response delay caused by the scan cycle with regard to the current value of a counter.
- When specifying the nibble of a bit device to (s1), specify a multiple of 16 (0, 16, 32, 64 ...) as a device number and always specify K8 for the number of digits.
- The value of (n) determines the number of target outputs ($1 \leq (n) \leq 64$).
- Even if the command input is set to OFF, the ON/OFF status of outputs does not change.

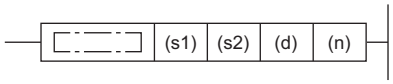
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of device points specified by (s1) or (d) is insufficient. |
| 3405H | The value specified by (n) is outside the following range.<br>1 to 64 |

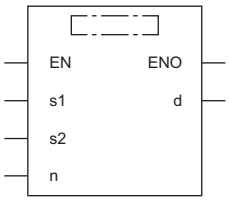
# Relative method

## INCD

This instruction creates many output patterns using a pair of counters.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=INCD(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head word device number storing the set value | — | 16-bit signed binary | ANY16 |
| (s2) | Head counter number for monitoring current value is monitored | — | 16-bit signed binary | ANY16 |
| (d) | Head bit device number to be output | — | Bit | ANY_BOOL |
| (n) | Number of output bit devices | 1 to 64 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*2 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

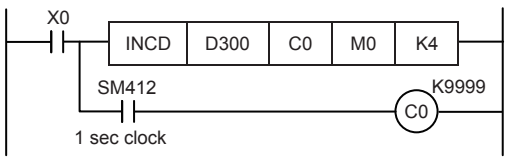
*1 Only C can be used.
*2 T, ST, C cannot be used.

### Processing details

- The current value of a counter is compared with the data table having "n" lines starting from (s1) (which occupies "n" lines × 1 device). When the value is equivalent to the table data, the current output is reset, and the next output is controlled. In this way, the ON/OFF status of specified outputs is controlled in turn.

#### ■Operation example

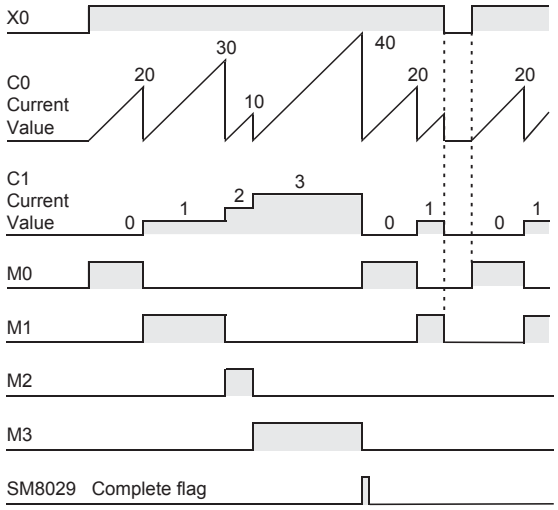
- The following ladder example shows the operation. (s2) occupies two points. In the following timing chart, C0 and C1 correspond to the two points.



• Suppose that the following data is written in advance by a transfer instruction:

| Device storing data | | Output | |
|---|---|---|---|
| — | Data value (example) | — | Example |
| (s1) | D300=20 | (d) | M0 |
| (s1)+1 | D301=30 | (d)+1 | M1 |
| (s1)+2 | D302=10 | (d)+2 | M2 |
| (s1)+3 | D303=40 | (d)+3 | M3 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| (s1)+(n)-1 | — | (d)+(n)-1 | — |

• Timing chart




• When the command contact turns on, the output M0 turns on.
• When the current value of C0 reaches the comparison value D300, the output M0 is reset, "1" is added to the count value of the process counter C1, and the current value of the counter C0 is reset.
• The next output M1 turns ON.
• When the current value of C0 reaches the comparison value D301, the output M1 is reset, "1" is added to the count value of the process counter C1, and the current value of the counter C0 is reset.
• The current value is compared for up to "n (K4)" outputs in the same way ($1 \leq (n) \leq 64$).
• When the final process specified by (n) is finished, the execution complete flag SM8029 turns on and remains on for one operation cycle. SM8029 is used for many instructions as the instruction execution complete flag. Use SM8029 as a contact just after a corresponding instruction.
• The program execution returns to the beginning, and outputs are repeated.

## Precautions

When specifying the nibble of a bit device to (s1), specify a multiple of 16 (0, 16, 32, 64 ...) as a device number.

## Operation error

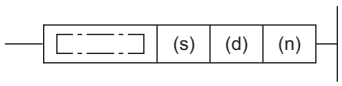
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The number of device points specified by (s1), (s2), or (d) is insufficient. |
| 3405H | The value specified by (n) is outside the following range.<br>1 to 64 |

# 8.19 Check Code

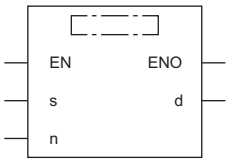
## Check code

### CCD(P)

These instructions calculate the horizontal parity value and sum check value in the error check methods used in communication. There is another check method, called CRC (cyclic redundancy check). For obtaining CRC value, use the CRC(P) instructions.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=CCD(EN,s,n,d);<br>ENO:=CCDP(EN,s,n,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number of applicable device | — | 16-bit signed binary | ANY16 |
| (d) | Head device number storing the calculated data | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 2) |
| (n) | Number of data | 1 to 32767 | 16-bit unsigned binary | ANY16_U |

**■Applicable devices**

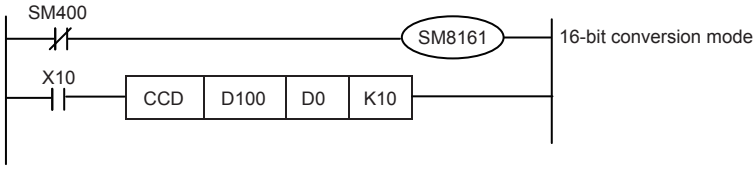
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

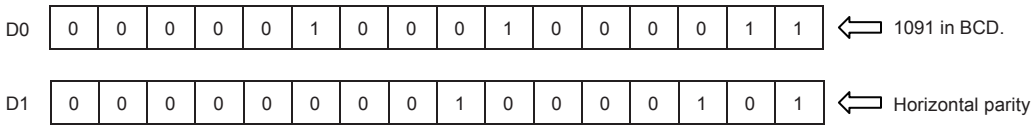
## Processing details

- These instructions calculate the addition data and horizontal parity value of data stored in (s) to (s)+(n)-1. The addition data is stored to (d), and the horizontal parity value is stored to (d)+1. The 16-bit mode and 8-bit mode are available for these instructions. For the operation in each mode, refer to the proceeding pages.
  - 16-bit conversion mode (while SM8161 is OFF)

    With regard to (n) data points starting from (s), the addition data and horizontal parity data of high-order 8 bits and low-order 8 bits are stored to (d) and (d)+1 respectively.
    SM8161 is shared with the RS2, ASCI(P), HEX(P), and CRC(P) instructions. SM8161 must always be off in the 16-bit mode.
    SM8161 is cleared when the CPU module mode is changed from RUN to STOP.
    In the following program, conversion is executed as follows:

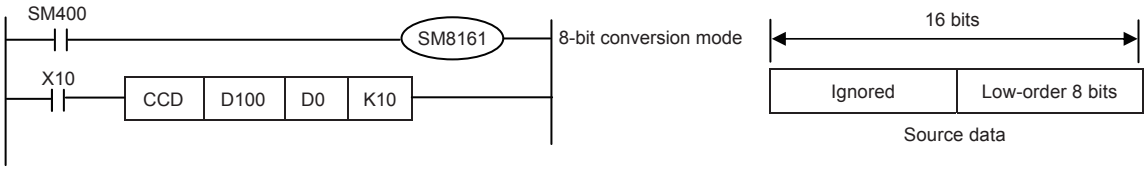



| (s) | Example of data contents | |
|---|---|---|
| D100 lowest-order byte | K100 = 0 1 1 0 0 1 0 0 | |
| D100 highest-order byte | K111 = 0 1 1 0 1 1 1 (1) | ← |
| D101 lowest-order byte | K100 = 0 1 1 0 0 1 0 0 | |
| D101 highest-order byte | K 98 = 0 1 1 0 0 0 1 0 | |
| D102 lowest-order byte | K123 = 0 1 1 1 1 0 1 (1) | ← |
| D102 highest-order byte | K 66 = 0 1 0 0 0 0 1 0 | |
| D103 lowest-order byte | K100 = 0 1 1 0 0 1 0 0 | |
| D103 highest-order byte | K 95 = 0 1 0 1 1 1 1 (1) | ← |
| D104 lowest-order byte | K210 = 1 1 0 1 0 0 1 0 | |
| D104 highest-order byte | K 88 = 0 1 0 1 1 0 0 0 | |
| Total | K1091 | |
| Horizontal parity | 1 0 0 0 0 1 0 (1) | ← When the number of "1" is odd, the horizontal parity is "1". When the number of "1" is even, the horizontal parity is "0". |

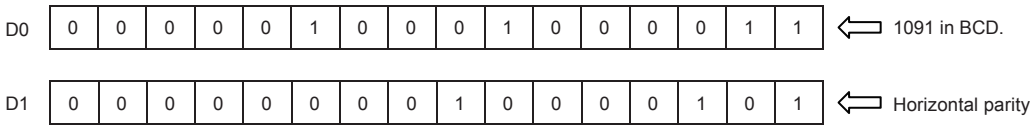


• 8-bit conversion mode (while SM8161 is ON)

With regard to (n) data points starting from (s), the addition data and horizontal parity data of only low-order 8 bits are stored to (d) and (d)+1 respectively.
SM8161 is shared with the RS2, ASCI(P), HEX(P), and CRC(P) instructions. SM8161 must always be on in the 8-bit mode.
SM8161 is cleared when the CPU module mode is changed from RUN to STOP.
In the following program, conversion is executed as follows:




| (s) | Example of data contents | |
|---|---|---|
| D100 | K100 = 0 1 1 0 0 1 0 0 | |
| D101 | K111 = 0 1 1 0 1 1 1 (1) | ← |
| D102 | K100 = 0 1 1 0 0 1 0 0 | |
| D103 | K 98 = 0 1 1 0 0 0 1 0 | |
| D104 | K123 = 0 1 1 1 1 0 1 (1) | ← |
| D105 | K 66 = 0 1 0 0 0 0 1 0 | |
| D106 | K100 = 0 1 1 0 0 1 0 0 | |
| D107 | K 95 = 0 1 0 1 1 1 1 (1) | ← |
| D108 | K210 = 1 1 0 1 0 0 1 0 | |
| D109 | K 88 = 0 1 0 1 1 0 0 0 | |
| Total | K1091 | |
| Horizontal parity | 1 0 0 0 0 1 0 (1) | ← When the number of "1" is odd, the horizontal parity is "1". When the number of "1" is even, the horizontal parity is "0". |




## Operation error

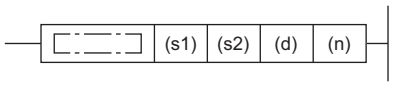
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s) or (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n) is outside the following range.<br>1 to 32767 |

# 8.20 Data Operation Instruction

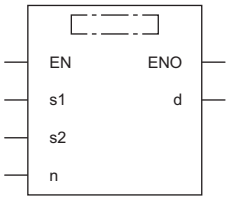
## Searching 16-bit data

### SERMM(P)

These instructions search for the same data, maximum value and minimum value in a data table.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SERMM(EN,s1,s2,n,d);<br>ENO:=SERMMP(EN,s1,s2,n,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number in which same data, maximum value and minimum value are searched | — | 16-bit signed binary | ANY16 |
| (s2) | Data to be searched for or device number storing data | — | 16-bit signed binary | ANY16 |
| (d) | Head device number storing number of same data, maximum value and minimum value detected by search | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 5) |
| (n) | Number of data in which same data, maximum value and minimum value are searched | 1 to 65535 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

#### Processing details

- These instructions search the same data as the 16-bit binary data of (s2) in (n) data starting from (s1), and store the search result in (d) to (d)+4.
- When the same data exists, five devices starting from (d) store the number of same data, first position, last position, maximum value position and minimum value position.
- When the same data does not exist, five devices starting from (d) store the number of same data, first position, last position, maximum value position and minimum value position. In this case, however, 0 is stored in three devices starting from (d) (which store the number of same data, first position and last position).

• The following table shows example of search result table configuration and data. (n=10)

| Searched device (s1) | Searched data (s1) value (example) | Comparison data (s2) value (example) | Data position | Search result | | |
|---|---|---|---|---|---|---|
| | | | | Maximum value (d)+4 | Same (d) | Minimum value (d)+3 |
| (s1) | K100 | K100 | 0 | | ○ (First time) | |
| (s1)+1 | K111 | | 1 | | | |
| (s1)+2 | K100 | | 2 | | ○ | |
| (s1)+3 | K98 | | 3 | | | |
| (s1)+4 | K123 | | 4 | | | |
| (s1)+5 | K66 | | 5 | | | ○ |
| (s1)+6 | K100 | | 6 | | ○ (Last) | |
| (s1)+7 | K95 | | 7 | | | |
| (s1)+8 | K210 | | 8 | ○ | | |
| (s1)+9 | K88 | | 9 | | | |

• The following table shows example of search result table.

| Device number | Description | Search result item |
|---|---|---|
| (d) | 3 | Number of same data |
| (d)+1 | 0 | Same data position (first position) |
| (d)+2 | 6 | Same data position (last position) |
| (d)+3 | 5 | Minimum value position (last position) |
| (d)+4 | 8 | Maximum value position (last position) |

## Precautions

- Comparison is executed algebraically. (-10<2)
- When there are two or more maximum or minimum values in the searched data, the last position of the max/min is stored respectively.
- When these instructions are driven, five devices ((d), (d)+1, (d)+2, (d)+3, and (d)+4) are occupied for storing the search result (d). Make sure that these devices are not used in other controls for the machine.

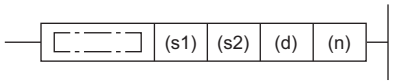
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s1) or (d) exceeds the corresponding device range. |
| 3405H | The value stored in a device specified by (n) is 0. |

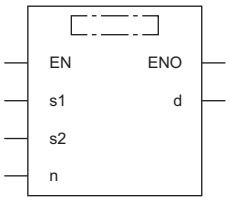
# Searching 32-bit data

## DSERMM(P)

These instructions search for the same data, maximum value and minimum value in a data table.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSERMM(EN,s1,s2,n,d);<br>ENO:=DSERMMP(EN,s1,s2,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number in which same data, maximum value and minimum value are searched | — | 32-bit signed binary | ANY32 |
| (s2) | Data to be searched for or device number storing data | — | 32-bit signed binary | ANY32 |
| (d) | Head device number storing number of same data, maximum value and minimum value detected by search | — | 32-bit signed binary | ANY32_ARRAY (Number of elements: 5) |
| (n) | Number of data in which same data, maximum value and minimum value are searched | 1 to 65535 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search the same data as the 32-bit binary data of (s2)+1 and (s2) in (n) data starting from (s1)+1 and (s1), and store the search result in (d)+1, (d) to (d)+9, and (d)+8.
- When the same data exists, five 32-bit binary data devices starting from (d)+1 and (d) store the number of same data, first position, last position, maximum value position and minimum value position.
- When the same data does not exist, five 32-bit binary data devices starting from (d)+1 and (d) store the number of same data, first position, last position, maximum value position and minimum value position. In this case, however, 0 is stored in three 32-bit devices starting from (d)+1 and (d) (which store the number of same data, first position and last position).

• The following table shows example of search result table configuration and data. (n=10)

| Searched device (s1) | Searched data (s1) value (example) | Comparison data (s2) value (example) | Data position | Search result | | |
|---|---|---|---|---|---|---|
| | | | | Maximum value (d)+9, (d)+8 | Same (d) | Minimum value (d)+7, (d)+6 |
| (s1)+1, (s) | K100000 | K100000 | 0 | | ○ (First time) | |
| (s1)+3, (s1)+2 | K110100 | | 1 | | | |
| (s1)+5, (s1)+4 | K100000 | | 2 | | ○ | |
| (s1)+7, (s1)+6 | K98000 | | 3 | | | |
| (s1)+9, (s1)+8 | K123000 | | 4 | | | |
| (s1)+11, (s1)+10 | K66000 | | 5 | | | ○ |
| (s1)+13, (s1)+12 | K100000 | | 6 | | ○ (Last) | |
| (s1)+15, (s1)+14 | K95000 | | 7 | | | |
| (s1)+17, (s1)+16 | K910000 | | 8 | ○ | | |
| (s1)+19, (s1)+18 | K910000 | | 9 | ○ | | |

• The following table shows example of search result table.

| Device number | Description | Search result item |
|---|---|---|
| (d)+1, (d) | 3 | Number of same data |
| (d)+3, (d)+2 | 0 | Same data position (first position) |
| (d)+5, (d)+4 | 6 | Same data position (last position) |
| (d)+7, (d)+6 | 5 | Minimum value position (last position) |
| (d)+9, (d)+8 | 9 | Maximum value position (last position) |

## Precautions

- Comparison is executed algebraically. (-10<2)
- When there are two or more maximum or minimum values in the searched data, the last position of the max/min is stored respectively.
- When these instructions are driven, five devices ([(d)+1,(d)], [(d)+3, (d)+2], [(d)+5, (d)+4], [(d)+7, (d)+6], and [(d)+9, (d)+8]) are occupied for storing the these result (d). Make sure that these devices are not used in other controls for the machine.

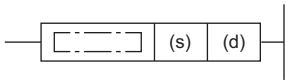
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s1) or (d) exceeds the corresponding device range. |
| 3405H | The value stored in a device specified by (n) is 0. |

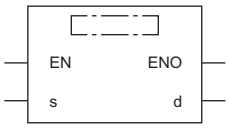
# Bit check of 16-bit data

## SUM(P)

These instructions store the total bits of 1 in the binary 16-bit data of the device specified by (s) to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SUM(EN,s,d);<br>ENO:=SUMP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

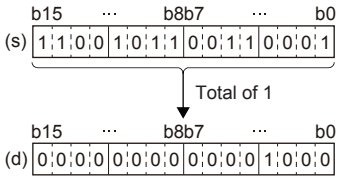
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number that counts the total bits of 1 | — | 16-bit signed binary | ANY16 |
| (d) | Head device number storing the total bits | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions store the total bits of 1 in the binary 16-bit data of the device specified by (s) to the device specified by (d).




The total of 1 is stored in the binary data. (In the example shown on the left, the total is 8.)

- When all binary 16-bit data of the device specified by (s) are 0 (off), the zero flag M8020 turns on.

### Precautions

While the command input is off, the instruction is not executed. The output of the number of bits in the on status is latched in the previous status.

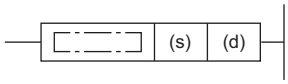
### Operation error

There is no operation error.

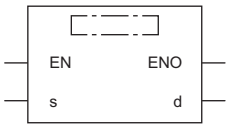
# Bit check of 32-bit data

## DSUM(P)

These instructions store the total bits of 1 in the binary 32-bit data of the device specified by (s) to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSUM(EN,s,d);<br>ENO:=DSUMP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

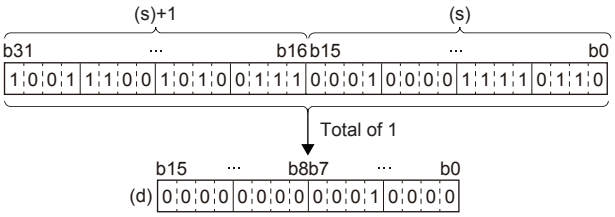
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number that counts the total bits of 1 | — | 32-bit signed binary | ANY32 |
| (d) | Head device number storing the total bits | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions store the total bits of 1 in the binary 32-bit data of the device specified by (s) to the device specified by (d).




The total of 1 is stored in the binary data. (In the example shown on the left, the total is 16.)

- When all binary 32-bit data of the device specified by (s) are 0 (off), the zero flag M8020 turns on.

### Precautions

While the command input is off, the instruction is not executed. The output of the number of bits in the on status is latched in the previous status.

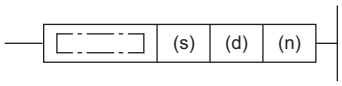
### Operation error

There is no operation error.

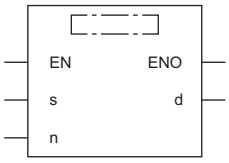
# Bit judgment of 16-bit data

## BON(P)

These instructions check whether (n) bit(s) of binary 16-bit data of the device specified by (s) are on or off, and output the result to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=BON(EN,s,n,d);<br>ENO:=BONP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Word device number storing the data | — | 16-bit signed binary | ANY16 |
| (d) | Bit device number to be driven | — | Bit | ANY_BOOL |
| (n) | Bit position to be checked | 0 to 15 | 16-bit unsigned binary | ANY16_U |

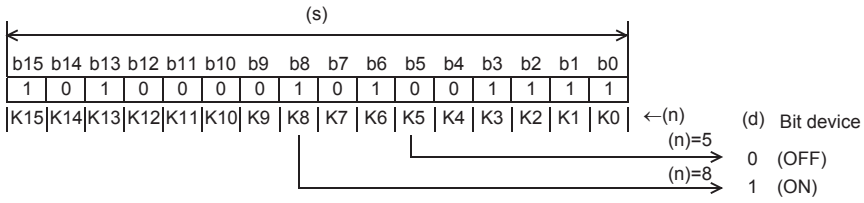
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | ○ | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions check whether (n) bit(s) of binary 16-bit data of the device specified by (s) are on or off, and output the result to the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.
- When a constant (K) is specified in the device specified by (s), it is automatically converted into binary.



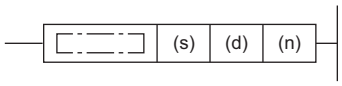

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The value specified by (n) is outside the following range.<br>0 to 15 |

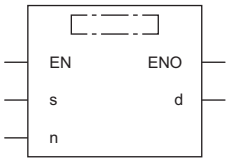
# Bit judgment of 32-bit data

## DBON(P)

These instructions check whether (n) bit(s) of binary 32-bit data of the device specified by (s) are on or off, and output the result to the device specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DBON(EN,s,n,d);<br>ENO:=DBONP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Word device number storing the data | — | 32-bit signed binary | ANY32 |
| (d) | Bit device number to be driven | — | Bit | ANY_BOOL |
| (n) | Bit position to be checked | 0 to 31 | 16-bit unsigned binary | ANY16_U |

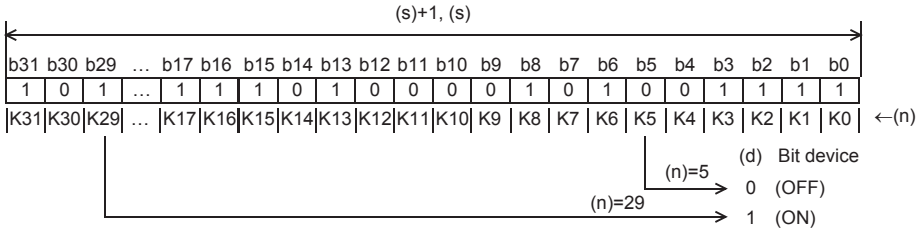
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | ○ | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- These instructions check whether (n) bit(s) of binary 32-bit data of the device specified by (s) are on or off, and output the result to the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.
- When a constant (K) is specified in the device specified by (s), it is automatically converted into binary.



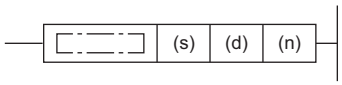

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The value specified by (n) is outside the following range.<br>0 to 31 |

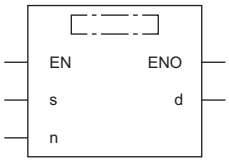
# Searching the maximum value of 16-bit data

## MAX(P)(_U)

These instructions search the maximum value from the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d).

| Ladder diagram | Structured text[*1] | |
|---|---|---|
|  | ENO:=MAXP(EN,s,n,d); | ENO:=MAXP_U(EN,s,n,d); |

**FBD/LD[*1]**




*1 The MAX and MAX_U instructions are not supported by the ST language and the FBD/LD language. Use MAX of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

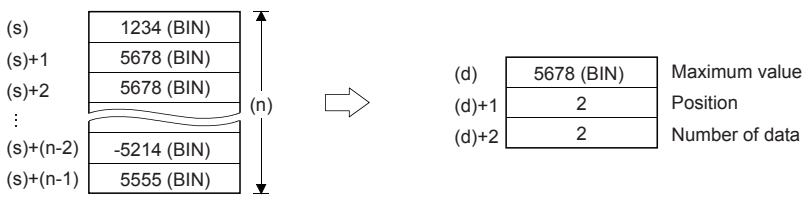
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | MAX(P) | Head device number where the maximum value is searched | — | 16-bit signed binary | ANY16_S |
| | MAX(P)_U | | | 16-bit unsigned binary | ANY16_U |
| (d) | MAX(P) | Head device number for storing the maximum value | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 3) |
| | MAX(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 3) |
| (n) | | Number of data to be searched | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search the maximum value from the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). These instructions start searching from the device specified by (s), and store the location from (s) of the first maximum value in (d)+1 and the number of maximum values in (d)+2.



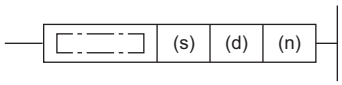
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) point(s) of data in the device starting from the one specified by (s) exceed the corresponding device range. |
| | The device specified by (d) exceeds the corresponding device range. |

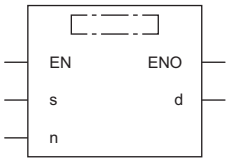
# Searching the maximum value of 32-bit data

## DMAX(P)(_U)

These instructions search the maximum value from the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d).

| Ladder diagram | Structured text[*1] | |
|---|---|---|
|  | ENO:=DMAXP(EN,s,n,d); | ENO:=DMAXP_U(EN,s,n,d); |

**FBD/LD[*1]**




*1 The DMAX and DMAX_U instructions are not supported by the ST language and the FBD/LD language. Use MAX of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

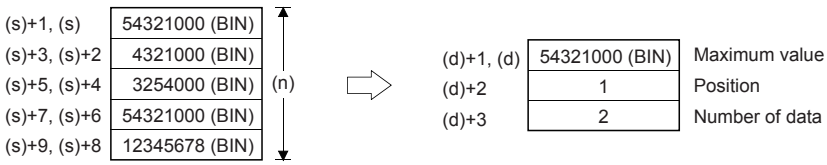
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DMAX(P) | Head device number where the maximum value is searched | — | 32-bit signed binary | ANY32_S |
| | DMAX(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (d) | DMAX(P) | Head device number for storing the maximum value | — | 32-bit signed binary | ANY32_S_ARRAY (Number of elements: 4) |
| | DMAX(P)_U | | | 32-bit unsigned binary | ANY32_U_ARRAY (Number of elements: 4) |
| (n) | | Number of data to be searched | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search the maximum value from the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d) and (d)+1. These instructions start searching from the device specified by (s), and store the location from (s) of the first minimum value in (d)+2 and the number of maximum values in (d)+3.



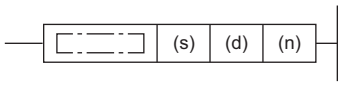
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) point(s) of data in the device starting from the one specified by (s) exceed the corresponding device range. |
| | The device specified by (d) exceeds the corresponding device range. |

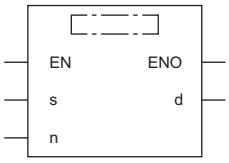
# Searching the minimum value of 16-bit data

## MIN(P)(_U)

These instructions search the minimum value from the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the minimum value in the device specified by (d).

| Ladder diagram | Structured text[*1] | |
|---|---|---|
|  | ENO:=MINP(EN,s,n,d); | ENO:=MINP_U(EN,s,n,d); |

**FBD/LD[*1]**




*1 The MIN and MIN_U instructions are not supported by the ST language and the FBD/LD language. Use MIN of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

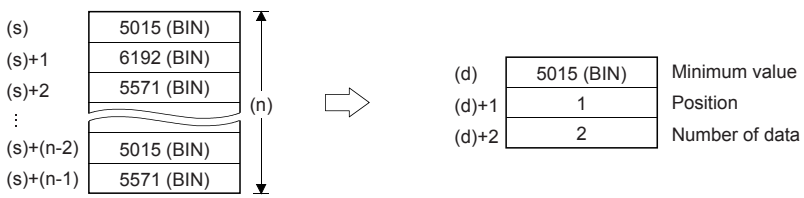
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | MIN(P) | Head device number where the minimum value is searched | — | 16-bit signed binary | ANY16_S |
| | MIN(P)_U | | | 16-bit unsigned binary | ANY16_U |
| (d) | MIN(P) | Head device number for storing the minimum value | — | 16-bit signed binary | ANY16_S_ARRAY (Number of elements: 3) |
| | MIN(P)_U | | | 16-bit unsigned binary | ANY16_U_ARRAY (Number of elements: 3) |
| (n) | | Number of data to be searched | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search the minimum value from the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the minimum value in the device specified by (d). These instructions start searching from the device specified by (s), and store the location from (s) of the first minimum value in (d)+1 and the number of minimum values in (d)+2.



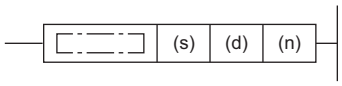
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) point(s) of data in the device starting from the one specified by (s) exceed the corresponding device range. |
| | The device specified by (d) exceeds the corresponding device range. |

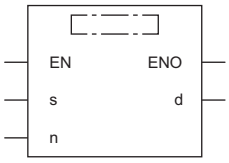
# Searching the minimum value of 32-bit data

## DMIN(P)(_U)

These instructions search the minimum value from the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the minimum value in the device specified by (d).

| Ladder diagram | Structured text*1 | |
|---|---|---|
|  | ENO:=DMINP(EN,s,n,d); | ENO:=DMINP_U(EN,s,n,d); |

**FBD/LD*1**




*1 The DMIN and DMIN_U instructions are not supported by the ST language and the FBD/LD language. Use MIN of the standard function.
☞ Page 899 MAX(_E), MIN(_E)

### Setting data

#### ■Descriptions, ranges, and data types

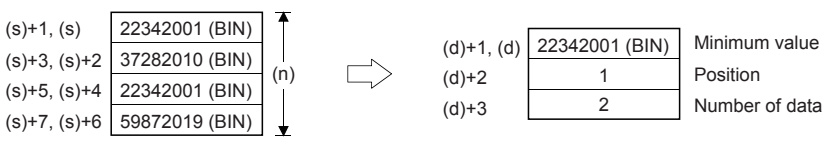
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DMIN(P) | Head device number where the minimum value is searched | — | 32-bit signed binary | ANY32_S |
| | DMIN(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (d) | DMIN(P) | Head device number for storing the minimum value | — | 32-bit signed binary | ANY32_S_ARRAY (Number of elements: 4) |
| | DMIN(P)_U | | | 32-bit unsigned binary | ANY32_U_ARRAY (Number of elements: 4) |
| (n) | | Number of data to be searched | 0 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions search the minimum value from the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the minimum value in the device specified by (d) and (d)+1. These instructions start searching from the device specified by (s), and store the location from (s) of the first minimum value in (d)+2 and the number of minimum values in (d)+3.



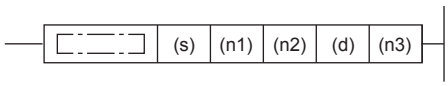
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) point(s) of data in the device starting from the one specified by (s) exceed the corresponding device range. |
| | The device specified by (d) exceeds the setting area in the device/label memory. |

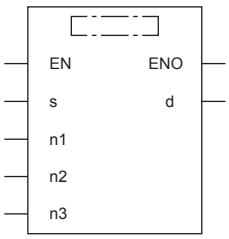
# Sorting 16-bit data

## SORTTBL(_U)

These instructions sort data lines in the data table (sorting source) having ((n1)×(n2)) points specified by (s) in the ascending order based on the group data in the column number (n3), and store the result in the data table (sorting result) having ((n1)×(n2)) points specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SORTTBL(EN,s,n1,n2,n3,d);<br>ENO:= SORTTBL_U(EN,s,n1,n2,n3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | SORTTBL | Head device number storing the data table | — | 16-bit signed binary | ANY16 |
| | SORTTBL_U | | | 16-bit unsigned binary | ANY16_U |
| (n1) | | Number of data (lines) | 1 to 32 | 16-bit unsigned binary | ANY16_U |
| (n2) | | Number of group data (columns) | 1 to 6 | 16-bit unsigned binary | ANY16_U |
| (d) | SORTTBL | Head device number for storing the operation result | — | 16-bit signed binary | ANY16 |
| | SORTTBL_U | | | 16-bit unsigned binary | ANY16_U |
| (n3) | | Column number of group data (column) used as the basis of sorting | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions sort data lines in the data table (sorting source) having ((n1)×(n2)) points specified by (s) in the ascending order based on the group data in the column number (n3), and store the result in the data table (sorting result) having (n1×n2) points specified by (d).

- The data table configuration is explained in an example in which the sorting source data table has 3 lines and 4 columns (n1 = K3, n2 = K4). For the sorting result data table, understand (s) as (d).

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 3 | Line No. 1 | (s) | (s)+3 | (s)+6 | (s)+9 |
| | Line No. 2 | (s)+1 | (s)+4 | (s)+7 | (s)+10 |
| | Line No. 3 | (s)+2 | (s)+5 | (s)+8 | (s)+11 |

- When the command input turns on, data sorting is started. Data sorting is completed after (n1) scans, and the instruction execution complete flag SM8029 is set to on.
- The following table shows an operation example based on the sorting source data below. It is recommended to put a serial number such as a control number in the first column so that the original line number can be estimated based on the contents.

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (s) | (s)+5 | (s)+10 | (s)+15 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 2 | (s)+1 | (s)+6 | (s)+11 | (s)+16 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 3 | (s)+2 | (s)+7 | (s)+12 | (s)+17 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 4 | (s)+3 | (s)+8 | (s)+13 | (s)+18 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 5 | (s)+4 | (s)+9 | (s)+14 | (s)+19 |
| | | 5 | 150 | 50 | 45 |

- Sorting result when the instructions are executed with (n3) = K2 (column No. 2)

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d) | (d)+5 | (d)+11 | (d)+15 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 2 | (d)+2 | (d)+6 | (d)+10 | (d)+16 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 3 | (d)+1 | (d)+7 | (d)+12 | (d)+17 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 4 | (d)+3 | (d)+8 | (d)+13 | (d)+18 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 5 | (d)+4 | (d)+9 | (d)+14 | (d)+19 |
| | | 2 | 180 | 50 | 40 |

• Sorting result when the instructions are executed with (n3) = K3 (column No. 3)

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d) | (d)+5 | (d)+10 | (d)+15 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 2 | (d)+1 | (d)+6 | (d)+11 | (d)+16 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 3 | (d)+2 | (d)+7 | (d)+12 | (d)+17 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 4 | (d)+3 | (d)+8 | (d)+13 | (d)+18 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 5 | (d)+4 | (d)+9 | (d)+14 | (d)+19 |
| | | 3 | 160 | 70 | 30 |

## Precautions

- Do not change the contents of operands and data during operation.
- To execute these instructions again, set the command input to off once, then on again.
- These instructions can only be used once in any program.
- When specifying the same device in (s) and (d), the source data is overwritten by the data acquired by sorting. Take special care so that the contents of (s) are not changed until execution is completed.

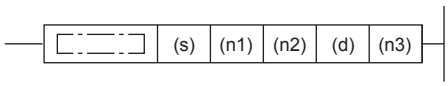
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 1811H | These instructions are used more than once. |
| 2820H | The device range specified by (s) exceeds the corresponding device range. |
| | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n1) is outside the following range.<br>1 to 32 |
| | The value specified by (n2) is outside the following range.<br>1 to 6 |
| | The value specified by (n3) is outside the following range.<br>1 to (n2) |

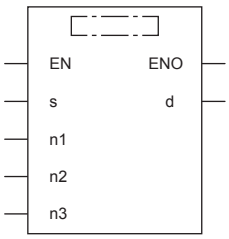
# 16-bit data alignment 2

## SORTTBL2(_U)

These instructions sort data lines in the data table (sorting source) of 16-bit binary data having (n1×n2) points specified by (s) in the ascending order or descending order based on the group data in the column number (n3), and store the result in the data table (sorting result) of 16-bit binary data having ((n1)×(n2)) points specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SORTTBL2(EN,s,n1,n2,n3,d);<br>ENO:=SORTTBL2_U(EN,s,n1,n2,n3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | SORTTBL2 | Head device number storing the data table | — | 16-bit signed binary | ANY16 |
| | SORTTBL2_U | | | 16-bit unsigned binary | ANY16_U |
| (n1) | | Number of data (lines) | 1 to 32 | 16-bit unsigned binary | ANY16_U |
| (n2) | | Number of group data (columns) | 1 to 6 | 16-bit unsigned binary | ANY16_U |
| (d) | SORTTBL2 | Head device number for storing the operation result | — | 16-bit signed binary | ANY16 |
| | SORTTBL2_U | | | 16-bit unsigned binary | ANY16_U |
| (n3) | | Column number of group data (column) used as the basis of sorting | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions sort data lines in the data table (sorting source) of 16-bit binary data having (n1×n2) points specified by (s) in the ascending order or descending order based on the group data in the column number (n3), and store the result in the data table (sorting result) of 16-bit binary data having ((n1)×(n2)) points specified by (d).

- The data table configuration is explained in an example in which the sorting source data table has 3 lines and 4 columns (n1 = K3, n2 = K4). For the sorting result data table, understand (s) as (d).

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 3 | Line No. 1 | (s) | (s)+1 | (s)+2 | (s)+3 |
| | Line No. 2 | (s)+4 | (s)+5 | (s)+6 | (s)+7 |
| | Line No. 3 | (s)+8 | (s)+9 | (s)+10 | (s)+11 |

- Set the sorting order by setting SM703 to on or off.

| | Sorting order |
|---|---|
| SM703 = ON | Descending order |
| SM703 = OFF | Ascending order |

- When the command input turns on, data sorting is started. Data sorting is completed after (n1) scans, and the instruction execution complete flag SM8029 is set to on.
- The following table shows an operation example based on the sorting source data below. It is recommended to put a serial number such as a control number in the first column so that the original line number can be estimated based on the contents.

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (s) | (s)+1 | (s)+2 | (s)+3 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 2 | (s)+4 | (s)+5 | (s)+6 | (s)+7 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 3 | (s)+8 | (s)+9 | (s)+10 | (s)+11 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 4 | (s)+12 | (s)+13 | (s)+14 | (s)+15 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 5 | (s)+16 | (s)+17 | (s)+18 | (s)+19 |
| | | 5 | 150 | 50 | 45 |

- Sorting result when the instructions are executed with (n3) = K2 (column No. 2) (in the case of ascending order SM703=OFF)

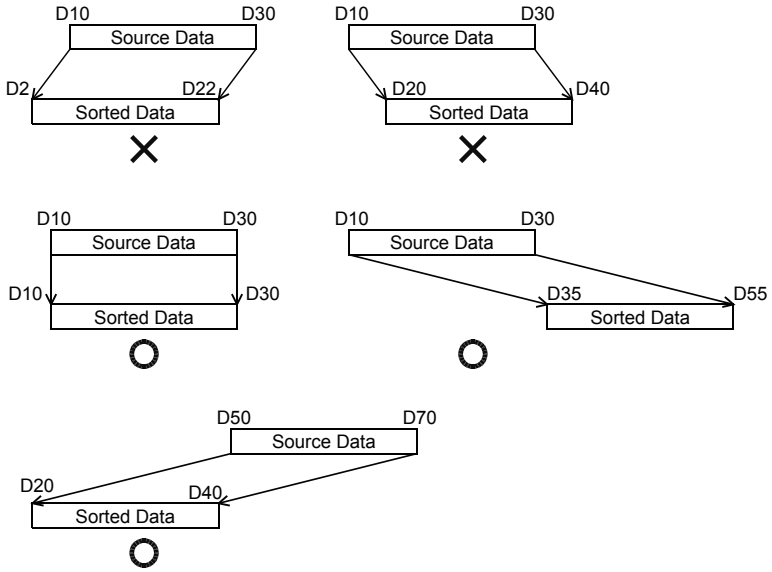
| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d) | (d)+1 | (d)+2 | (d)+3 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 2 | (d)+4 | (d)+5 | (d)+6 | (d)+7 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 3 | (d)+8 | (d)+9 | (d)+10 | (d)+11 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 4 | (d)+12 | (d)+13 | (d)+14 | (d)+15 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 5 | (d)+16 | (d)+17 | (d)+18 | (d)+19 |
| | | 2 | 180 | 50 | 40 |

- Sorting result when the instructions are executed with (n3) = K3 (column No. 3) (in the case of descending order SM703=ON)

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d) | (d)+1 | (d)+2 | (d)+3 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 2 | (d)+4 | (d)+5 | (d)+6 | (d)+7 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 3 | (d)+8 | (d)+9 | (d)+10 | (d)+11 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 4 | (d)+12 | (d)+13 | (d)+14 | (d)+15 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 5 | (d)+16 | (d)+17 | (d)+18 | (d)+19 |
| | | 4 | 100 | 20 | 8 |

## Precautions

- Do not change the contents of operands and data during operation.
- To execute these instructions again, set the command input to off once, then on again.
- These instructions can be used up to twice in any program.
- When specifying the same device in (s) and (d), the source data is overwritten by the data acquired by sorting. Take special care so that the contents of (s) are not changed until execution is completed.
- Ensure that the sorted data does not overlap with the source data.



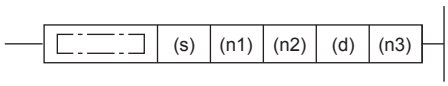

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s) exceeds the corresponding device range. |
| | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n1) is outside the following range.<br>1 to 32 |
| | The value specified by (n2) is outside the following range.<br>1 to 6 |
| | The value specified by (n3) is outside the following range.<br>1 to (n2) |

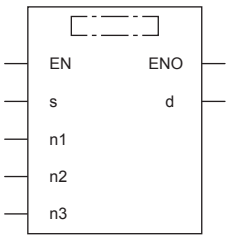
# 32-bit data alignment 2

## DSORTTBL2(_U)

These instructions sort data lines in the data table (sorting source) of 32-bit binary data having (n1×n2) points specified by (s) in the ascending order or descending order based on the group data in the column number (n3), and store the result in the data table (sorting result) of 32-bit binary data having ((n1)×(n2)) points specified by (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSORTTBL2(EN,s,n1,n2,n3,d);<br>ENO:= DSORTTBL2_U(EN,s,n1,n2,n3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DSORTTBL2 | Head device number storing the data table | — | 32-bit signed binary | ANY32 |
| | DSORTTBL2_U | | | 32-bit unsigned binary | ANY32_U |
| (n1) | | Number of data (lines) | 1 to 32 | 16-bit unsigned binary | ANY16_U |
| (n2) | | Number of group data (columns) | 1 to 6 | 16-bit unsigned binary | ANY16_U |
| (d) | DSORTTBL2 | Head device number for storing the operation result | — | 32-bit signed binary | ANY32 |
| | DSORTTBL2_U | | | 32-bit unsigned binary | ANY32_U |
| (n3) | | Column number of group data (column) used as the basis of sorting | — | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | — | ○ | — | ○ | — | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions sort data lines in the data table (sorting source) of 32-bit binary data having (n1×n2) points specified by (s) in the ascending order or descending order based on the group data in the column number (n3), and store the result in the data table (sorting result) of 32-bit binary data having ((n1)×(n2)) points specified by (d).

- The data table configuration is explained in an example in which the sorting source data table has 3 lines and 4 columns (n1 = K3, n2 = K4). For the sorting result data table, understand (s) as (d).

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 3 | Line No. 1 | (s)+1, (s) | (s)+3, (s)+2 | (s)+5, (s)+4 | (s)+7, (s)+6 |
| | Line No. 2 | (s)+9, (s)+8 | (s)+11, (s)+10 | (s)+13, (s)+12 | (s)+15, (s)+14 |
| | Line No. 3 | (s)+17, (s)+16 | (s)+19, (s)+18 | (s)+21, (s)+20 | (s)+23, (s)+22 |

- Set the sorting order by setting SM703 to on or off.

| | Sorting order |
|---|---|
| SM703 = ON | Descending order |
| SM703 = OFF | Ascending order |

- When the command input turns on, data sorting is started. Data sorting is completed after (n1) scans, and the instruction execution complete flag SM8029 is set to on.
- The following table shows an operation example based on the sorting source data below. It is recommended to put a serial number such as a control number in the first column so that the original line number can be estimated based on the contents.

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (s)+1, (s) | (s)+3, (s)+2 | (s)+5, (s)+4 | (s)+7, (s)+6 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 2 | (s)+9, (s)+8 | (s)+11, (s)+10 | (s)+13, (s)+12 | (s)+15, (s)+14 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 3 | (s)+17, (s)+16 | (s)+19, (s)+18 | (s)+21, (s)+20 | (s)+23, (s)+22 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 4 | (s)+25, (s)+24 | (s)+27, (s)+26 | (s)+29, (s)+28 | (s)+31, (s)+30 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 5 | (s)+33, (s)+32 | (s)+35, (s)+34 | (s)+37, (s)+36 | (s)+39, (s)+38 |
| | | 5 | 150 | 50 | 45 |

- Sorting result when the instructions are executed with (n3) = K2 (column No. 2) (in the case of ascending order SM703=OFF)

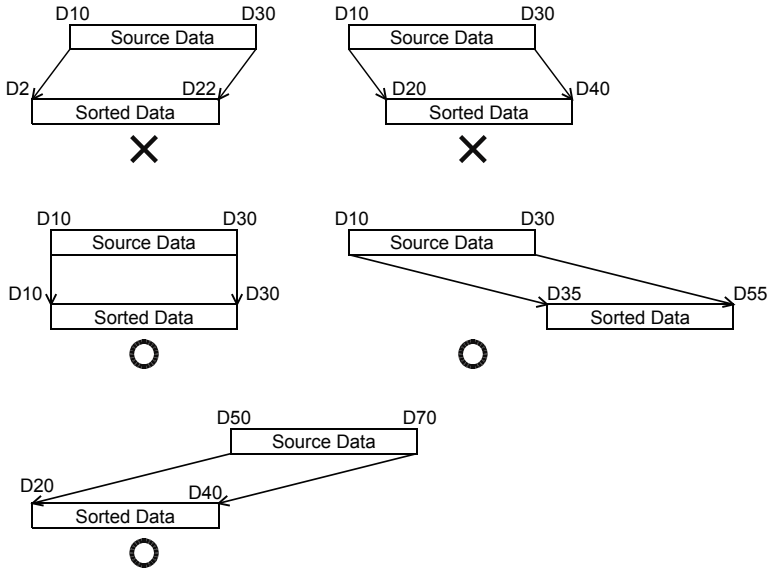
| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d)+1, (d) | (d)+3, (d)+2 | (d)+5, (d)+4 | (d)+7, (d)+6 |
| | | 4 | 100 | 20 | 8 |
| | Line No. 2 | (d)+9, (d)+8 | (d)+11, (d)+10 | (d)+13, (d)+12 | (d)+15, (d)+14 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 3 | (d)+17, (d)+16 | (d)+19, (d)+18 | (d)+21, (d)+20 | (d)+23, (d)+22 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 4 | (d)+25, (d)+24 | (d)+27, (d)+26 | (d)+29, (d)+28 | (d)+31, (d)+30 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 5 | (d)+33, (d)+32 | (d)+35, (d)+34 | (d)+37, (d)+36 | (d)+39, (d)+38 |
| | | 2 | 180 | 50 | 40 |

- Sorting result when the instructions are executed with (n3) = K3 (column No. 3) (in the case of descending order SM703=ON)

| | | Number of groups (n2 = K4) | | | |
|---|---|---|---|---|---|
| | | Column No. 1 | Column No. 2 | Column No. 3 | Column No. 4 |
| | | Control number | Height | Weight | Age |
| Number of data (n1) = 5 | Line No. 1 | (d)+1, (d) | (d)+3, (d)+2 | (d)+5, (d)+4 | (d)+7, (d)+6 |
| | | 3 | 160 | 70 | 30 |
| | Line No. 2 | (d)+9, (d)+8 | (d)+11, (d)+10 | (d)+13, (d)+12 | (d)+15, (d)+14 |
| | | 2 | 180 | 50 | 40 |
| | Line No. 3 | (d)+17, (d)+16 | (d)+19, (d)+18 | (d)+21, (d)+20 | (d)+23, (d)+22 |
| | | 5 | 150 | 50 | 45 |
| | Line No. 4 | (d)+25, (d)+24 | (d)+27, (d)+26 | (d)+29, (d)+28 | (d)+31, (d)+30 |
| | | 1 | 150 | 45 | 20 |
| | Line No. 5 | (d)+33, (d)+32 | (d)+35, (d)+34 | (d)+37, (d)+36 | (d)+39, (d)+38 |
| | | 4 | 100 | 20 | 8 |

## Precautions

- Do not change the contents of operands and data during operation.
- To execute these instructions again, set the command input to off once, then on again.
- These instructions can be used up to or twice in any program.
- When specifying the same device in (s) and (d), the source data is overwritten by the data acquired by sorting. Take special care so that the contents of (s) are not changed until execution is completed.
- Ensure that the sorted data does not overlap with the source data.



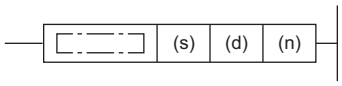

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s) exceeds the corresponding device range. |
| | The device range specified by (d) exceeds the corresponding device range. |
| 3405H | The value specified by (n1) is outside the following range.<br>1 to 32 |
| | The value specified by (n2) is outside the following range.<br>1 to 6 |
| | The value specified by (n3) is outside the following range.<br>1 to (n2) |

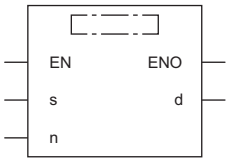
# Adding 16-bit data

## WSUM(P)(_U)

These instructions add the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=WSUM(EN,s,n,d);<br>ENO:=WSUMP(EN,s,n,d); | ENO:=WSUM_U(EN,s,n,d);<br>ENO:=WSUMP_U(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

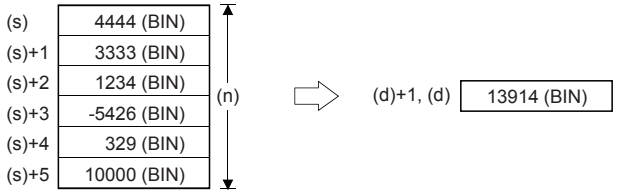
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | WSUM(P) | Head device number where the addition target data are stored | — | 16-bit signed binary | ANY16_S |
| | WSUM(P)_U | | | 16-bit unsigned binary | ANY16_U |
| (d) | WSUM(P) | Head device number storing sum | — | 32-bit signed binary | ANY32_S |
| | WSUM(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (n) | | Number of data | — | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions add the (n) point(s) of 16-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).



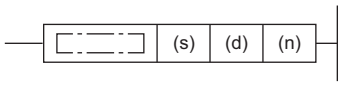

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| | The (n) point(s) of data in the device starting from (s) exceed the corresponding device range. |
| 3405H | The data stored in a device specified by (n) is 0. |

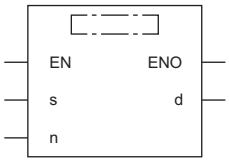
# Adding 32-bit data

## DWSUM(P)(_U)

These instructions add the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).

| Ladder diagram | Structured text | Structured text |
|---|---|---|
|  | ENO:=DWSUM(EN,s,n,d);<br>ENO:=DWSUMP(EN,s,n,d); | ENO:=DWSUM_U(EN,s,n,d);<br>ENO:=DWSUMP_U(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

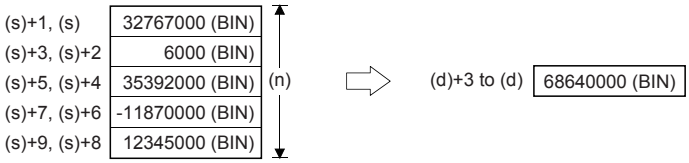
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DWSUM(P) | Head device number where the addition target data are stored | — | 32-bit signed binary | ANY32_S |
| | DWSUM(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (d) | DWSUM(P) | Head device number storing sum | — | 64-bit signed binary | ANY32_ARRAY |
| | DWSUM(P)_U | | | 64-bit unsigned binary | |
| (n) | | Number of data | — | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions add the (n) point(s) of 32-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).




### Precautions

In the 32-bit operation, the acquired sum is 64-bit data. The FX5 CPU module cannot handle 64-bit data. When the sum is within the numeric range of 32-bit data (K-2147483648 to K2147483647), however, the FX5 CPU module can handle the low-order 32 bits of 32-bit data as the sum while ignoring the high-order 32 bits.

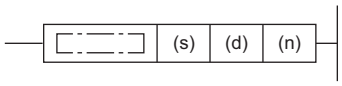
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |
| | The (n) point(s) of data in the device starting from (s) exceed the corresponding device range. |
| 3405H | The data stored in a device specified by (n) is 0. |

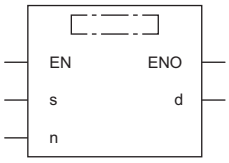
# Calculating the mean value of 16-bit data

## MEAN(P)(_U)

These instructions calculate the mean value of the (n) point(s) of 16-bit data units starting from the one specified by (s), and store the operation result in (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=MEAN(EN,s,n,d);<br>ENO:=MEANP(EN,s,n,d); | ENO:=MEAN_U(EN,s,n,d);<br>ENO:=MEANP_U(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

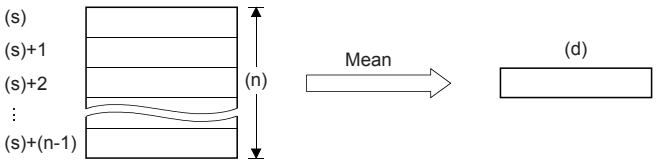
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | MEAN(P) | Head device number where the mean value target data are stored | — | 16-bit signed binary | ANY16_S |
| | MEAN(P)_U | | | 16-bit unsigned binary | ANY16_U |
| (d) | MEAN(P) | Head device number storing mean value | — | 16-bit signed binary | ANY16_S |
| | MEAN(P)_U | | | 16-bit unsigned binary | ANY16_U |
| (n) | | Number of data or the device number storing the number of data | 1 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions calculate the mean value of the (n) point(s) of 16-bit data starting from the one specified by (s), and store the operation result in a device specified by (d).




- The sum is obtained as algebraic sum, and divided by (n).
- The remainder is ignored.

### Precautions

When a device number is exceeded, (n) is handled as a smaller value in the possible range.

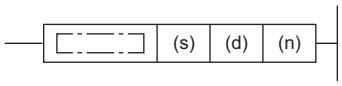
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The value stored in a device specified by (n) is 0. |

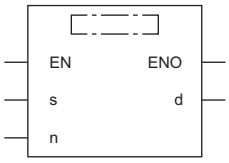
# Calculating the mean value of 32-bit data

## DMEAN(P)(_U)

These instructions calculate the mean value of the (n) point(s) of 32-bit data units starting from the one specified by (s), and store the operation result in (d).

| Ladder diagram | Structured text | |
|---|---|---|
|  | ENO:=DMEAN(EN,s,n,d);<br>ENO:=DMEANP(EN,s,n,d); | ENO:=DMEAN_U(EN,s,n,d);<br>ENO:=DMEANP_U(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

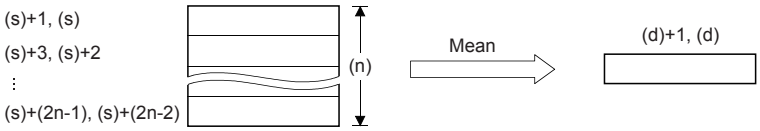
| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s) | DMEAN(P) | Head device number where the mean value target data are stored | — | 32-bit signed binary | ANY32_S |
| | DMEAN(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (d) | DMEAN(P) | Head device number storing mean value | — | 32-bit signed binary | ANY32_S |
| | DMEAN(P)_U | | | 32-bit unsigned binary | ANY32_U |
| (n) | | Number of data or the device number storing the number of data | 1 to 65535 | 16-bit unsigned binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | ○ | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

- These instructions calculate the mean value of the (n) point(s) of 32-bit data starting from the one specified by (s), and store the operation result in a device specified by (d).




- The sum is obtained as algebraic sum, and divided by (n).
- The remainder is ignored.

### Precautions

When a device number is exceeded, (n) is handled as a smaller value in the possible range.

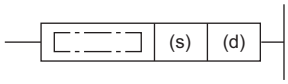
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The value stored in a device specified by (n) is 0. |

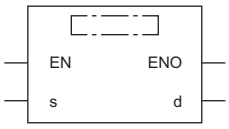
# Calculating the square root of 16-bit data

## SQRT(P)

These instructions calculate the square root of binary 16-bit data specified by (s1), and store the operation result in (d).

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=SQRTP(EN,s,d); |

**FBD/LD*1**




*1 The SQRT instruction is not supported by the ST language and the FBD/LD language. Use SQRT of the standard function.
☞ Page 861 SQRT(_E)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device where the data whose square root is operated is calculated | — | 16-bit signed binary | ANY16 |
| (d) | Device for storing the calculated square root | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

### Processing details

- These instructions calculate the square root of binary 16-bit data specified by (s1), and store the operation result in (d).

$\sqrt{(s)} \rightarrow (d)$

### Precautions

- The obtained square root is an integer because the decimal point is ignored. When the calculated decimal value is ignored, SM8021 (borrow flag) turns on.
- When the operation result is true 0, SM8020 (zero flag) turns on.

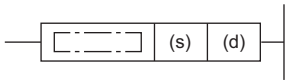
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | In (s), a negative value is specified. |

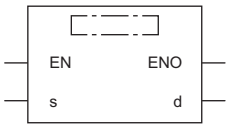
# Calculating the square root of 32-bit data

## DSQRT(P)

These instructions calculate the square root of binary 32-bit data specified by (s1), and store the operation result in (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSQRT(EN,s,d);<br>ENO:=DSQRTP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device where the data whose square root is operated is calculated | — | 32-bit signed binary | ANY32 |
| (d) | Device for storing the calculated square root | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

### Processing details

- These instructions calculate the square root of binary 32-bit data specified by (s1), and store the operation result in (d).

$\sqrt{(s)+1, (s)} \rightarrow (d)+1, (d)$

### Precautions

- The obtained square root is an integer because the decimal point is ignored. When the calculated decimal value is ignored, SM8021 (borrow flag) turns on.
- When the operation result is true 0, SM8020 (zero flag) turns on.

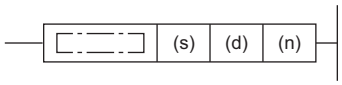
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | In (s), a negative value is specified. |

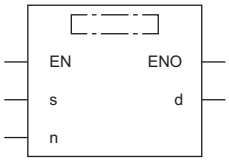
# CRC calculation

## CRC(P)

These instructions calculate the CRC (cyclic redundancy check) value which is an error check method used in communication. In addition to CRC value, parity check and sum check are available. For obtaining the horizontal parity value and sum check value, the CCD(P) instruction is available. For the generation of CRC value (CRC-16), these instructions use "$X^{16} + X^{15} + X^2 + 1$" in a polynomial.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=CRC(EN,s,n,d);<br>ENO:=CRCP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number storing data for which the CRC value is generated | — | 16-bit signed binary | ANY16 |
| (d) | Device number storing the generated CRC value | — | 16-bit signed binary | ANY16 |
| (n) | Number of 8-bit (1-byte) data for which the CRC value is generated or the device number storing the number of data | 1 to 32767 | 16-bit unsigned binary | ANY16_U |

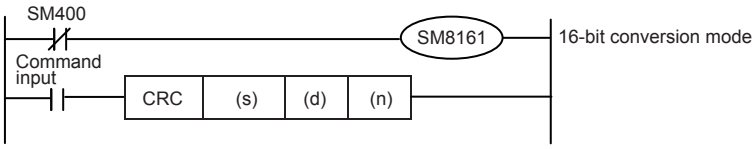
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- These instructions generate CRC value for (n) 8-bit data (unit: byte) starting from a device specified in (s), and store to (d). The 16-bit conversion mode and 8-bit conversion mode are available for these instructions. For the operation in each mode, refer to the proceeding pages.
  - 16-bit conversion mode (while SM8161 is OFF)

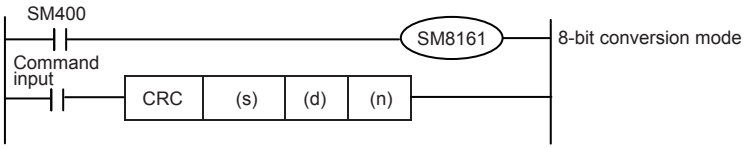
    In this mode, the operation is executed for high-order 8 bits (1 byte) and low-order 8 bits (1 byte) of a device specified in (s). The operation result is stored to one 16-bit device specified in (d).
    In the following program, conversion is executed as follows:




| | | | Example) (s) = D100, (d) = D0, (n) = 6 | | |
|---|---|---|---|---|---|
| | | | Device | Contents of target data | |
| | | | | 8 bits | 16 bits |
| Device storing data for which the CRC value is generated | (s) | Low-order byte | Low-order bits of D100 | 01H | 0301H |
| | | High-order byte | High-order bits of D100 | 03H | |
| | (s)+1 | Low-order byte | Low-order bits of D101 | 03H | 0203H |
| | | High-order byte | High-order bits of D101 | 02H | |
| | (s)+2 | Low-order byte | Low-order bits of D102 | 00H | 1400H |
| | | High-order byte | High-order bits of D102 | 14H | |
| | ⋮ | ⋮ | — | | |
| | (s)+(n)/2-1 | Low-order byte | — | | |
| | | High-order byte | | | |
| Device storing the generated CRC value | (d) | Low-order byte | Low-order bits of D0 | E4H | 41E4H |
| | | High-order byte | High-order bits of D0 | 41H | |

  - 8-bit conversion mode (while SM8161 is ON)

    In this mode, the operation is executed only for low-order 8 bits (low-order 1 byte) of a device specified by (s). With regard to the operation result, low-order 8 bits (1 byte) are stored to a device specified by (d), and high-order 8 bits (1 byte) are stored to a device specified by (d)+1.
    In the following program, conversion is executed as follows:




| | | | Example) (s) = D100, (d) = D0, (n) = 6 | |
|---|---|---|---|---|
| | | | Device | Contents of target data |
| Device storing data for which the CRC value is generated | (s) | Low-order byte | Low-order bits of D100 | 01H |
| | (s)+1 | Low-order byte | Low-order bits of D101 | 03H |
| | (s)+2 | Low-order byte | Low-order bits of D102 | 03H |
| | (s)+3 | Low-order byte | Low-order bits of D103 | 02H |
| | (s)+4 | Low-order byte | Low-order bits of D104 | 00H |
| | (s)+5 | Low-order byte | Low-order bits of D105 | 14H |
| | ⋮ | | — | |
| | (s)+(n)-1 | Low-order byte | — | |
| Device storing the generated CRC value | (d) | Low-order byte | Low-order bits of D0 | E4H |
| | (d)+1 | High-order byte | High-order bits of D0 | 41H |

## Precautions

- In these instructions, "$X^{16}+X^{15}+X^{2}+1$" is used in a polynomial for generating the CRC value (CRC-16). There are many other standard polynomials for generating the CRC value. Note that the CRC value completely differs if an adopted polynomial is different. Major polynomials for generating the CRC value are shown below.

| Name | Polynomial |
|---|---|
| CRC-12 | $X^{12} + X^{11} + X^{3} + X^{2} + X + 1$ |
| CRC-16 | $X^{16} + X^{15} + X^{2} + 1$ |
| CRC-32 | $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$ |
| CRC-CCITT | $X^{16} + X^{12} + X^{5} + 1$ |

## Operation error

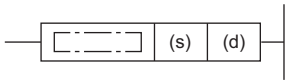
There is no operation error.

# 8.21 Indirect Address Read Instruction

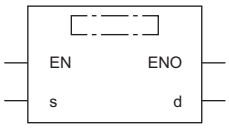
## Reading the indirect address

### ADRSET(P)

These instructions store the indirect address of the device specified by (s) to the device specified by (d).
The addresses stored in the device specified by (d)+0 and (d)+1 are used by the program to execute the indirect address of the device.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ADRSET(EN,s,d);<br>ENO:=ADRSETP(EN,s,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Device number for reading the indirect address | — | Device name | ANY_ELEMENTARY |
| (d) | Device number for storing the indirect address of the device specified by (s) | — | 32-bit signed binary | ANY32 |

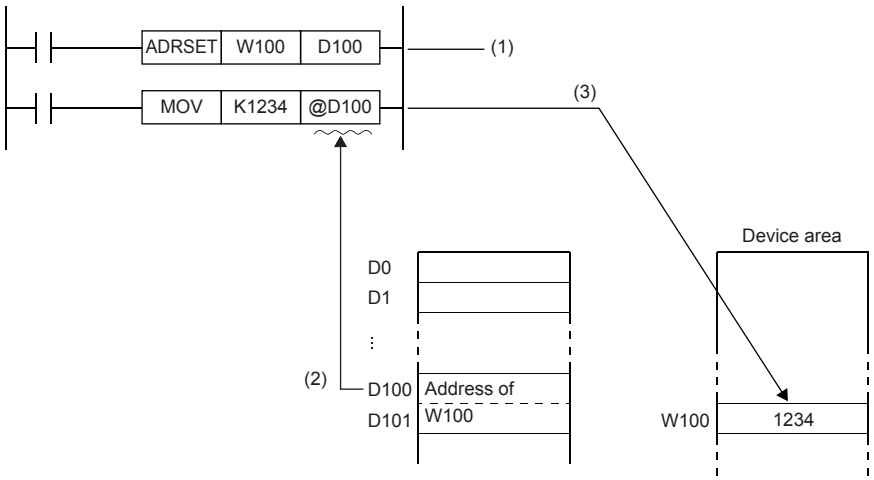
**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

*1 T, ST, C cannot be used.

#### Processing details

- These instructions store the indirect address of the device specified by (s) to the device specified by (d). The addresses stored in the device specified by (d)+0 and (d)+1 are used by the program to execute the indirect address of the device.

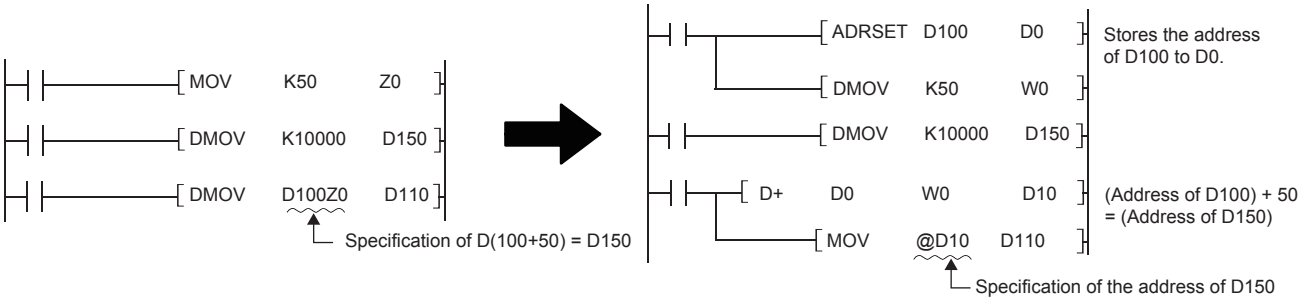



(1) Stores the address of W100 to D100 and D101.
(2) Specifies the content of D100 and D101 (address of W100).
(3) Writes "1234" to W100.

- The nibble of a bit device, and the bit of a word device cannot be specified in (s).

## Precautions

- In the indirect specification, the device address used in sequence program is specified with a word device of 2 words (2-word devices). Use the indirect specification as an index when index register is insufficient.




[When the index register is used] [When the indirect specification is used]

- In the indirect specification, the device which specify the address of the specified device is specified by "@+(word device number)". For example, when "@D100" is specified, and the content of D101 and D100 becomes the device address.

## Operation error

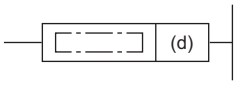
There is no operation error.

# 8.22 Clock Instruction

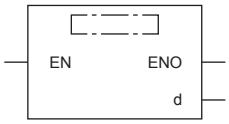
## Reading clock data

### TRD(P)

These instructions read the clock data from the built-in real time clock in the CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TRD(EN,d);<br>ENO:=TRDP(EN,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Head device number where the read clock data is stored | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 7) |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

#### Processing details

- These instructions read the clock data (SD210 to SD216) from the built-in real time clock in the CPU module to the device numbers (d) to (d)+6 in the following format.

| | Device | Item | Clock data | | Device | Item |
|---|---|---|---|---|---|---|
| Special registers | SD210 | Year | 1980 to 2079 (year, four digits) | → | (d) | Year |
| | SD211 | Month | 1 to 12 | → | (d)+1 | Month |
| | SD212 | Day | 1 to 31 | → | (d)+2 | Day |
| | SD213 | Hour data | 0 to 23 | → | (d)+3 | Hour data |
| | SD214 | Minute data | 0 to 59 | → | (d)+4 | Minute data |
| | SD215 | Second data | 0 to 59 | → | (d)+5 | Second data |
| | SD216 | Day-of-the-week data | 0 (Sunday) to 6 (Saturday) | → | (d)+6 | Day-of-the-week data |

- The table below shows the related devices. The clock data stored in these special registers is updated during the END processing.

| Device | Name | Description |
|---|---|---|
| Binary code | | |
| SD210 | Binary clock data (year) | The year data in the clock data is stored as a four-digit binary code. |
| SD211 | Binary clock data (month) | The month data in the clock data is stored as a binary code. |
| SD212 | Binary clock data (day) | The day data in the clock data is stored as a binary code. |
| SD213 | Binary clock data (hour) | The hour data in the clock data is stored as a binary code. |
| SD214 | Binary clock data (minute) | The minute data in the clock data is stored as a binary code. |
| SD215 | Binary clock data (second) | The second data in the clock data is stored as a binary code. |
| SD216 | Binary clock data (day of the week) | The day-of-a-week data in the clock data (0: Sunday, 1: Monday, ..., 6: Saturday) is stored as a binary code. |
| Binary code (FX3 compatible area) | | |
| SD8013 | Binary clock data (second) | The second data in the clock data is stored as a binary code. |
| SD8014 | Binary clock data (minute) | The minute data in the clock data is stored as a binary code. |
| SD8015 | Binary clock data (hour) | The hour data in the clock data is stored as a binary code. |
| SD8016 | Binary clock data (day) | The day data in the clock data is stored as a binary code. |
| SD8017 | Binary clock data (month) | The month data in the clock data is stored as a binary code. |
| SD8018 | Binary clock data (year) | The year data in the clock data is stored as a four-digit binary code. |
| SD8019 | Binary clock data (day of the week) | The day-of-a-week data in the clock data (0: Sunday, 1: Monday, ..., 6: Saturday) is stored as a binary code. |

## Precautions

- These instructions occupy seven points of device starting from device number specified by (d). Make sure that these devices are not used by other machine controls.

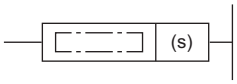
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (d) exceeds the corresponding device range. |

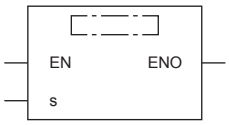
# Writing clock data

## TWR(P)

This instruction writes the clock data to the built-in CPU module real time clock.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TWR(EN,s);<br>ENO:=TWRP(EN,s); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the clock write source data is stored | — | 16-bit signed binary | ANY16_ARRAY<br>(Number of elements: 7) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

### Processing details

- These instructions write the clock data stored in device numbers (s) to (s)+6 to the clock data area (SD210 to SD216 and SD8013 to SD8019) of the built-in real time clock in the CPU module.

| Time setting data | | | | Special registers | |
|---|---|---|---|---|---|
| Device | Item | Clock data | | Device | Item |
| (s) | Year | 1980 to 2079 (year, four digits) | → | SD210, SD8018 | Year |
| (s)+1 | Month | 1 to 12 | → | SD211, SD8017 | Month |
| (s)+2 | Day | 1 to 31 | → | SD212, SD8016 | Day |
| (s)+3 | Hour data | 0 to 23 | → | SD213, SD8015 | Hour data |
| (s)+4 | Minute data | 0 to 59 | → | SD214, SD8014 | Minute data |
| (s)+5 | Second data | 0 to 59 | → | SD215, SD8013 | Second data |
| (s)+6 | Day-of-the-week data | 0 (Sunday) to 6 (Saturday) | → | SD216, SD8019 | Day-of-the-week data |

- Executing these instructions immediately changes the real time clock data. Therefore, transfer the clock data of a few minutes ahead the current time to the clock data area (s) to (s)+6 in advance. Execute the instruction when the actual time matches the clock data time.
- When using these instructions to set the clock data (i.e., performing time adjustment), control of special relay SM8015 (clock stop/adjustment) is not required.
- If incorrect values (i.e., values out of range) are set to the write source area, the clock data will not be updated. In this case, correct the clock data in the write source area and execute the instruction.
- Day of the week (SD216 and SD8019) is automatically corrected.

• The table below shows the related devices.

| Device | Name | Description |
|---|---|---|
| SM8019 | Real time clock error | This special data register turns on when the clock data value in the special register is exceeding the setting range. |
| Binary code | | |
| SD210 | Binary clock data (year) | The year data in the clock data is stored as a four-digit binary code. |
| SD211 | Binary clock data (month) | The month data in the clock data is stored as a binary code. |
| SD212 | Binary clock data (day) | The day data in the clock data is stored as a binary code. |
| SD213 | Binary clock data (hour) | The hour data in the clock data is stored as a binary code. |
| SD214 | Binary clock data (minute) | The minute data in the clock data is stored as a binary code. |
| SD215 | Binary clock data (second) | The second data in the clock data is stored as a binary code. |
| SD216 | Binary clock data (day of the week) | The day-of-a-week data in the clock data (0: Sunday, 1: Monday, ..., 6: Saturday) is stored as a binary code. |
| Binary code (FX3 compatible area) | | |
| SD8013 | Binary clock data (second) | The second data in the clock data is stored as a binary code. |
| SD8014 | Binary clock data (minute) | The minute data in the clock data is stored as a binary code. |
| SD8015 | Binary clock data (hour) | The hour data in the clock data is stored as a binary code. |
| SD8016 | Binary clock data (day) | The day data in the clock data is stored as a binary code. |
| SD8017 | Binary clock data (month) | The month data in the clock data is stored as a binary code. |
| SD8018 | Binary clock data (year) | The year data in the clock data is stored as a four-digit binary code. |
| SD8019 | Binary clock data (day of the week) | The day-of-a-week data in the clock data (0: Sunday, 1: Monday, ..., 6: Saturday) is stored as a binary code. |

## Precautions

• These instructions occupy seven points of device starting from device number specified by (s). Make sure that these devices are not used by other machine controls.

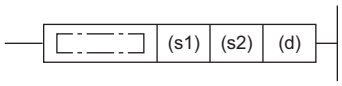
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s) exceeds the corresponding device range. |

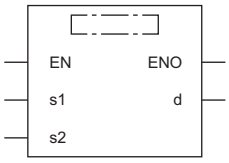
# Adding clock data

## TADD(P)

These instructions add the time data stored in the device number specified by (s2) and later to the clock data stored in the device number specified by (s1) and later, and store the result to the device number specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TADD(EN,s1,s2,d);<br>ENO:=TADDP(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number where the clock data to be added is stored. | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (s2) | Head device number where the adding time value (or clock data value) is stored. | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Head device number where the resultant clock data (or time value) is stored. | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

### Processing details

- These instructions add the time data stored in the device numbers starting from (s2) to the clock data stored in the device numbers starting from (s1), and store the result to the device numbers starting from (d).

| | | Data range | | | | Data range | | | | Data range |
|---|---|---|---|---|---|---|---|---|---|---|
| (s1) | Hour | (0 to 23) | | (s2) | Hour | (0 to 23) | | (d) | Hour | (0 to 23) |
| (s1)+1 | Minute | (0 to 59) | + | (s2)+1 | Minute | (0 to 59) | ⇨ | (d)+1 | Minute | (0 to 59) |
| (s1)+2 | Second | (0 to 59) | | (s2)+2 | Second | (0 to 59) | | (d)+2 | Second | (0 to 59) |

Ex.

When adding 7:48:10 to 6:32:40

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (s1) | 6 | | (s2) | 7 | | (d) | 14 |
| (s1)+1 | 32 | + | (s2)+1 | 48 | ⇨ | (d)+1 | 20 |
| (s1)+2 | 40 | | (s2)+2 | 10 | | (d)+2 | 50 |

- If the sum of two values exceeds 24:00:00, the carry flag turns on, and the result will be the sum minus 24:00:00. For example, if a time value of 20:20:20 is added to another time value of 14:30:30, the sum is 34:40:50. However, the actual addition result will be 10:40:50.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (s1) | 14 | | (s2) | 20 | | (d) | 10 |
| (s1)+1 | 20 | + | (s2)+1 | 20 | ⇨ | (d)+1 | 40 |
| (s1)+2 | 30 | | (s2)+2 | 20 | | (d)+2 | 50 |

- If the result is 0 (0:00:00), the zero flag turns on.
- If 1 second is added to 23:59:59, the result will be 0:00:00. This turns on both the carry flag and the zero flag.
- The table below shows the related devices.

| Device | Name | Description |
|---|---|---|
| SM700 | Carry | If the result exceeds the maximum value of the time data, 23:59:59, this special relay turns on. |
| SM8020 | Zero | If the result is 0:00:00, this special relay turns on. |
| SM8022 | Carry | If the result exceeds the maximum value of the time data, 23:59:59, this special relay turns on. |

## Precautions

- These instructions occupy three points for each of three devices starting from device number specified by (s1), (s2), and (d) respectively. Make sure that these devices are not used by other machine controls.
- When using the time value (hour, minute, second) of the built-in real time clock in the CPU module for the operation, use the TRD(P) operation to read the special register values first. Then specify the word devices where the read values are stored to each operand.

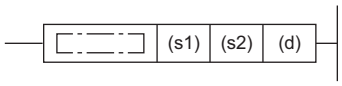
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s1), (s2), and (d) exceeds the corresponding device range. |
| 3405H | Any of values specified by (s1) and (s2) is outside the following range.<br>0 to 23 |
| | Any of values specified by (s1)+1, (s2)+1, (s1)+2, and (s2)+2 is outside the following range.<br>0 to 59 |

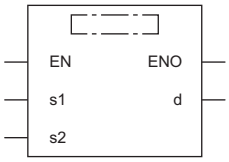
# Subtracting clock data

## TSUB(P)

These instructions subtract the time data stored in the device numbers starting from (s2) from the clock data stored in the device numbers starting from (s1), and store the result to the device numbers starting from (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TSUB(EN, s1, s2,d);<br>ENO:=TSUBP(EN, s1, s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

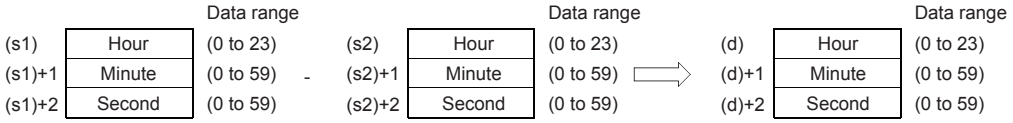
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number where the clock data that is subtracted is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (s2) | Head device number where the subtracting time value (or clock data value) is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Head device number where the resultant clock data (or time value) is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

### Processing details

- These instructions subtract the time data stored in the device numbers starting from (s2) from the clock data stored in the device numbers starting from (s1), and store the result to the device numbers starting from (d).



Ex.

When subtracting 3:50:10 from 10:40:20

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (s1) | 10 | | (s2) | 3 | | (d) | 6 |
| (s1)+1 | 40 | - | (s2)+1 | 50 | ⇨ | (d)+1 | 50 |
| (s1)+2 | 20 | | (s2)+2 | 10 | | (d)+2 | 10 |

- If the remainder is a negative time value, the borrow flag turns on. The actual result will be the remainder plus 24:00:00. For example, if a time value of 10:42:12 is subtracted from another time value of 4:50:32, the remainder is -6:08:20. However, the actual subtraction result will be 18:08:20.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (s1) | 4 | | (s2) | 10 | | (d) | 18 |
| (s1)+1 | 50 | - | (s2)+1 | 42 | ⇨ | (d)+1 | 8 |
| (s1)+2 | 32 | | (s2)+2 | 12 | | (d)+2 | 20 |

- If the result is 0 (0:00:00), the zero flag turns on.
- The table below shows the related devices.

| Device | Name | Description |
|---|---|---|
| SM8020 | Zero | If the result is 0:00:00, this special relay turns on. |
| SM8021 | Borrow | If the execution result of the TSUB(P) instruction is less than 0:00:00, this special relay turns on. |

## Precautions

- These instructions occupy three points for each of three devices starting from device number specified by (s1), (s2), and (d) respectively. Make sure that these devices are not used by other machine controls.
- When using the time value (hour, minute, second) of the built-in real time clock in the CPU module for the operation, use the TRD(P) operation to read the special register values first. Then specify the word devices where the read values are stored to each operand.

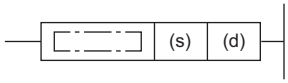
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified by (s1), (s2), and (d) exceeds the corresponding device range. |
| 3405H | Any of values specified by (s1) and (s2) is outside the following range.<br>0 to 23 |
| | Any of values specified by (s1)+1, (s2)+1, (s1)+2, and (s2)+2 is outside the following range.<br>0 to 59 |

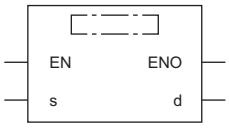
# Converting time data from hour/minute/second to seconds in 16 bits

## HTOS(P)

These instructions convert the time data stored in the device numbers starting from (s) to the time value in seconds, and store the converted data in the device numbers starting from (d) as 16-bit binary.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=HTOS(EN,s,d);<br>ENO:=HTOSP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

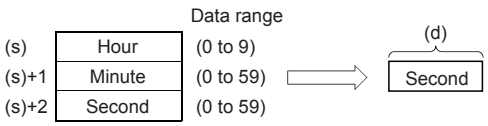
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the clock data before conversion is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Head device number where the clock data after conversion is stored | — | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |

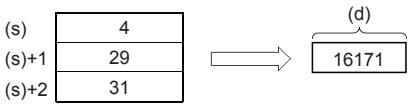
### Processing details

- These instructions convert the time data stored in the device numbers starting from (s) to the time value in seconds, and store the converted data in the device numbers starting from (d).




**Ex.**

When specifying 4 hours 29 minutes 31 seconds in (s)



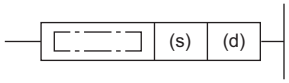
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Any of the device area ranges specified in (s) and (d) exceed the corresponding device range. |
| 3403H | The result is outside the following range.<br>0 to 32767 |
| 3405H | A value specified by (s) is outside the following range.<br>0 to 9 |
| | Any of values specified by (s)+1 and (s)+2 is outside the following range.<br>0 to 59 |

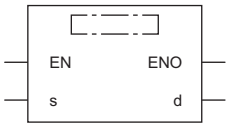
# Converting time data from hour/minute/second to seconds in 32 bits

## DHTOS(P)

These instructions convert the time data stored in the device numbers starting from (s) to the time value in seconds, and store the converted data in the device numbers starting from (d) as 32-bit binary.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHTOS(EN,s,d);<br>ENO:=DHTOSP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

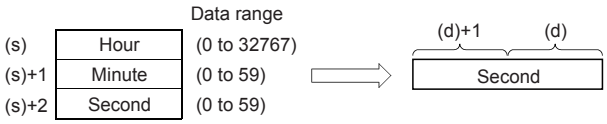
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the clock data before conversion is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Head device number where the clock data after conversion is stored | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |

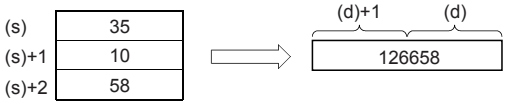
### Processing details

- These instructions convert the time data stored in the device numbers starting from (s) to the time value in seconds, and store the converted data in the device numbers starting from (d).




Ex.

When specifying 35 hours 10 minutes 58 seconds in (s)



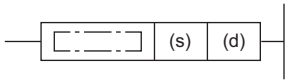
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | Any of the device area ranges specified in (s) and (d) exceed the corresponding device range. |
| 3405H | A value specified by (s) is outside the following range.<br>0 to 32767 |
| | Any of values specified by (s)+1 and (s)+2 is outside the following range.<br>0 to 59 |

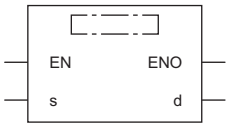
# Converting time data from seconds to hour/minute/second in 16 bits

## STOH(P)

These instructions convert the 16-bit time value in seconds stored in the device numbers starting from (s) to the time value in the HHMMDD format, and store the converted data in the device numbers starting from (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=STOH(EN,s,d);<br>ENO:=STOHP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

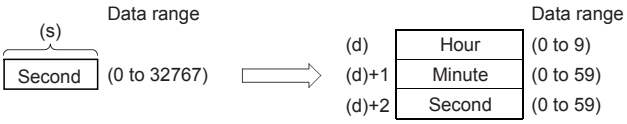
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the clock data before conversion is stored | — | 16-bit signed binary | ANY16 |
| (d) | Head device number where the clock data after conversion is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

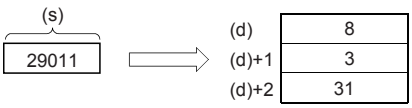
### Processing details

- These instructions convert the time value in seconds stored in the device numbers starting from (s) to the time value in HHMMDD format, and store the converted data in the device numbers starting from (d).




Ex.

When specifying 29011 seconds in (s)



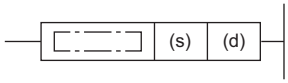

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The specified device area exceeds the corresponding device range. |
| 3405H | The value specified by (s) is outside the range. |

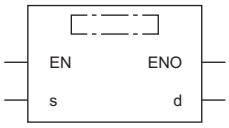
# Converting time data from seconds to hour/minute/second in 32 bits

## DSTOH(P)

These instructions convert the 32-bit time value in seconds stored in the device numbers starting from (s) to the time value in the HHMMDD format, and store the converted data in the device numbers starting from (d).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSTOH(EN,s,d);<br>ENO:=DSTOHP(EN,s,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

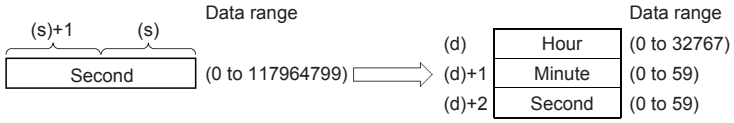
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number where the clock data before conversion is stored | — | 32-bit signed binary | ANY32 |
| (d) | Head device number where the clock data after conversion is stored | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |

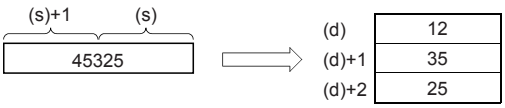
### Processing details

- These instructions convert the time value in seconds stored in the device numbers starting from (s) to the time value in HHMMDD format, and store the converted data in the device numbers starting from (d).




Ex.

When specifying 45325 seconds in (s)



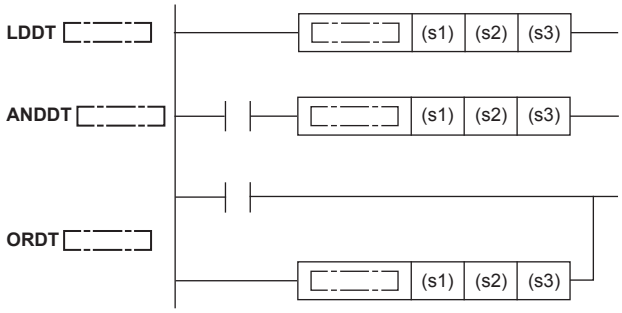

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The specified device area exceeds the corresponding device range. |
| 3405H | The value specified by (s) is outside the range. |

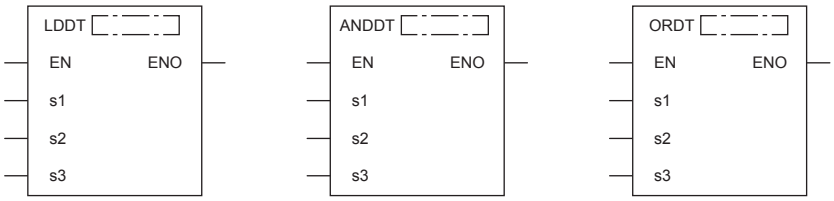
# Comparing date data

## LDDT□, ANDDT□, ORDT□

These instructions compare the date data in the devices specified by (s1) and (s2). Or, these instructions compare the date data in the device specified by (s1) with the current date.
Set the comparison target by (s3).

| Ladder diagram | Structured text |
|---|---|
|  ("=", "<>", ">", "<=", "<", ">=" enters □.) | Not supported |

**FBD/LD**




("_EQ", "_NE", "_GT", "_LE", "_LT", "_GE" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number where the comparison data is stored | — | 16-bit signed binary | ANY_DT |
| (s2) | Head device number where the comparison data is stored | — | 16-bit signed binary | ANY_DT |
| (s3) | Comparison target setting value or the number of comparison target data | 0001H to 0007H, 8001H to 8007H | 16-bit signed binary | ANY16 |

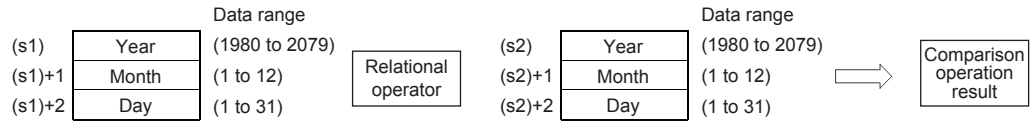
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s3) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

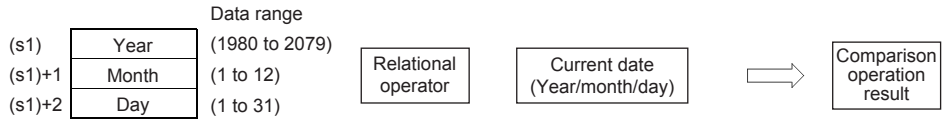
- These instructions compare the date data in the devices specified by (s1) and (s2), or compare the date data in the device specified by (s1) with the current date. Set the comparison target by (s3).
  - Comparing two specified date data

    These instructions compare the date data in the device specified by (s1) with the date data in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)

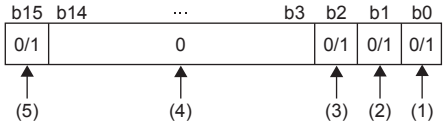



  - Comparing the specified date data with the current date

    These instructions compare the date data in the device specified by (s1) with the current date data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The date data in the device specified by (s2) is regarded as dummy data and ignored.




- Set each data in binary.
- Set the 4 digit "year" data in the devices specified by (s1) and (s2) within the range 1980 to 2079.
- Set the "month" data in the devices specified by (s1)+1 and (s2)+1 within the range 1 to 12.
- Set the "date" data in the devices specified by (s1)+2 and (s2)+2 within the range 1 to 31.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).




(1) Set "day" as comparison target.
(2) Set "month" as comparison target.
(3) Set "year" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current date in accordance with the conditions set in the 0 to 2 bits.

- When 0 is set to the 0 to 2 bits, the date data are not compared. When 1 is set, the entire date data (year, month, and day) are compared.
- When 0 is set to the 15 bit, the data in the device specified by (s1) and the date data in the device specified by (s2) are compared. When 1 is set, the data in the device specified by (s1) is compared with the current date. The date data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

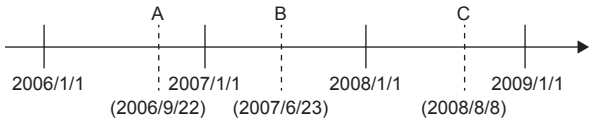
| (s3) value when comparing two specified date data | (s3) value when comparing the specified date data with the current date | Comparison target | Contents of processing |
|---|---|---|---|
| 0001H | 8001H | Day | Only data in the device specified by (s1)+2 is compared. |
| 0002H | 8002H | Month | Only data in the device specified by (s1)+1 is compared. |
| 0003H | 8003H | Month, day | Data in the device areas specified by (s1)+2 and (s1)+2 are compared. |
| 0004H | 8004H | Year | Only data in the device specified by (s1) is compared. |
| 0005H | 8005H | Year, day | Data in the device areas specified by (s1) and (s1)+2 are compared. |
| 0006H | 8006H | Year, month | Data in the device areas specified by (s1) and (s1)+1 are compared. |
| 0007H | 8007H | Year, month, day | The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared. |
| Other than 0001H to 0007H, 8001H to 8007H | | None | The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.) |

- If the comparison target data in the device are not recognized as date data, SM709 turns on after the instruction is executed and the operation result will be non-continuity. Even if the data are not recognized as date data, SM709 does not turn on if the data are within the setting range. If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the corresponding device range, SM709 turns on after the instruction is executed and the operation result will be non-continuity as well. Once SM709 turns on, the on state is held until the CPU module is powered off or reset. Turn off SM 709 as needed.
- The following table lists the comparison operation results of each instruction.

| Instruction symbol | Condition | Result | Instruction symbol | Condition | Result |
|---|---|---|---|---|---|
| DT= | (s1)=(s2) | Conductive state | DT= | (s1)≠(s2) | Non-conductive state |
| DT<> | (s1)≠(s2) | | DT<> | (s1)=(s2) | |
| DT> | (s1)>(s2) | | DT> | (s1)≤(s2) | |
| DT<= | (s1)≤(s2) | | DT<= | (s1)>(s2) | |
| DT< | (s1)<(s2) | | DT< | (s1)≥(s2) | |
| DT>= | (s1)≥(s2) | | DT>= | (s1)<(s2) | |

**Ex.**

The date data A, B, and C are compared.




- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

○: Continuity, ×: Non-continuity

| Comparison target data | Condition | | |
|---|---|---|---|
| | A<B | B<C | A<C |
| Day | ○ | × | × |
| Month | × | ○ | × |
| Month, day | × | ○ | × |
| Year | ○ | ○ | ○ |
| Year, day | ○ | ○ | ○ |
| Year, month | ○ | ○ | ○ |
| Year, month, day | ○ | ○ | ○ |
| None | × | × | × |

- Even though the specified date does not exist, the comparison operation is performed in accordance with the conditions in the following table as long as the date data are within the valid range.
  - Date A: 2006/02/30 (Even though the date does not exist, this date can be set.)
  - Date B: 2007/03/29
  - Date A: 2008/02/31 (Even though the date does not exist, this date can be set.)

○: Continuity, ×: Non-continuity

| Comparison target data | Condition | | |
|---|---|---|---|
| | A<B | B<C | A<C |
| Day | × | × | ○ |
| Month | × | × | × |
| Month, day | ○ | × | ○ |
| Year | ○ | ○ | ○ |
| Year, day | ○ | ○ | ○ |
| Year, month | ○ | ○ | ○ |
| Year, month, day | ○ | ○ | ○ |
| None | × | × | × |

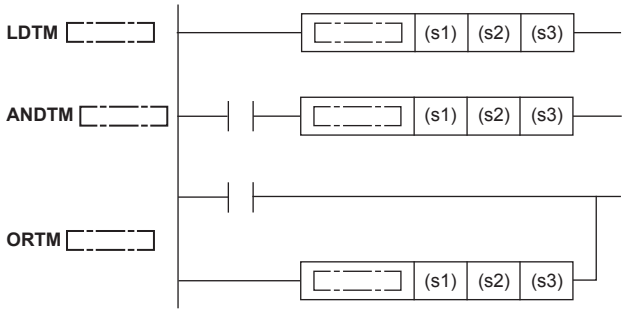
## Operation error

There is no operation error.

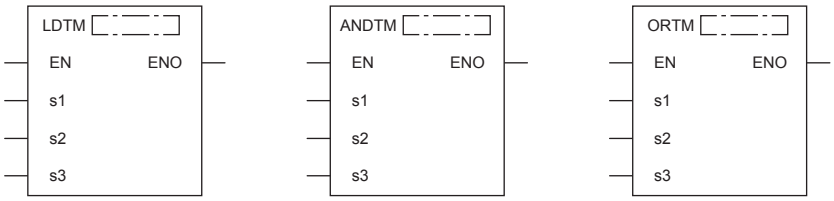
# Comparing time data

## LDTM□, ANDTM□, ORTM□

These instructions compare the time data in the devices specified by (s1) and (s2). Or, these instructions compare the time data in the device specified by (s1) with the current time.
Set the comparison target by (s3).

| Ladder diagram | Structured text |
|---|---|
|  ("=", "<>", ">", "<=", "<", ">=" enters □.) | Not supported |

**FBD/LD**




("_EQ", "_NE", "_GT", "_LE", "_LT", "_GE" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Head device number where the comparison data is stored | — | 16-bit signed binary | ANY_TM |
| (s2) | Head device number where the comparison data is stored | — | 16-bit signed binary | ANY_TM |
| (s3) | Comparison target setting value or the number of comparison target data | 0001H to 0007H, 8001H to 8007H | 16-bit signed binary | ANY16 |

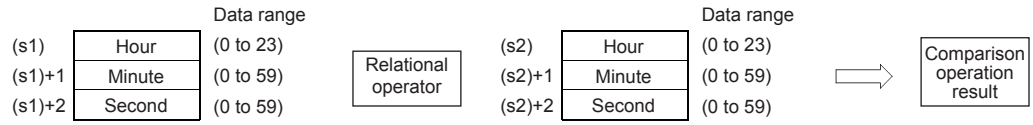
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s3) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

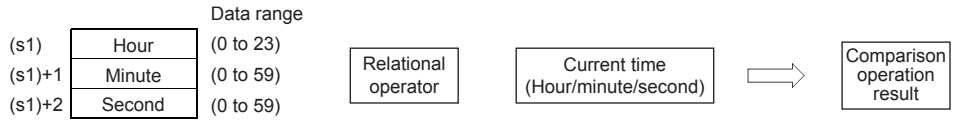
- These instructions compare the time data in the devices specified by (s1) and (s2), or compare the time data in the device specified by (s1) with the current time. Set the comparison target by (s3).
  - Comparing two specified time data

    These instructions compare the time data in the device specified by (s1) with the time data in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)

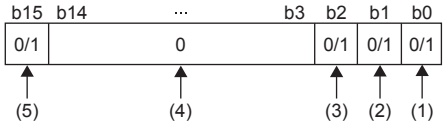



  - Comparing specified time data with current time data

    These instructions compare the time data in the device specified by (s1) with the current time data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The time data in the device specified by (s2) is regarded as dummy data and ignored.




- Set each data in binary.
- Set the "hour" data as in the 24-hour clock in the devices specified by (s1) and (s2) within the range 0 to 23.
- Set the "minute" data in the devices specified by (s1)+1 and (s2)+1 within the range 0 to 59.
- Set the "second" data in the devices specified by (s1)+2 and (s2)+2 within the range 0 to 59.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).




(1) Set "second" as comparison target.
(2) Set "minute" as comparison target.
(3) Set "hour" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current time in accordance with the conditions set in the 0 to 2 bits.

- When 0 is set to the 0 to 2 bits, the time data (hour, minute, and second) are not compared. When 1 is set, the entire time data (hour, minute, and second) are compared.
- When 0 is set to the 15 bit, the data in the device specified by (s1) and the time data in the device specified by (s2) are compared. When 1 is set, the data in the device specified by (s1) is compared with the current time. The time data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

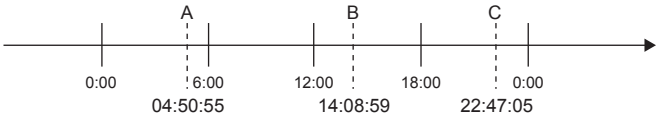
| (s3) value when comparing two specified time data | (s3) value when comparing the specified time data with the current time | Comparison target | Contents of processing |
|---|---|---|---|
| 0001H | 8001H | Second data | Only data in the device specified by (s1)+2 is compared. |
| 0002H | 8002H | Minute data | Only data in the device specified by (s1)+1 is compared. |
| 0003H | 8003H | Minute and second data | Data in the device areas specified by (s1)+2 and (s1)+2 are compared. |
| 0004H | 8004H | Hour data | Only data in the device specified by (s1) is compared. |
| 0005H | 8005H | Hour and second data | Data in the device areas specified by (s1) and (s1)+2 are compared. |
| 0006H | 8006H | Hour and minute data | Data in the device areas specified by (s1) and (s1)+1 are compared. |
| 0007H | 8007H | Hour, minute, and second data | The entire time data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared. |
| Other than 0001H to 0007H, 8001H to 8007H | | None | The entire time data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.) |

- If the comparison target data in the device are not recognized as time data, SM709 turns on after the instruction is executed and the operation result will be non-continuity. If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the corresponding device range, SM709 turns on after the instruction is executed and the operation result will be non-continuity as well. Once SM709 turns on, the on state is held until the CPU module is powered off or reset. Turn off SM709 as needed.
- The following table lists the comparison operation results of each instruction.

| Instruction symbol | Condition | Result | Instruction symbol | Condition | Result |
|---|---|---|---|---|---|
| TM= | (s1)=(s2) | Conductive state | TM= | (s1)≠(s2) | Non-conductive state |
| TM<> | (s1)≠(s2) | | TM<> | (s1)=(s2) | |
| TM> | (s1)>(s2) | | TM> | (s1)≤(s2) | |
| TM<= | (s1)≤(s2) | | TM<= | (s1)>(s2) | |
| TM< | (s1)<(s2) | | TM< | (s1)≥(s2) | |
| TM>= | (s1)≥(s2) | | TM>= | (s1)<(s2) | |

**Ex.**

The time data A, B, and C are compared.




- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

○: Continuity, ×: Non-continuity

| Comparison target data | Condition | | |
|---|---|---|---|
| | A<B | B<C | A<C |
| Second data | ○ | × | × |
| Minute data | × | ○ | × |
| Minute and second data | × | ○ | × |
| Hour data | ○ | ○ | ○ |
| Hour and second data | ○ | ○ | ○ |
| Hour and minute data | ○ | ○ | ○ |
| Hour, minute, and second data | ○ | ○ | ○ |
| None | × | × | × |

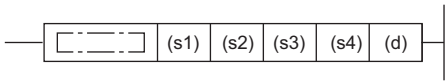
## Operation error

There is no operation error.

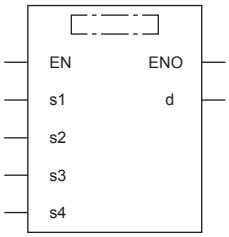
# Comparing clock data

## TCMP(P)

These instructions compare the time specified by (s1), (s2), and (s3) with the time data specified by (s4), and turn on/off the bit device specified by (d) depending on the size match.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TCMP(EN,s1,s2,s3,s4,d);<br>ENO:=TCMPP(EN,s1,s2,s3,s4,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

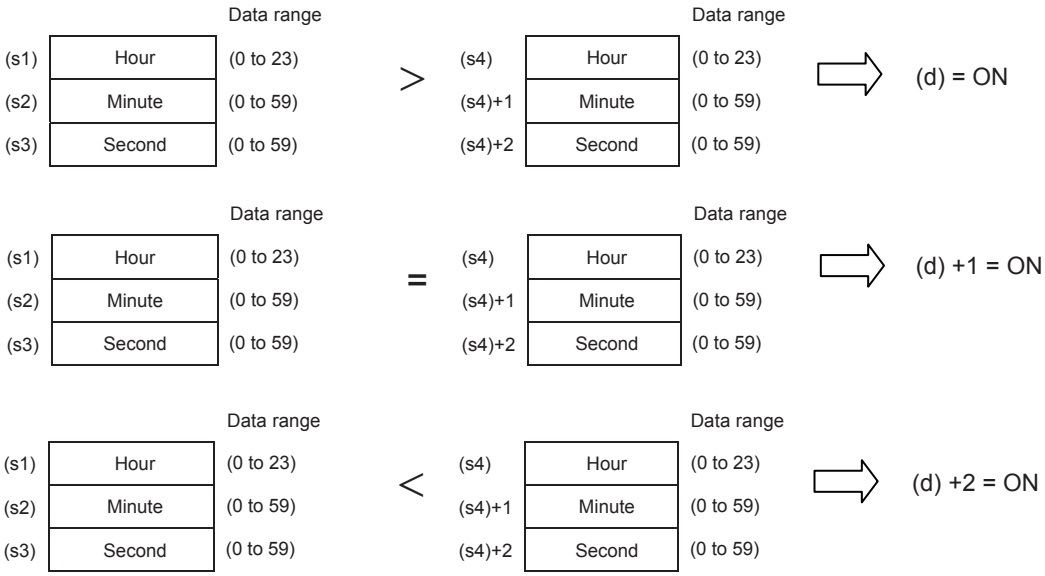
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Specify the "hour" of the time comparison | 0 to 23 | 16-bit signed binary | ANY16 |
| (s2) | Specify the "minute" of the time comparison | 0 to 59 | 16-bit signed binary | ANY16 |
| (s3) | Specify the "second" of the time comparison | 0 to 59 | 16-bit signed binary | ANY16 |
| (s4) | Specify the time data (hour, minute, and second) | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Specify the Bit device that turns on/off depending on the comparison result | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s4) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

## Processing details

- These instructions compare the time specified by (s1), (s2), and (s3) with the time data specified by (s4), and turn on/off the bit device specified by (d) depending on the size match.




- (d), (d)+1, and (d)+2 hold the state before the command contact is turned off even if, the TCMP instruction is not executed by switching on → off the command contact.

## Precautions

- Three devices are occupied by (s4) and (d). Make sure that these devices are not used by other machine controls.
- Specify each operand of the word device after reading the value of the special register used in the TRD(P) instruction when the time (hour, minute, second) of the clock data of the built-in real time clock in the CPU module is used.

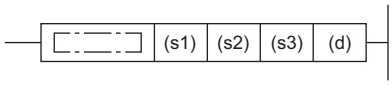
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified exceeds the corresponding device range. |
| 3405H | The value specified by (s1) and (s4) is outside the following range.<br>0 to 23 |
| | The value specified by (s2), (s3), (s4)+1, and (s4)+2 is outside the following range.<br>0 to 59 |

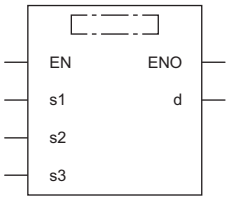
# Comparing clock data zones

## TZCP(P)

This instruction compares two comparison time (comparison time zone) specified by (s1) and (s2) with the time data specified by (s3), and turns on or off the specified bit devices (d) according to the comparison results.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TZCP(EN,s1,s2,s3,d);<br>ENO:=TZCPP(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

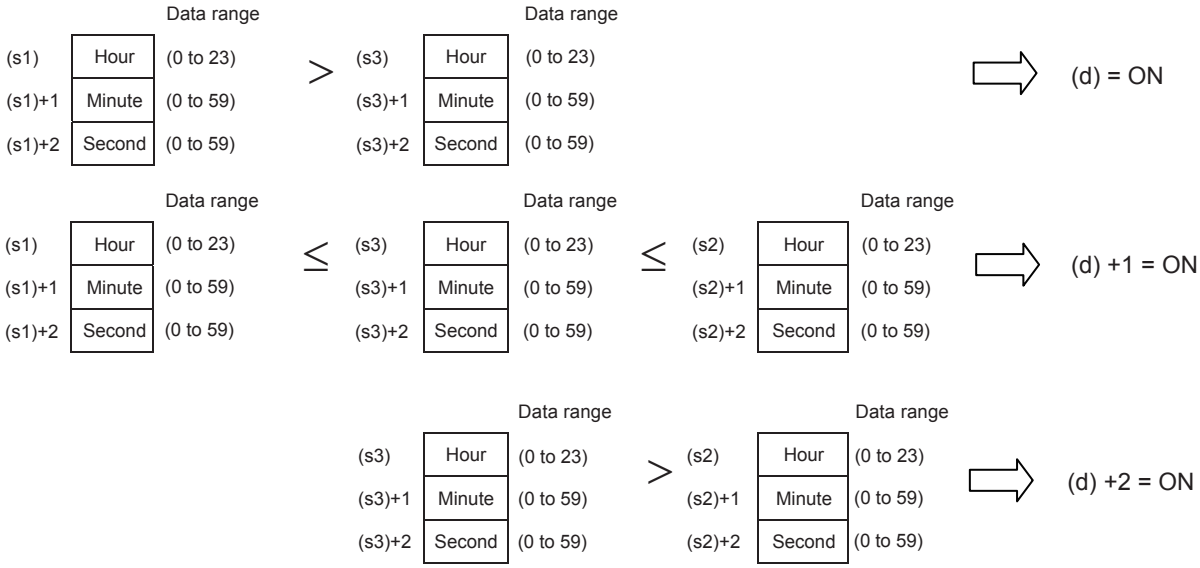
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Specify the lower limit of time comparison (hour, minute, and second). | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (s2) | Specify the upper limit of time comparison (hour, minute, and second). | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (s3) | Specify the time data (hour, minute, and second). | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 3) |
| (d) | Specify the Bit device that turns on/off depending on the comparison result | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (s2) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (s3) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

## Processing details

- This instruction compares two comparison time (comparison time zone) specified by (s1) and (s2) with the time data specified by (s3), and turns on or off the specified bit devices (d) according to the comparison results.




- Even if the command contact turns off from on and the TZCP instruction is not executed, (d), (d)+1, and (d)+2 hold the status before the command contact turned off.

## Precautions

- Three devices are occupied by (s1), (s2), (s3), and (d). Make sure that these devices are not used by other machine controls.
- When the time (hour, minute, second) of the clock data of the real time clock built in the CPU module is used, read the values of special registers by the TRD instruction, and then specify those word devices as the operands.
- Make (s1) ≤ (s2).

## Operation error

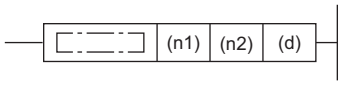
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device range specified exceeds the corresponding device range. |
| 3405H | The value specified by (s1), (s2), and (s3) is outside the following range.<br>0 to 23 |
| | The value specified by (s1)+1, (s2)+1, (s3)+1, (s1)+2, (s2)+2, and (s3)+2 is outside the following range.<br>0 to 59 |

# 8.23 Timing Check Instruction

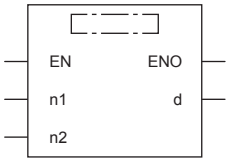
## Generating timing pulses

### DUTY

This instruction sets user timing clock output destinations (SM420 to SM424 and SM8330 to SM8334) specified by (d) to on for the number of scans specified by (n1) and to off for the number of scans specified by (n2).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DUTY(EN,n1,n2,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (n1) | Number of scans to be turned on | 0 to 32767 | 16-bit unsigned binary | ANY16 |
| (n2) | Number of scans to be turned off | 0 to 32767 | 16-bit unsigned binary | ANY16 |
| (d) | Special relay of the timing clock output destination | (SM420 to SM424, SM8330 to SM8334) | Bit | ANYBIT_BOOL |

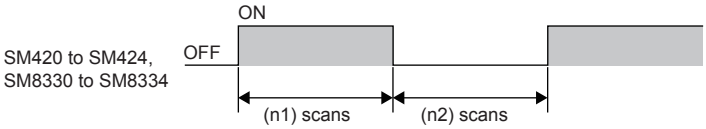
##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only SM can be used.

#### Processing details

- This instruction sets user timing clock output destinations (SM420 to SM424 and SM8330 to SM8334) specified by (d) to on for the number of scans specified by (n1) and to off for the number of scans specified by (n2).




- Specify SM420 to SM424 (SM8330 to SM8334) in the special relay of the timing clock output destination specified by (d).
- In SM420 to SM424 (SM8330 to SM8334), when one device is turned on, another device is also turned on at the same time.
- The counted number of scans is stored among SD8330 to SD8334 in accordance with the special relay of the timing clock output destination specified by (d).

- The counted number of scans stored among SD8330 to SD8334 is reset when the counted value reaches "(n1)+(n2)" or when the command input (instruction) is set to on.

| Special relay (d) for outputting the timing clock | Scan counting device |
|---|---|
| SM420(SM8330) | SD8330 |
| SM421(SM8331) | SD8331 |
| SM422(SM8332) | SD8332 |
| SM423(SM8333) | SD8333 |
| SM424(SM8334) | SD8334 |

- When the command input is set to ON, the operation is started. The special relay of the timing clock output destination is set to ON or OFF by the END instruction. Even if the command input is set to OFF, the operation is not stopped. In the STOP mode, the operation is stopped. When the power to the CPU module is turned OFF, the operation is stopped.
- When (n1) and (n2) are set to "0", the status is as shown below:

| Status of (n1) and (n2) | ON/OFF status of (d) |
|---|---|
| (n1)=0, (n2)≥0 | (d)= Fixed to OFF |
| (n1)>0, (n2)=0 | (d)= Fixed to ON |

- The table below shows the related devices.

| Special relay | Name | Description |
|---|---|---|
| SM420(SM8330) | Timing clock output 1 | Timing clock output in the DUTY instruction |
| SM421(SM8331) | Timing clock output 2 | |
| SM422(SM8332) | Timing clock output 3 | |
| SM423(SM8333) | Timing clock output 4 | |
| SM424(SM8334) | Timing clock output 5 | |

| Special register | Name | Description |
|---|---|---|
| SD8330 | Counted number of scans for timing clock output 1 | Counted number of scans for timing clock output 1 in the DUTY instruction |
| SD8331 | Counted number of scans for timing clock output 2 | Counted number of scans for timing clock output 2 in the DUTY instruction |
| SD8332 | Counted number of scans for timing clock output 3 | Counted number of scans for timing clock output 3 in the DUTY instruction |
| SD8333 | Counted number of scans for timing clock output 4 | Counted number of scans for timing clock output 4 in the DUTY instruction |
| SD8334 | Counted number of scans for timing clock output 5 | Counted number of scans for timing clock output 5 in the DUTY instruction |

## Precautions

- The DUTY instruction can be used up to 5 times (points). It is not permitted, however, to use the same timing clock output destination device for two or more DUTY instructions.

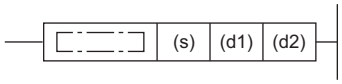
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device specified by (d) is out of the range from SM420 to SM424 (SM8330 to SM8334). |
| 3405H | The value specified by (n1), (n2) is other than 0 to 32767. |

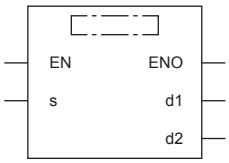
# Hour meter

## HOURM

This instruction measures the on time of the input contact in units of hour.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=HOURM(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Time after which the alarm (d2) is set to on (unit: hour) | — | 16-bit signed binary | ANY16 |
| (d1) | Device for storing the measured current value (latched (battery backed) type data register) | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 2) |
| (d2) | Device to be turned on when timeout occurs (alarm output) | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

### Processing details

- This instruction measures the period of time for which the input contact is on in units of hour, and turns on the device specified by (d2) when the accumulated ON time exceeds the time (16-bit binary data) specified in (s).
- In (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1).
- The measured current value of less than one hour (in units of second) is stored in (d1)+1.
- (d2) is set to on when the current value in (d1) exceeds the time specified by (s).
- Specify a latched (battery backed) type data register as (d1) so that the current value data can be continuously used even after the power to CPU module turns off. If a general data type register is used, the current value data is cleared when the power to the CPU module is turned OFF or when the controller mode switches from STOP to RUN.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When the current value reaches the maximum value of 16-bit data, the measurement is stopped. For continuing the measurement, clear the current value stored in (d1) to (d1)+1.

### Precautions

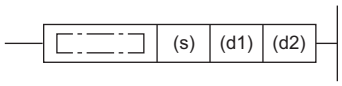
- Two devices are occupied by (d1). Make sure that these devices are not used by other machine controls.

## Operation error

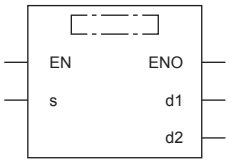
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device areas specified by (d1) exceed the corresponding device range. |
| 3405H | The value of (s) is negative. |

# DHOURM

This instruction measures the on time of the input contact in units of hour.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHOURM(EN,s,d1,d2); |

**FBD/LD**




## Setting data

### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Time after which the alarm (d2) is set to on (unit: hour) | — | 32-bit signed binary | ANY32 |
| (d1) | Device for storing the measured current value (latched (battery backed) type data register) | — | 32-bit signed binary | ANY32_ARRAY (Number of elements: 2) |
| (d2) | Device to be turned on when timeout occurs (alarm output) | — | Bit | ANY_BOOL |

### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○ | — | — | — | — | — | — | — | — | — |

## Processing details

- This instruction measures the period of time for which the input contact is on in units of hour, and turns on the device specified by (d2) when the accumulated ON time exceeds the time (32-bit binary data) specified in (s).
- In (s)+1 and (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1)+1 and (d1). ((d1)+1: highest-order, (d1): lowest-order)
- The measured current value of less than one hour (in units of second) is stored in (d1)+2.
- (d2) is set to on when the current value in (d1)+1 and (d1) exceeds the time specified by (s).
- Specify a latched (battery backed) type data register as (d1) so that the current value data can be continuously used even after the power to CPU module turns off. If a general data type register is used, the current value data is cleared when the power to the CPU module is turned OFF or when the controller mode switches from STOP to RUN.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When the current value reaches the maximum value of 32-bit data, the measurement is stopped. For continuing the measurement, clear the current value stored in (d1) to (d1)+2.

## Precautions

- Three devices are occupied by (d1). Make sure that these devices are not used by other machine controls.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device areas specified by (d1) exceed the corresponding device range. |
| 3405H | The value of (s) is negative. |

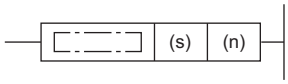
# 8.24 Module Access Instruction

## I/O refresh

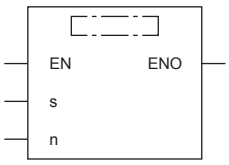
### REF(P)/RFS(P)

These instructions refresh the (n) points of devices starting from the device specified by (s), and receive an external input or generate an output.

The REF(P) instructions can also be used as RFS(P).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=REF(EN,s,n);<br>ENO:=REFP(EN,s,n);<br>ENO:=RFS(EN,s,n);<br>ENO:=RFSP(EN,s,n); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number to be refreshed | — | Bit | ANY_BOOL |
| (n) | Number of devices to be refreshed | 0 to 65535 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

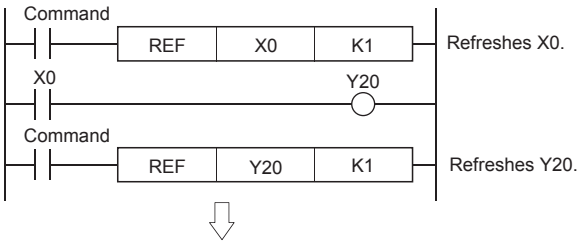
*1 Only X and Y can be used.

#### Processing details

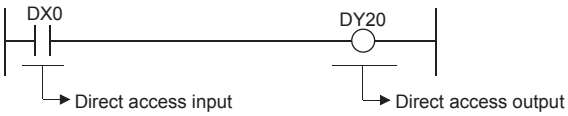
- This function refreshes only the corresponding devices in the middle of a scan and receives an external input or generates an output.
- Since the input receptions and external outputs are performed at one time only after the END instruction is executed in the program, a pulse signal cannot be output externally in the middle of a scan. The execution of the I/O refresh instruction forcibly refreshes the corresponding input (X) or output (Y) in the middle of program execution, and then a pulse signal can be output externally in the middle of a scan.

• To refresh an input (X) or an output (Y) in 1 point units, use the direct access input (DX) or the direct access output (DY).

[Program based on the REF instruction]




[Program based on direct access input and direct access output]




## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The (n) points of device range starting from the device specified by (s) exceed the range of proximal I/O. |

# Reading 1-word/2-word data from another module

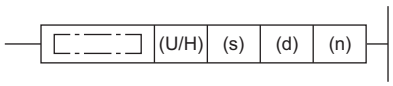
## FROM(P), DFROM(P)

• FROM(P)

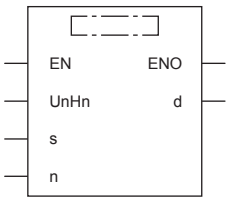
These instructions read (n) words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later (d).

• DFROM(P)

These instructions read the (n) × 2 words of data from the buffer memory specified by (s) intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FROM(EN,UnHn,s,n,d);<br>ENO:=FROMP(EN,UnHn,s,n,d);<br>ENO:=DFROM(EN,UnHn,s,n,d);<br>ENO:=DFROMP(EN,UnHn,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (U/H)*1 | | Unit number | H1 to H10 | 16-bit unsigned binary | ANY16 |
| (s) | | Start address of the buffer memory where the read-target data is stored | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (d) | FROM(P) | Head device number for storing the read data | — | 16-bit signed binary | ANY16 |
| | DFROM(P) | | | 32-bit signed binary | ANY32 |
| (n) | | Number of read data | 1 to 65535 | 16-bit unsigned binary | ANY16 |

*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

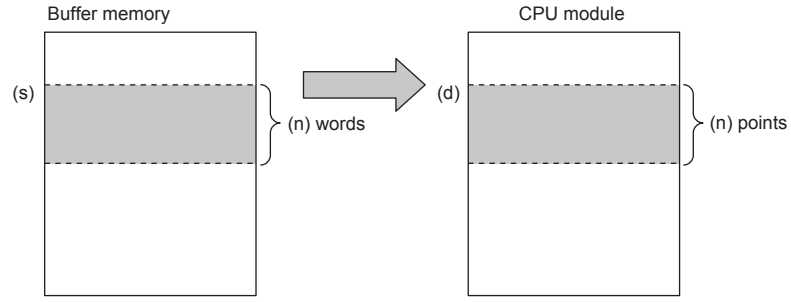
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (U) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | ○ |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | ○ | ○*1 | ○*1 | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only the DFROM(P) instruction can be used.

## Processing details

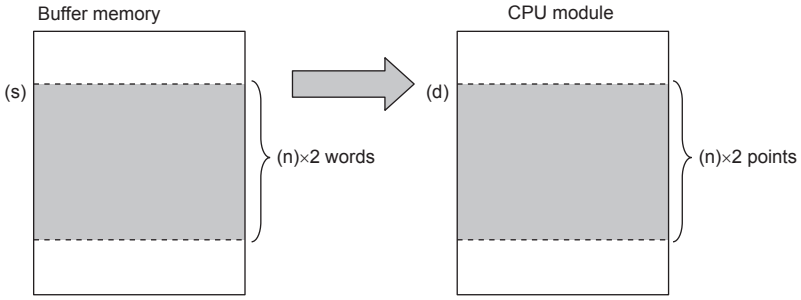
### ■FROM(P)

- These instructions read (n) words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.



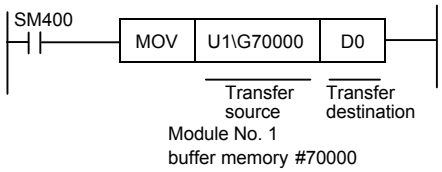

### ■DFROM(P)

- These instructions read the (n) × 2 words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.




## Precautions

- For the nibble of a bit device specified by (d), specify K1 to K4 in the FROM(P) instruction and K1 to K8 in the DFROM(P) instruction.
- When a number greater than 65535 is specified as the buffer memory specified by (s), use the FROMD(P) instruction or use U□\G□ in the MOV(P) instruction. The following shows the program to transfer the buffer memory #70000 in the intelligent function module No. 1 to D0.




## Operation error

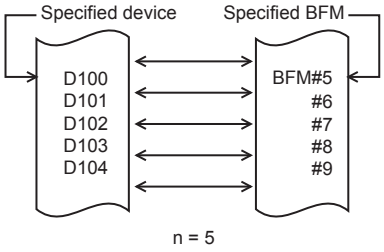
| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Updating procedure with the unit was not properly completed during the execution of the instruction. |
| 2801H | The unit number specified by (U/H) does not exist. |
| 2823H | The buffer memory number specified by (s) exceeds the buffer memory area. |
| | The buffer memory number specified by (s) + the number of transfer points specified by (n) exceeds the buffer memory area. |
| 2820H | The device number specified by (d) + the number of read data specified by (n) exceeds the corresponding device range. |
| 3056H | Timeout occurred while communicating with the connected units during the execution of the instruction. |
| 3060H | Signal error is detected while accessing the connected units during the execution of the instruction. |
| 3580H | The instruction that is disabled in the interrupt routine program is used. |

## Common items among the FROM(P), DFROM(P), TO(P), and DTO(P) (details)

- Use the module number to specify which Intelligent function module the instruction works for. The setting range is from H1 to H10 (K1 to K16).

| | | Module No. 1 | | Module No. 2 | | Module No. 3 | Module No. 4 | Module No. 5 |
|---|---|---|---|---|---|---|---|---|
| CPU module | I/O module | Intelligent function module | Extension power supply unit | Intelligent function module | I/O module | Intelligent function module | Bus conversion module | Intelligent function module |

- A module number is automatically assigned to each intelligent function module connected to a CPU module. The module number is assigned in the way "No. 1 → No. 2 → No. 3 ..." starting from the equipment nearest the CPU module.
- 16-bit RAM memories are built in an intelligent function module, and they are called buffer memories. The contents of buffer memories vary depending on the purpose of control of each Intelligent function module, and the setting range is from K0 to K65535.
- The number of read data is specified by (n), and the setting range is from K1 to K65535.



# Writing 1-word/2-word data to another module

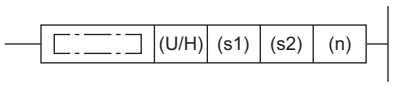
## TO(P), DTO(P)

• TO(P)

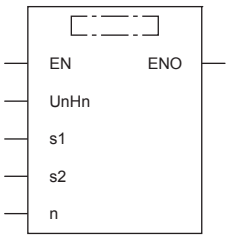
These instructions write the (n) points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

• DTO(P)

These instructions write the (n) × 2 points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TO(EN,UnHn,s1,s2,n);<br>ENO:=TOP(EN,UnHn,s1,s2,n);<br>ENO:=DTO(EN,UnHn,s1,s2,n);<br>ENO:=DTOP(EN,UnHn,s1,s2,n); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (U/H)*1 | | Unit number | H1 to H10 | 16-bit unsigned binary | ANY16 |
| (s1) | | Start address of the buffer memory for writing the data | 0 to 65535 | 16-bit unsigned binary | ANY16 |
| (s2) | TO(P) | Write data, or head device number which stores the write data | — | 16-bit signed binary | ANY16 |
| | DTO(P) | | — | 32-bit signed binary | ANY32 |
| (n) | | Number of write data | 1 to 65535 | 16-bit unsigned binary | ANY16 |

*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

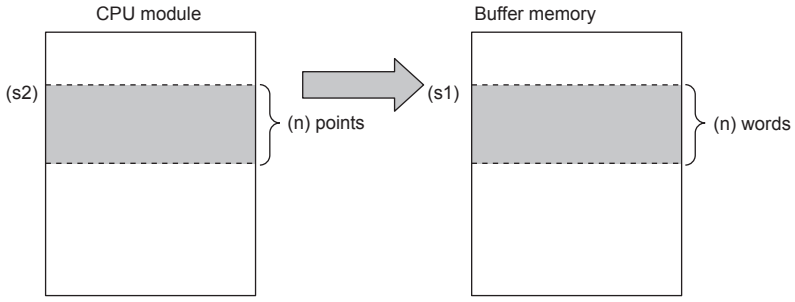
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (U) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | ○ |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | — | ○ | ○*1 | ○*1 | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only the DTO(P) instruction can be used.

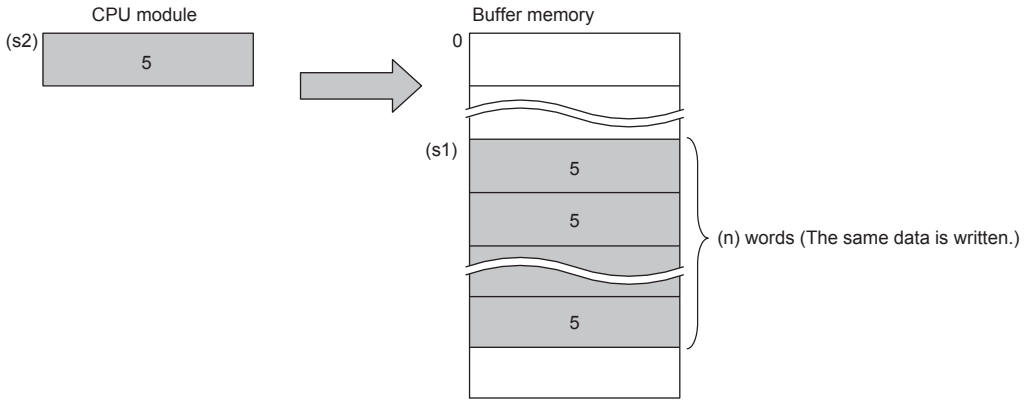
## Processing details

### ■TO(P)

- These instructions write the (n) points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

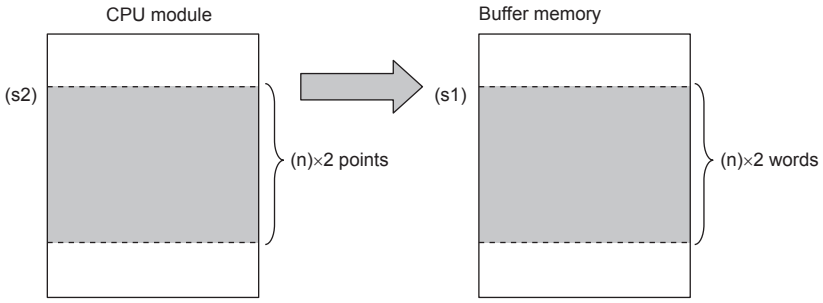



- When a constant is specified in (s2), (n) words of the same data (the value specified by (s2)) is written starting from the specified buffer memory address.

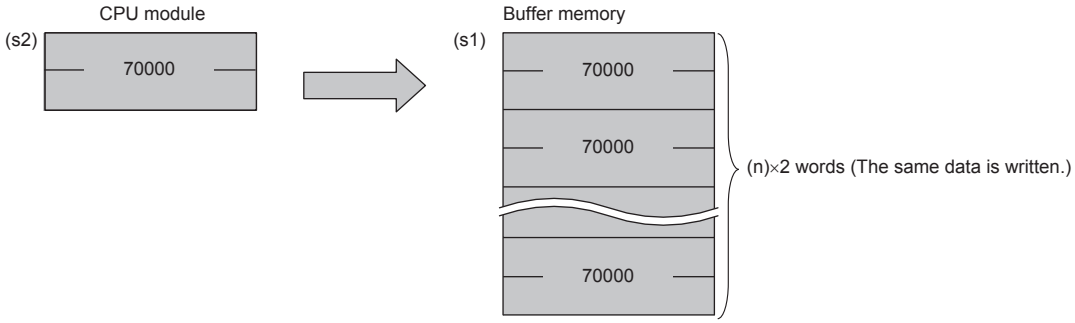


## ■DTO(P)

- These instructions write the (n) × 2 points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).




- When a constant is specified in (s2), (n) x 2 words of the same data (the value specified by (s2)) is written starting from the specified buffer memory address.




## Precautions

- For the nibble of a bit device specified by (s2), specify K1 to K4 in the TO(P) instruction and K1 to K8 in the DTO(P) instruction.
- When a number greater than 65535 is specified as the buffer memory specified by (s1), use the TOD(P) instruction or use U\G in the MOV(P) instruction.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Updating procedure with the unit was not properly completed during the execution of the instruction. |
| 2801H | The unit number specified by (U/H) does not exist. |
| 2823H | The buffer memory number specified by (s1) exceeds the buffer memory area. |
| | The buffer memory number specified by (s1) + the number of transfer points specified by (n) exceeds the buffer memory area. |
| 2820H | The device number specified by (s2) + the number of write data specified by (n) exceeds the corresponding device range. |
| 3056H | Timeout occurred while communicating with the connected units during the execution of the instruction. |
| 3060H | Signal error is detected while accessing the connected units during the execution of the instruction. |
| 3580H | The instruction that is disabled in the interrupt routine program is used. |

# Reading 1-word/2-word data from another module

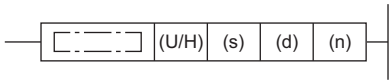
## FROMD(P), DFROMD(P)

• FROMD(P)

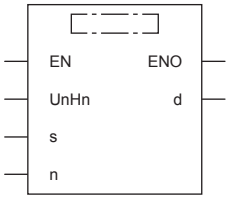
These instructions read (n) words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.

• DFROMD(P)

These instructions read the (n) × 2 words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=FROMD(EN,UnHn,s,n,d);<br>ENO:=FROMDP(EN,UnHn,s,n,d);<br>ENO:=DFROMD(EN,UnHn,s,n,d);<br>ENO:=DFROMDP(EN,UnHn,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (U/H)*1 | | Unit number | H1 to H10 | 16-bit unsigned binary | ANY16 |
| (s) | | Start address of the buffer memory where the read-target data is stored | 0 to 4294967295 | 32-bit unsigned binary | ANY32 |
| (d) | FROMD(P) | Head device number for storing the read data | — | 16-bit signed binary | ANY16 |
| | DFROMD(P) | | | 32-bit signed binary | ANY32 |
| (n) | | Number of read data | 1 to 65535 | 32-bit unsigned binary | ANY32 |

*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

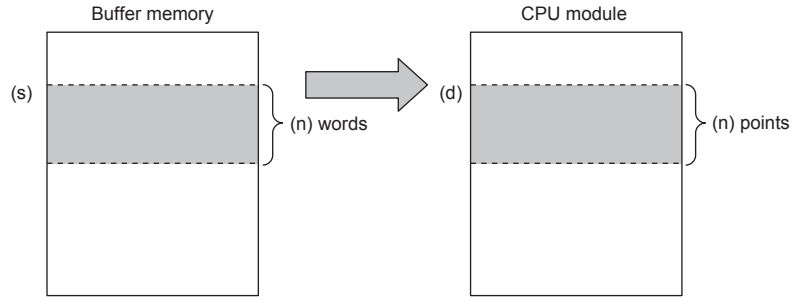
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (U) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | ○ |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | ○ | ○*1 | ○*1 | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |

*1 Only the DFROMD(P) instruction can be used.

## Processing details

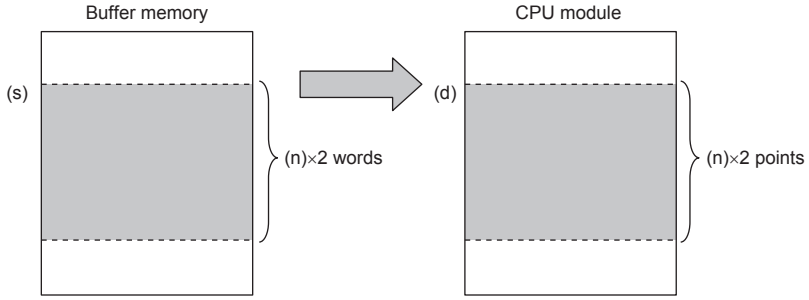
### ■FROMD(P)

- These instructions read (n) words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.




### ■DFROMD(P)

- These instructions read the (n) × 2 words of data from the buffer memory specified by (s) in intelligent function module specified by (U/H), and store the data to the device specified by (d) and later.




## Precautions

- For the nibble of a bit device specified by (d), specify K1 to K4 in the FROMD(P) instruction and K1 to K8 in the DFROMD(P) instruction.

## Operation error

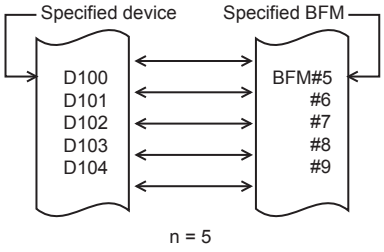
| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Updating procedure with the unit was not properly completed during the execution of the instruction. |
| 2801H | The unit number specified by (U/H) does not exist. |
| 2823H | The buffer memory number specified by (s) exceeds the buffer memory area. |
| | The buffer memory number specified by (s) + the number of transfer points specified by (n) exceeds the buffer memory area. |
| 2820H | The device number specified by (d) + the number of read data specified by (n) exceeds the corresponding device range. |
| 3056H | Timeout occurred while communicating with the connected units during the execution of the instruction. |
| 3060H | Signal error is detected while accessing the connected units during the execution of the instruction. |
| 3580H | The instruction that is disabled in the interrupt routine program is used. |

## Common items among the FROMD(P), DFROMD(P), TOD(P), and DTOD(P) (details)

- Use the module number to specify which intelligent function module the instruction works for. The setting range is from H1 to H10 (K1 to K16).

| | | Module No. 1 | | Module No. 2 | | Module No. 3 | Module No. 4 | Module No. 5 |
|---|---|---|---|---|---|---|---|---|
| CPU module | I/O module | Intelligent function module | Extension power supply unit | Intelligent function module | I/O module | Intelligent function module | Bus conversion module | Intelligent function module |

- A module number is automatically assigned to each intelligent function module connected to a CPU module. The module number is assigned in the way "No. 1 → No. 2 → No. 3 ..." starting from the equipment nearest the CPU module.
- 16-bit RAM memories are built in an intelligent function module, and they are called buffer memories. The contents of buffer memories vary depending on the purpose of control of each intelligent function module, and the setting range is from K0 to K4294967295.
- The number of read data is specified by (n), and the setting range is from K1 to K65535.



# Writing 1-word/2-word data to another module (32-bit specification)

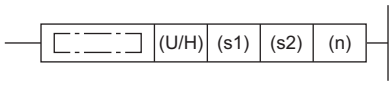
## TOD(P), DTOD(P)

• TOD(P)

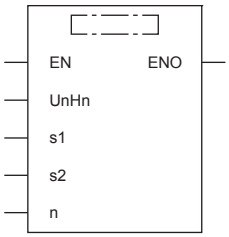
These instructions write the (n) points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

• DTOD(P)

These instructions write the (n) × 2 points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TOD(EN,UnHn,s1,s2,n);<br>ENO:=TODP(EN,UnHn,s1,s2,n);<br>ENO:=DTOD(EN,UnHn,s1,s2,n);<br>ENO:=DTODP(EN,UnHn,s1,s2,n); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (U/H)*1 | | Unit number | H1 to H10 | 16-bit unsigned binary | ANY16 |
| (s1) | | Start address of the buffer memory for writing the data | 0 to 4294967295 | 32-bit unsigned binary | ANY32 |
| (s2) | TOD(P) | Write data, or head device number which stores the write data | — | 16-bit signed binary | ANY16 |
| | DTOD(P) | | — | 32-bit signed binary | ANY32 |
| (n) | | Number of write data | 1 to 65535 | 32-bit unsigned binary | ANY32 |

*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

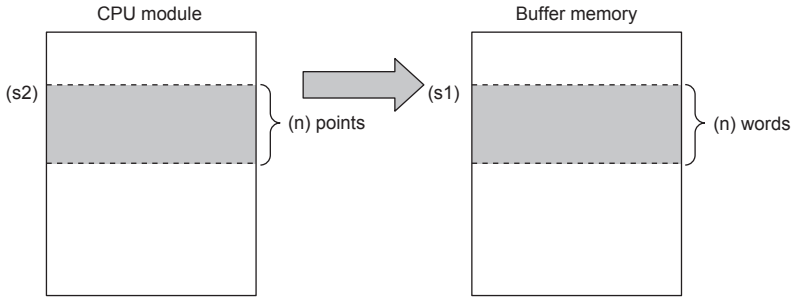
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (U) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | ○ |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | — | ○ | ○*1 | ○*1 | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |

*1 Only the DTOD(P) instruction can be used.

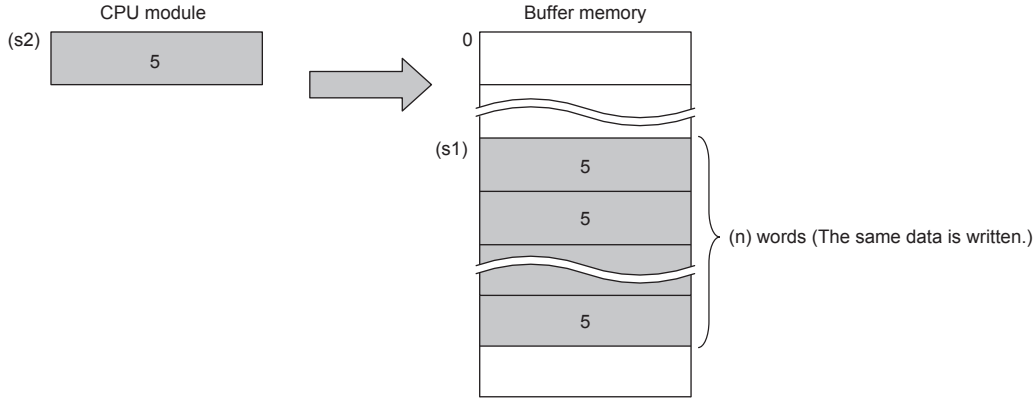
## Processing details

### ■TOD(P)

- These instructions write the (n) points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).

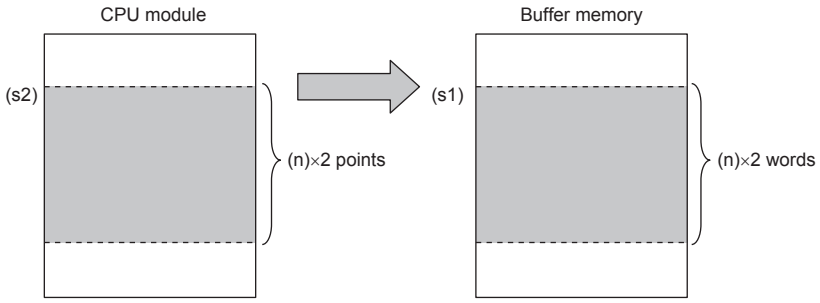



- When a constant is specified in (s2), (n) words of the same data (the value specified by (s2)) is written starting from the specified buffer memory address.

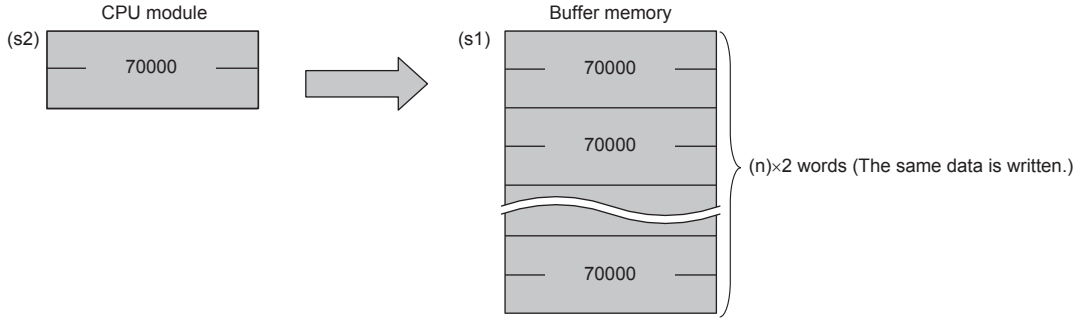


## ■DTOD(P)

- These instructions write the (n) × 2 points of data in the device starting from the one specified by (s2) to the buffer memory address specified by (s1) in intelligent function module specified by (U/H).




- When a constant is specified in (s2), (n) × 2 words of the same data (the value specified by (s2)) is written starting from the specified buffer memory address.




## Precautions

- For the nibble of a bit device specified by (s2), specify K1 to K4 in the TOD(P) instruction and K1 to K8 in the DTOD(P) instruction.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Updating procedure with the unit was not properly completed during the execution of the instruction. |
| 2801H | The unit number specified by (U/H) does not exist. |
| 2823H | The buffer memory number specified by (s1) exceeds the buffer memory area. |
| | The buffer memory number specified by (s1) + the number of transfer points specified by (n) exceeds the buffer memory area. |
| 2820H | The device number specified by (s2) + the number of write data specified by (n) exceeds the corresponding device range. |
| 3056H | Timeout occurred while communicating with the connected units during the execution of the instruction. |
| 3060H | Signal error is detected while accessing the connected units during the execution of the instruction. |
| 3580H | The instruction that is disabled in the interrupt routine program is used. |

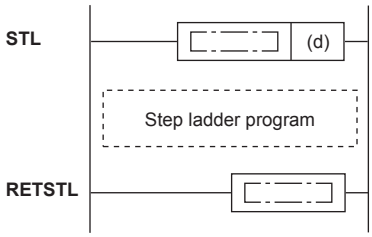
# 9 STEP LADDER INSTRUCTIONS

## 9.1 Starts/Ends Step Ladder

### STL, RETSTL

STL: This instruction starts step ladder.

RETSTL: This instruction ends step ladder.

| Ladder diagram | Structured text |
|---|---|
|  | Not supported. |

| FBD/LD |
|---|
| Not supported. |

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | The step relay number which assigns the State | 0 to 4096 | Bit | ANY_BOOL |

##### ■Applicable devices

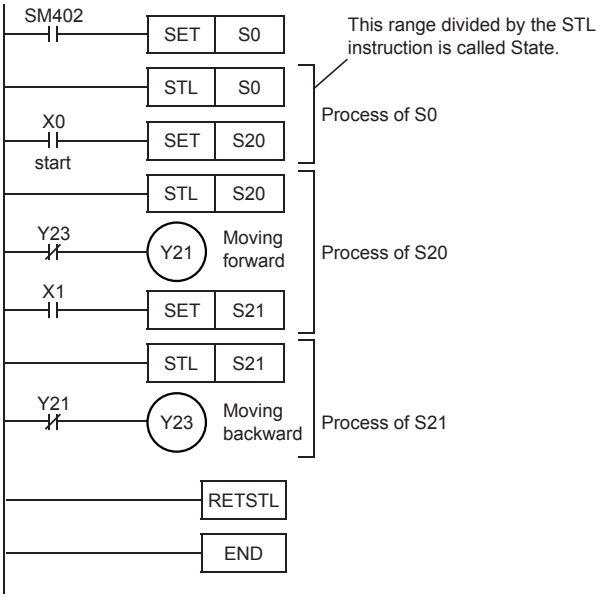
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Only S can be used.

#### Processing details

- In programs using step ladder instructions, a step relay S is assigned to each process based on machine operations, and sequences of input condition and output control are programmed as circuits connected to contacts (STL contacts) of state relays.
- In a step ladder program, a step relay S is regarded as one control process, and a sequence of input conditions and output controls are programmed in a state relay. Because the preceding process is stopped when the program execution proceeds to the next process, a machine can be controlled using simple sequences for each process.
- The step relay number specified by STL instruction is assigned to the State. The start and completion of the State use SET instruction, RST instruction, and ZRST instruction.
- Program a step ladder program starting from the initial state relay in the order of state relay ON status transfer. Make sure to put the RETSTL instruction at the end of a step ladder program. The RETSTL instruction is omissible anywhere other than the last Step Ladder.

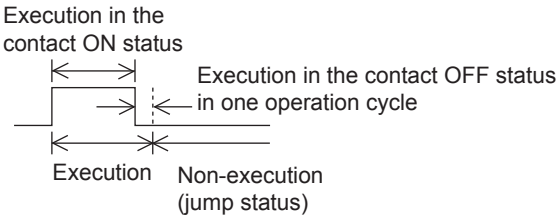
The actual step ladder program is as follows.



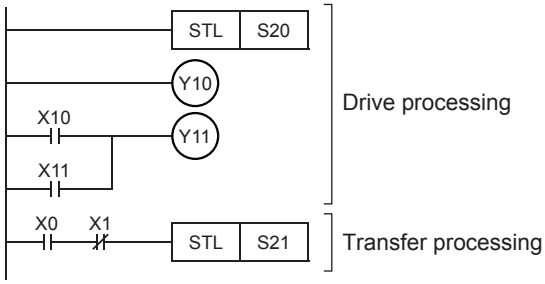

- A step ladder program is expressed as a relay ladder, but it can be created according to the machine control flow using state relays. A state relay consists of a drive coil and contact (STL contact) in the same way as other relays. Use SET or OUT instructions to drive a coil, and use STL instructions for a contact.
- The table below shows operations of internal circuits connected to the state relays,

| Internal circuit diagram operation | |
|---|---|
| Execution in the contact ON status | When a state relay turns ON, a connected circuit (internal circuit) is activated with a STL contact. |
| Execution in the contact OFF status (for one operation cycle) | When a condition (transfer condition) provided between state relays is satisfied, the next state relay turns ON, and the state relay which was been ON before hand turns OFF (transfer operation). In the state relay ON status transfer process, both state relays are ON for one operation cycle.<br>In the next operation cycle after the ON status is transferred to the next state relay, the former state relay is reset to OFF. A drive instruction connected to the bus line of the reset state relay is executed in the contact OFF status for one operation cycle regardless of the actual contact status before the drive instruction.<br>When the transfer state relay is used in a contact instruction, however, the contact image is executed in the OFF status immediately after the transfer condition is satisfied. |
| Non-execution | An instruction is not executed in the contact OFF status after the operation cycle where the instruction was executed in the contact OFF status (jump status). |

- The figure below shows the timing chart of the state relay (internal circuit) activation status.

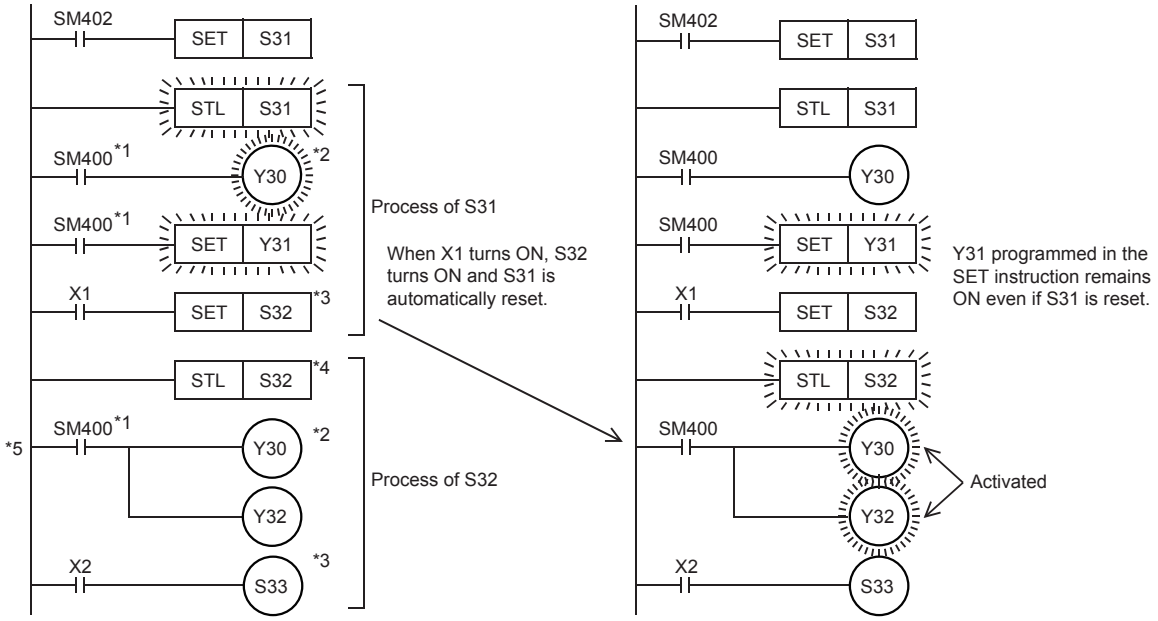



- Each state relay has three functions, driving a load, specifying a transfer destination and specifying a transfer condition. Such as the following program, driving a load and then transition processing. In a state relay without any load, the drive processing is not required.



## ■Operation of program

The step ladder program operates as follows.




*1 It is always necessary to program to drive an output.
*2 Output coils can be used again in different state relays.
*3 Each OUT and SET instruction for step relays automatically resets the transfer source, and has the self-holding function.
*4 One step relay number can only be used once.
*5 It is not possible to place a pointer immediately after the STL instruction. If a pointer is placed, a program error (33E2H) occurs.

**Point**

When the step relay is latched, the ON/OFF status is backed up by nonvolatile memory against power failure. Use this type of state relay if the operation should be restarted from the last point before power failure. These relays hold the operation status also at the time of RUN to STOP. If it RUN(s) again, the operation will be resumed from the State before STOP.

## ■Related devices

| Device | Name | Remarks |
|---|---|---|
| SM8040 | STL transfer disable | When SM8040 is set to ON, transfer of the ON status is disabled among all state relays. |
| SM8046 | STL state ON | When the step relay turns on, SM8046 turns on automatically. |
| SM8047 | Enable STL monitoring | When SM8047 is turned on, it stores in SD8040 to SD8047 sequentially from the young number of the step relay which operates in the step relay. |
| SD8040 to SD8047 | ON step relay numbers | The turned-on step relay number is stored in SD8040 to SD8047 (Max. 8 points) sequentially from the smaller number. |

### Precautions

When the step relay is not latched, step relay clears at power supply ON→OFF or RUN→STOP. If the State is valid and power supply ON→OFF or PLC RUN→STOP, operation can not be restarted from the last point before power-supply ON→OFF or RUN→STOP.

### Operation error

There is no operation error.

# 10 BUILT-IN ETHERNET FUNCTION INSTRUCTIONS

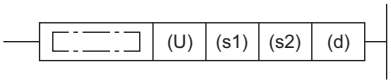


## 10.1 Open/Close Processing Instructions

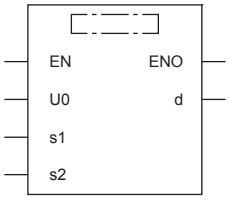
### Opening a connection

#### SP.SOCOPEN

This instruction opens a connection.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_SOCOPEN(EN,U0,s1,s2,d); |

**FBD/LD**




("SP_SOCOPEN" enters ☐.)

##### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for storing the control data | Refer to Control data (☞ Page 694) | Word | ANY16_ARRAY (Number of elements: 10) |
| (d) | Head device number which turns ON when the execution of the instruction is completed and remains ON for 1 scan. If the instruction is completed with an error, (d)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

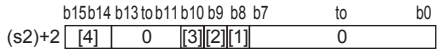
*1 In the case of the ST language and the FBD/LD language, U displays as U0.

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U☐\G☐ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U☐\G☐ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## ■Control data

| Device | Item | Description | Setting range | Set by[*1] |
|---|---|---|---|---|
| (s2)+0 | Execution type/ completion type | Specify whether to use the parameter value set using the engineering tool or to use the set values of the control data (s2)+2 to (s2)+9 during the open processing of the connection.<br><br>0000H:<br>The open processing is performed with the settings configured using "External Device Configuration" of the engineering tool.<br>8000H:<br>The open processing is performed with the set values of the control data (s2)+2 to (s2)+9. | 0000H<br>8000H | User |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |
| (s2)+2 | Application setting area |  <br>[1] Communication method (protocol)<br>0: TCP/IP<br>1: UDP/IP<br>[2] Socket communications function procedure<br>0: Communication protocol<br>1: Socket communications (No procedure)<br>[3] Communication protocol setting<br>0: Do not use the communication protocol support function (use the socket communications function)<br>1: Use the protocol support function<br>[4] Open method<br>00: Active open or UDP/IP<br>10: Unpassive open<br>11: Fullpassive open | As shown on the left | User |
| (s2)+3 | Host station port number | Specify the host station port number. | 1 to 5548, 5570 to 65534 (0001H to 15ACH, 15C2H to FFFEH)[*3] | |
| (s2)+4<br>(s2)+5 | Target device IP address[*2] | Specify the IP address of the target device. | 1 to 3758096382 (00000001H to DFFFFFFEH)<br>(FFFFFFFFH: Simultaneous broadcast) | |
| (s2)+6 | Target device port number[*2] | Specify the port number of the target device. | 1 to 65534 (0001H to FFFEH)<br>(FFFFH: Simultaneous broadcast) | |
| (s2)+7 to (s2)+9 | — | Use prohibited | — | System |

*1 User: Data to be set before the execution of the instruction. System: The CPU module stores the execution result of the instruction.

*2 When Unpassive open is selected, the target device IP address and target device port number are ignored.

*3 Of the host station port numbers, 1 to 1023 (0001H to 03FFH) are generally reserved port numbers and 61440 to 65534 (F000H to FFFEH) are used by other communication functions. Thus, using 1024 to 5548 and 5570 to 61439 (0400H to 15ACH and 15C2H to EFFFH) as the port numbers is recommended. Do not specify 5549 to 5569 (15ADH to 15C1H) since they are used by the system.

## Processing details

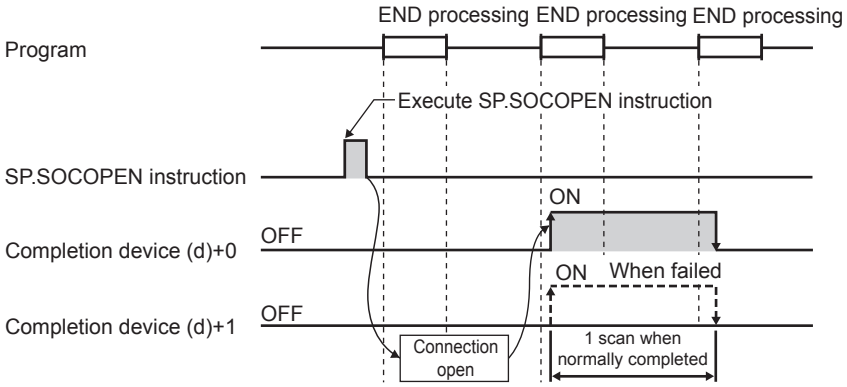
This instruction performs the open processing for the connection specified by (s1).

The setting value used by the open processing is selected by (s2)+0.

The completion of the SP.SOCOPEN instruction can be checked using the completion devices (d)+0 and (d)+1.

- Completion device (d)+0: Turns ON during the END processing for the scan in which the SP.SOCOPEN instruction is completed, and turns OFF during the next END processing.
- Completion device (d)+1: Turns ON or OFF depending on the status of when the SP.SOCOPEN instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the SP.SOCOPEN instruction is completed, and turns OFF during the next END processing. |




- The connection in which no protocol is set with the parameter can be opened and used. In this case, specify 8000H in (s2)+0 and the contents of the open processing in (s2)+2 to (s2)+9.

For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

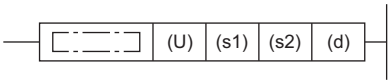
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The device number specified by (s2) or (d) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |
| 3582H | When an instruction which cannot be used in interruption routine program is used. |

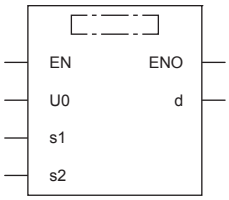
# Closing a connection

## SP.SOCCLOSE

This instruction closes a connection.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_SOCCLOSE(EN,U0,s1,s2,d); |

**FBD/LD**




("SP_SOCCLOSE" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for storing the control data | Refer to Control data (☞ Page 696) | Word | ANY16_ARRAY (Number of elements: 2) |
| (d) | Head device number which turns on when the execution of the instruction is completed and remains on for 1 scan. If the instruction is completed with an error, (d)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

#### ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s2)+0 | System area | — | — | — |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to 📖MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |

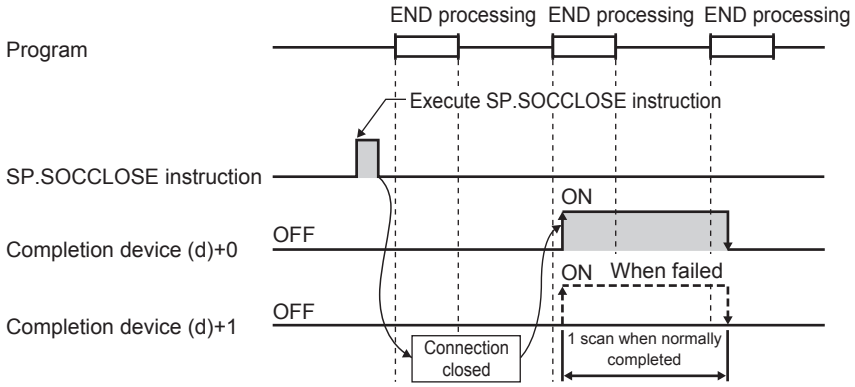
*1 System: The CPU module stores the execution result of the instruction.

## Processing details

This instruction performs the close processing for the connection specified by (s1). (Connection disconnection)
The completion of the SP.SOCCLOSE instruction can be checked using the completion devices (d)+0 and (d)+1.

- Completion device (d)+0: Turns ON during the END processing for the scan in which the SP.SOCCLOSE instruction is completed, and turns OFF during the next END processing.
- Completion device (d)+1: Turns ON or OFF depending on the status when the SP.SOCCLOSE instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the SP.SOCCLOSE instruction is completed, and turns OFF during the next END processing. |




For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The device number specified by (s2) or (d) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |
| 3582H | When an instruction which cannot be used in interruption routine program is used. |

**Point**

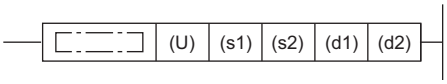
Do not execute the SP.SOCCLOSE instruction when Passive open is selected. Since the open completion signal and open request signal of the corresponding connection turn OFF and close processing is executed, the communication is disabled.

# 10.2 Socket Communications Function Instructions

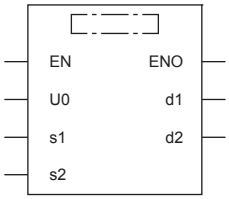
## Reading receive data during the END processing

### SP.SOCRCV

This instruction reads the receive data. (Reading during END processing)

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_SOCRCV(EN,U0,s1,s2,d1,d2); |

**FBD/LD**




("SP_SOCRCV" enters □.)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for specifying the control data | Refer to Control data (☞ Page 699) | Word | ANY16_ARRAY (Number of elements: 2) |
| (d1) | Head device number for storing the receive data | — | Word | ANY16 |
| (d2) | Head device number which turns ON when the execution of the instruction is completed and remains ON for 1 scan. If the instruction is completed with an error, (d2)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d1) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s2)+0 | System area | — | — | — |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |
| (d1)+0 | Receive data length | The data length of the data read from the socket communication receive data area is stored. (Number of bytes) | 0 to 2046 | System |
| (d1)+1 to (d1)+n | Receive data | The data read from the socket communication receive data area is sequentially stored. | — | System |

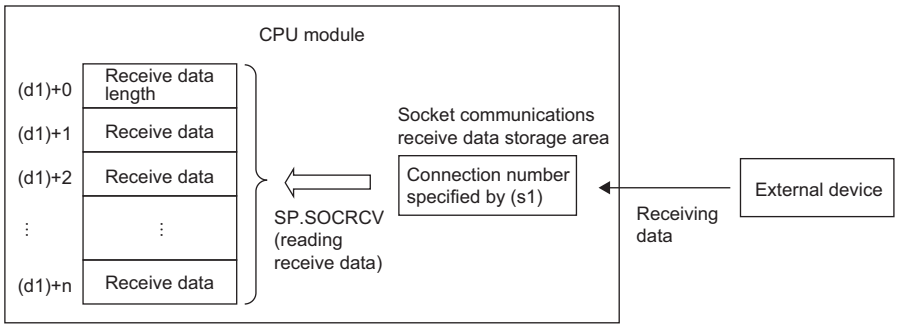
*1 System: The CPU module stores the execution result of the instruction.

**Point**

- When the SP.SOCRCV instruction is executed, reading data from the socket communication receive data area is executed with the END processing. Thus, executing the SP.SOCRCV instruction extends the scan time.
- When the data of odd-number of bytes is received, invalid data is stored in the higher byte of the device where the last receive data is stored.

## Processing details

In the END processing after the execution of the SP.SOCRCV instruction, the receive data of the connection specified by (s1) is read from the socket communication receive data area.

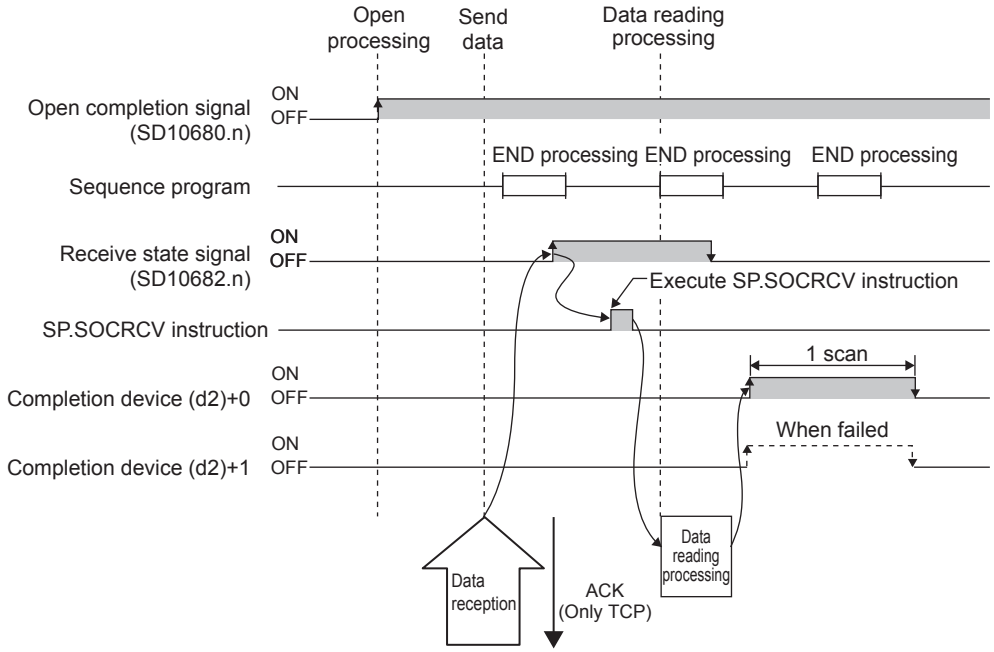



The completion of the SP.SOCRCV instruction can be checked using the completion devices (d2)+0 and (d2)+1.

- Completion device (d2)+0: Turns ON during the END processing for the scan in which the SP.SOCRCV instruction is completed, and turns OFF during the next END processing.
- Completion device (d2)+1: Turns ON or OFF depending on the status when the SP.SOCRCV instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the SP.SOCRCV instruction is completed, and turns OFF during the next END processing. |

The following figure shows the timing of the receive processing with the SP.SOCRCV instruction.




For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

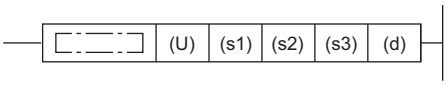
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The size of the receive data exceeds the size of the receive data storage device. |
| | The device number specified by (s2), (d1), or (d2) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |
| 3582H | When an instruction which cannot be used in interruption routine program is used. |

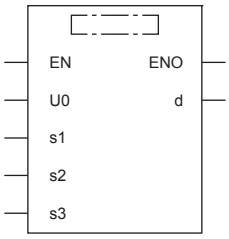
# Sending data

## SP.SOCSND

This instruction sends data.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_SOCSND(EN,U0,s1,s2,s3,d); |

**FBD/LD**




("SP_SOCSND" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for specifying the control data | Refer to Control data (☞ Page 702) | Word | ANY16_ARRAY (Number of elements: 2) |
| (s3) | Head device number for storing the send data | — | Word | ANY16 |
| (d) | Head device number which turns ON when the execution of the instruction is completed and remains on for 1 scan. If the instruction is completed with an error, (d)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (s3) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s2)+0 | System area | — | — | — |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |
| (s3)+0 | Send data length | Specifies the send data length. (Number of bytes) | 1 to 2046 | User |
| (s3)+1 to (s3)+n | Send data | Specifies the send data. | — | User |

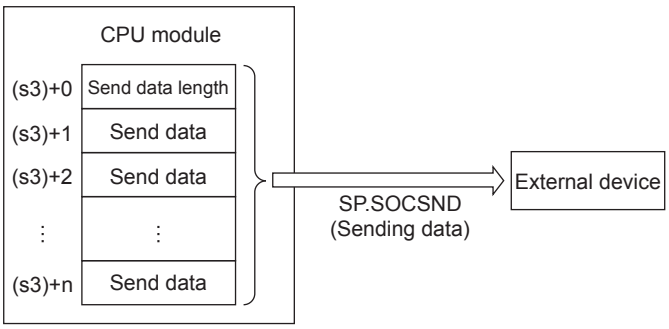
*1 User: Data to be set before the execution of the instruction. System: The CPU module stores the execution result of the instruction.

**Point**

When TCP is used, specify send data length that is smaller than the maximum window size of the target device (Receive data buffer of TCP). Data whose size exceeds the maximum window size of the target device cannot be sent.

## Processing details

This instruction send the data set by (s3) to the target device of the connection specified by (s1).



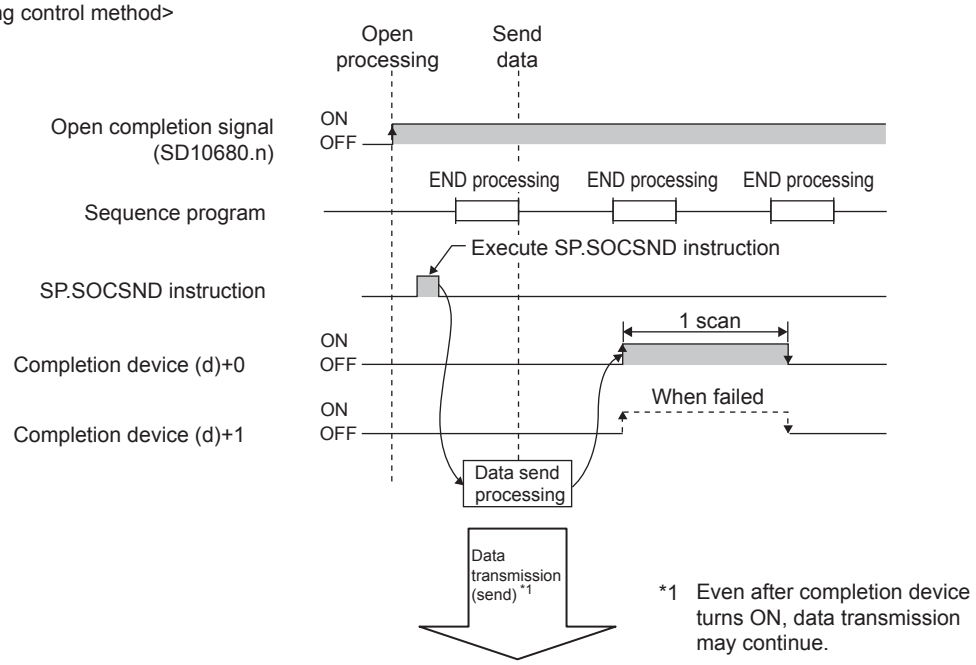

The completion of the SP.SOCSND instruction can be checked using the completion devices (d)+0 and (d)+1.

- Completion device (d)+0: Turns ON during the END processing for the scan in which the SP.SOCSND instruction is completed, and turns OFF during the next END processing.
- Completion device (d)+1: Turns ON or OFF depending on the status when the SP.SOCSND instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the SP.SOCSND instruction is completed, and turns OFF during the next END processing. |

The following figure shows the timing of the send processing with the SP.SOCSND instruction.

<Sending control method>




For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

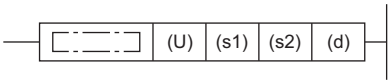
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The device number specified by (s2), (s3), or (d) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |
| 3582H | When an instruction which cannot be used in interruption routine program is used. |

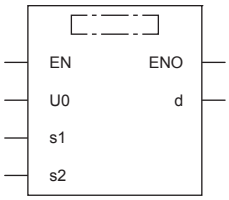
# Reading connection information

## SP.SOCCINF

This instruction reads the connection information.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_SOCCINF(EN,U0,s1,s2,d); |

**FBD/LD**




("SP_SOCCINF" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

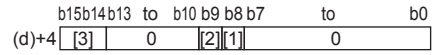
| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for storing the control data | Refer to Control data (☞ Page 705) | Word | ANY16_ARRAY (Number of elements: 2) |
| (d) | Head device number for storing the connection information | — | Word | ANY16_ARRAY (Number of elements: 5) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (DY) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |

## ■Control data

| Device | Item | Description | Setting range | Set by[*1] |
|---|---|---|---|---|
| (s2)+0 | System area | — | — | — |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to 📖MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |
| (d)+0<br>(d)+1 | Target device IP address | The IP address of the target device is stored. | 1 to 3758096382 (00000001H to DFFFFFFEH)[*2] | System |
| (d)+2 | Target device port number | The port number of the target device is stored. | 1 to 65534 (0001H to FFFEH)[*2] | |
| (d)+3 | Host station port number | The host station port number is stored. | 1 to 5548, 5570 to 65534 (0001H to 15ACH, 15C2H to FFFEH)[*2*3] | |
| (d)+4 | Application setting area |  <br>[1] Communication method (protocol)<br>0: TCP/IP<br>1: UDP/IP<br>[2] Procedure of the socket communication function<br>1: Non-protocol method<br>[3] Open method<br>00: Active open or UDP/IP<br>10: Unpassive open<br>11: Fullpassive open | As shown on the left[*2] | |

*1 System: The CPU module stores the execution result of the instruction.

*2 When the instruction is executed for a connection that is not open, 0H is returned.

*3 Of the host station port numbers, 1 to 1023 (0001H to 03FFH) are generally reserved port numbers and 61440 to 65534 (F000H to FFFEH) are used by other communication functions. Thus, using 1024 to 5548 and 5570 to 61439 (0400H to 15ACH and 15C2H to EFFFH) as the port numbers is recommended. Do not specify 5549 to 5569 (15ADH to 15C1H) since they are used by the system.

## Processing details

This instruction reads the connection information of the connection specified by (s1).

For details, refer to 📖MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

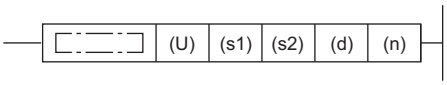
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The device number specified by (s2) or (d) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |

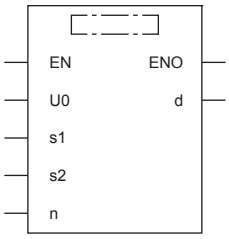
# Reading socket communications receive data

## S(P).SOCRDATA

This instruction reads the data in the socket communication receive data area.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=S_SOCRDATA(EN,U0,s1,s2,n,d);<br>ENO:=SP_SOCRDATA(EN,U0,s1,s2,n,d); |

**FBD/LD**




("S_SOCRDATA", "SP_SOCRDATA" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Head device number for storing the control data | Refer to Control data (☞ Page 706) | Word | ANY16_ARRAY (Number of elements: 2) |
| (d) | Head device number for storing the read data | — | Word | ANY16 |
| (n) | Number of the read data (1 to 1024 words) | 1 to 1024 | 16-bit signed binary | ANY16 |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | — | — | — | — | — | — | — | — | ○ | — |
| (s1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (n) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |

#### ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s2)+0 | System area | — | — | — |
| (s2)+1 | Completion status | The status at the completion of the instruction is stored.<br>0000H: Completed successfully<br>Other than 0000H: Completed with an error (error code)<br>For error codes, refer to 📖MELSEC iQ-F FX5 User's Manual (Ethernet Communication). | — | System |

*1 System: The CPU module stores the execution result of the instruction.

## Processing details

These instructions read the data for the number of words specified by (n) from the socket communication receive data area of the connection specified by (s1) to the devices from the device specified by (d) onwards. No processing is performed when (n) is 0.



**Point**

- When (n) is 1, the receive data length can be read. By doing this, the device for storing the receive data can be changed when the SP.SOCRCV instruction is executed.

For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

## Precautions

- Even when the S(P).SOCRDATA instructions are executed, the socket communication receive data area is not cleared and the receiving status signal does not change. Therefore, the next receive data is not stored in the socket communication receive data area.
- To update receive data, use the SP.SOCRCV instruction to read the receive data.

## Operation error

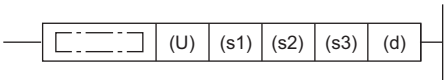
| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The connection number specified by (s1) is other than 1 to 8. |
| 2820H | The device number specified by (s2), (d), or (n) is outside the range of the number of device points. |
| 2822H | Device that cannot be specified is specified. |

# 10.3 Predefined Protocol Support Function Instruction

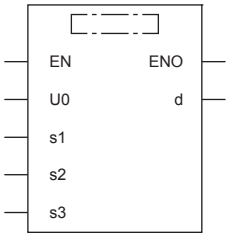
## Executing the registered protocols

### SP.ECPRTCL

This instruction executes the communication protocol registered using the engineering tool.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=SP_ECPRTCL(EN,U0,s1,s2,s3,d); |

**FBD/LD**




("SP_ECPRTCL" enters □.)

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (s1) | Connection number | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s2) | Number of protocols to be executed continuously | 1 to 8 | 16-bit unsigned binary | ANY16 |
| (s3) | Head device number for storing the control data | Refer to Control data (☞ Page 708) | Word | ANY16_ARRAY (Number of elements: 18) |
| (d) | Head device number which turns ON when the execution of the instruction is completed and remains on for 1 scan. If the instruction is completed with an error, (d)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | ○ | — | — | — | — | ○ | — | — | ○ | — |
| (s1) | ○ | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

##### ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s3)+0 | Resulting number of executed protocols | The number of protocols executed by the SP.ECPRTCL instruction is stored. Any protocol where an error occurred is also included in the execution number. If the setting of setting data or control data contains an error, "0" is stored. | 0, 1 to 8 | System |

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s3)+1 | Completion status | The completion status is stored upon completion of the instruction.<br>When two or more protocols are executed, the execution result of the protocol executed last is stored.<br>• 0: Normal completion<br>• Other than 0: Error completion (error code) | — | System |
| (s3)+2 | Execution protocol number 1 | Specify the number of the protocol to be executed first. | 1 to 64 | User |
| (s3)+3 | Execution protocol number 2 | Specify the number of the protocol to be executed second. | 0, 1 to 64 | |
| (s3)+4 | Execution protocol number 3 | Specify the number of the protocol to be executed third. | 0, 1 to 64 | |
| (s3)+5 | Execution protocol number 4 | Specify the number of the protocol to be executed fourth. | 0, 1 to 64 | |
| (s3)+6 | Execution protocol number 5 | Specify the number of the protocol to be executed fifth. | 0, 1 to 64 | |
| (s3)+7 | Execution protocol number 6 | Specify the number of the protocol to be executed sixth. | 0, 1 to 64 | |
| (s3)+8 | Execution protocol number 7 | Specify the number of the protocol to be executed seventh. | 0, 1 to 64 | |
| (s3)+9 | Execution protocol number 8 | Specify the number of the protocol to be executed eighth. | 0, 1 to 64 | |
| (s3)+10 | Collation match Receive packet number 1 | If receiving is included in the communication type of the protocol that has been executed first, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the first protocol, "0" is stored. | 0, 1 to 16 | System |
| (s3)+11 | Collation match Receive packet number 2 | If receiving is included in the communication type of the protocol that has been executed second, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the second protocol, "0" is stored. If the number of protocols executed is less than 2, "0" is stored. | 0, 1 to 16 | |
| (s3)+12 | Collation match Receive packet number 3 | If receiving is included in the communication type of the protocol that has been executed third, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the third protocol, "0" is stored. If the number of protocols executed is less than 3, "0" is stored. | 0, 1 to 16 | |
| (s3)+13 | Collation match Receive packet number 4 | If receiving is included in the communication type of the protocol that has been executed fourth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fourth protocol, "0" is stored. If the number of protocols executed is less than 4, "0" is stored. | 0, 1 to 16 | |
| (s3)+14 | Collation match Receive packet number 5 | If receiving is included in the communication type of the protocol that has been executed fifth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fifth protocol, "0" is stored. If the number of protocols executed is less than 5, "0" is stored. | 0, 1 to 16 | |
| (s3)+15 | Collation match Receive packet number 6 | If receiving is included in the communication type of the protocol that has been executed sixth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the sixth protocol, "0" is stored. If the number of protocols executed is less than 6, "0" is stored. | 0, 1 to 16 | |
| (s3)+16 | Collation match Receive packet number 7 | If receiving is included in the communication type of the protocol that has been executed seventh, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the seventh protocol, "0" is stored. If the number of protocols executed is less than 7, "0" is stored. | 0, 1 to 16 | |
| (s3)+17 | Collation match Receive packet number 8 | If receiving is included in the communication type of the protocol that has been executed eighth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the eighth protocol, "0" is stored. If the number of protocols executed is less than 8, "0" is stored. | 0, 1 to 16 | |

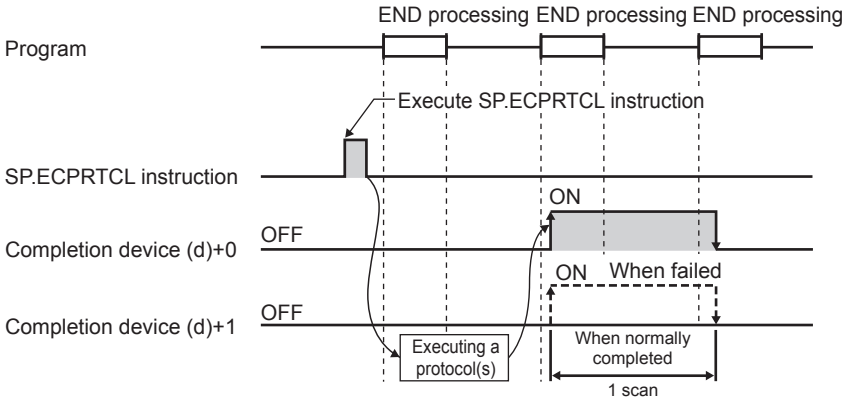
*1 System: The CPU module stores the execution result of the instruction.

## Processing details

This instruction executes the protocol registered using the engineering tool. Using the connection specified by (s1), the instruction executes the protocol in accordance with the control data stored in the device specified by (s3) and later. The instruction continuously executes as many protocols as specified by (s2) (a maximum of 8 protocols) at one time.
The number of executed protocols is stored in the device specified by (s3)+0.
The completion of the SP.ECPRTCL instruction can be checked using the completion devices (d)+0 and (d)+1.

- Completion device (d)+0: Turns ON during the END processing for the scan in which the SP.ECPRTCL instruction is completed, and turns OFF during the next END processing.
- Completion device (d)+1: Turns ON or OFF depending on the status when the SP.ECPRTCL instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the SP.ECPRTCL instruction is completed, and turns OFF during the next END processing. |




For details, refer to MELSEC iQ-F FX5 User's Manual (Ethernet Communication).

## Precautions

- If an error occurs in the mth protocol while multiple protocols are being executed, the instruction does not execute the "m+1"th protocol and after and is completed with an error.
- The connections for which the SP.ECPRTCL instruction can be executed are only those for which "Communication protocol" is specified for the communication means.
- If a cancel request is received during execution of the mth protocol while multiple protocols are executed continuously, the following is stored in (s3).

| Device | Item | Description |
|---|---|---|
| (s3)+0 | Resulting number of executed protocols | The executed protocol number. |
| (s3)+1 | Completion status | The error codes. |
| (s3)+10 | Collation match<br>Receive packet number 1 | The receive packet number successful in collation match for the already executed protocol. |
| ⋮ | ⋮ | |
| (s3)+m+8 | Collation match<br>Receive packet number m-1 | |

- If same instructions are executed for the same connection, the subsequent instruction is ignored and is not executed until the preceding instruction is completed.
- The SP.ECPRTCL instruction itself does not open/close a connection and therefore the SP.SOCOPEN/SP.SOCCLOSE instructions need to be used to open/close the connection.

☞ Refer to the Page 693 SP.SOCOPEN and Page 696 SP.SOCCLOSE

## Operation error

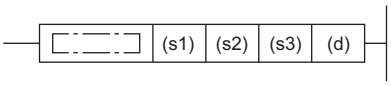
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device used exceeded the specified range. |
| 2821H | The device used to store data are overlapping. |
| 2822H | Device that cannot be specified is specified. |
| 3405H | The input data was out of range. |

# 11 PID CONTROL INSTRUCTION

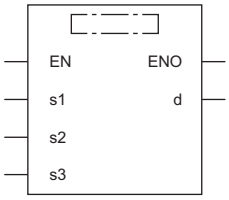
## 11.1 PID Control Loop

### PID

This instruction executes PID control which changes the output value according to the input variation.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PID(EN,s1,s2,s3,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Device number storing the target value (SV) | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Device number storing the measured value (PV) | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s3) | Device number storing a parameter | 1 to 32767 | 16-bit signed binary | ANY16 |
| (d) | Device number storing the output value (MV) | -32768 to +32767 | 16-bit signed binary | ANY16 |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | — | — | — | — | — | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | — | — | — | — | — | — | — | — |
| (s3) | — | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (d) | — | — | — | ○*1 | ○ | — | — | — | — | — | — | — | — |

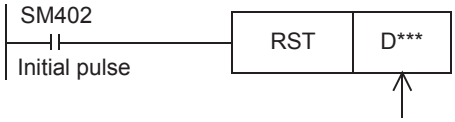
*1 Only D, SD, R can be used.

#### Processing details

When the target value (s1), measured value (s2), and parameters (s3) to (s3)+6 are set and a program is executed, the operation result (MV) is stored to the output value (d) at every sampling time (s3). For details, refer to the 📖MELSEC iQ-F FX5 User's Manual (Application).

## Precautions

- When auto tuning is not used, twenty-five devices are occupied from the head device specified in (s3).
- When auto tuning (limit cycle method) is used, twenty-nine devices are occupied from the head device specified in (s3).
- When auto tuning (step response method) is used, twenty-five devices are occupied from the head device specified in (s3).
- Two or more PID instructions can be executed at the same time. (There is no limitation in the number of loops.) However, make sure that (s3), (d) and other operands specified in each instruction are different to each other.
- For the output value (MV) in the PID instruction, specify a data register outside the latched area. When specifying a data register in the latched area, make sure to clear the latched (backed up) contents when the PLC mode is set to RUN using the following program.




Data register number in the latched area specified in (d).

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The connection number specified by (s1) is other than 1 to 8. |
| 2822H | The device number specified by (s2), (d), or (n) is outside the range of the number of device points. |
| 3500H | Incorrect sampling time (TS) (TS ≤ 0) |
| 3502H | Incompatible input filter constant (α) (α < 0 or 100 ≤ α) |
| 3503H | Incompatible proportional gain (KP) (KP ≤ 0) |
| 3504H | Incompatible integral time (TI) (TI < 0) |
| 3505H | Incompatible derivative gain (KD) (KD < 0 or 201 ≤ KD) |
| 3506H | Incompatible derivative time (TD) (TD < 0) |
| 350AH | Sampling time (TS) ≤ Scan time |
| 350CH | Variation of measured value exceeds limit. (△PV < -32768 or +32767 < △PV) |
| 350DH | Deviation exceeds limit. (EV < -32768 or +32767 < EV) |
| 350EH | Integral result exceeds limit. (Outside range from -32768 to +32767) |
| 350FH | Derivative value exceeds limit due to derivative gain (KD). |
| 3510H | Derivative result exceeds limit. (Outside range from -32768 to +32767) |
| 3511H | PID operation result exceeds limit. (Outside range from -32768 to +32767) |
| 3512H | PID output upper limit set value < PID output lower limit set value. |
| 3513H | Abnormal PID input variation alarm set value or output variation alarm set value. (Set value < 0) |
| 3514H | Step response method<br>Improper auto tuning result. |
| 3515H | Step response method<br>Auto tuning operation direction mismatch. |
| 3516H | Step response method<br>Improper auto tuning operation. |
| 3517H | Limit cycle method<br>Abnormal output set value for auto tuning [ULV (upper limit) ≤ LLV (lower limit)] |
| 3518H | Limit cycle method<br>Abnormal PV threshold (hysteresis) set value for auto tuning (SHPV < 0) |
| 3519H | Limit cycle method<br>Abnormal auto tuning transfer status (Data of device controlling transfer status is abnormally overwritten.) |
| 351AH | Limit cycle method<br>Abnormal result due to excessive auto tuning measurement time (τon > τ, τon < 0, τ < 0) |
| 351BH | Limit cycle method<br>Auto tuning result exceeds proportional gain. (KP = outside range from 1 to 32767) |
| 351CH | Limit cycle method<br>Auto tuning result exceeds integral time. (TI = outside range from 0 to 32767) |
| 351DH | Limit cycle method<br>Auto tuning result exceeds derivative time. (TD = outside range from 0 to 32767) |

# MEMO

# PART 4 MODULE DEDICATED INSTRUCTION

This part consists of the following chapters.

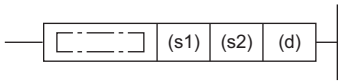
# 12 HIGH-SPEED COUNTER INSTRUCTION

## 12.1 High-speed Processing Instruction

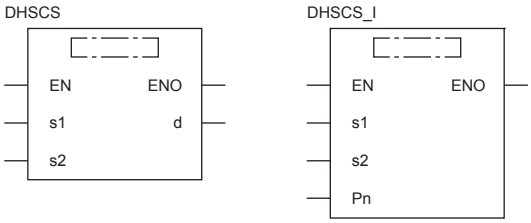
### Setting 32-bit data comparison

#### DHSCS

This instruction compares the value counted by a high-speed counter with a specified value, and immediately sets a bit device if the two values are equivalent to each other.

| Ladder diagram | Structured text*1 |
|---|---|
|  | ENO:=DHSCS(EN,s1,s2,d);<br>ENO:=DHSCS_I(EN,s1,s2,Pn); |

**FBD/LD*1**




*1 When the interrupt pointer (I) is specified in operand (d) by ST language and the FBD/LD language, use the DHSCS_I instruction.

#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|---|
| (s1) | | Data to be compared with the current value of a high-speed counter or word device number storing the data to be compared | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | | Channel number of a high-speed counter | K1 to 8 | 32-bit signed binary | ANY32 |
| (d) | DHSCS | Bit device number to be set to ON when the compared two values are equivalent to each other | — | Bit | ANY_BOOL |
| | DHSCS_I *1 | | | — | POINTER |

*1 In the case of the ST language and the FBD/LD language, d displays as Pn.

##### ■Applicable devices

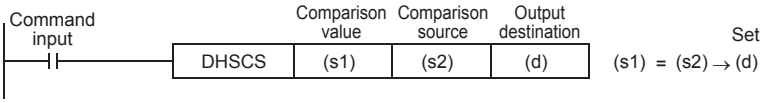
| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others (I) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○*1 | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | — | — | — | — | — | — | — | — | — | ○*2 |

*1 Enable the FX3 compatible function, and specify which devices between LC35 and 55 are to be designated as the FX3 compatible high-speed counter. For FX3 compatible function, refer to MELSEC iQ-F FX5 User's Manual (Application).

*2 I16 to I23 can be used.

## Processing details

- When the current value of a high-speed counter of the channel specified in (s2) becomes the comparison value (s1) (for example, when the current value changes from "199" to "200" or from "201" to "200" if the comparison value is K200), the bit device (d) is set to ON regardless of the scan time. In this instruction, the comparison processing is executed after the count processing in the high-speed counter. For details, refer to MELSEC iQ-F FX5 User's Manual (Application).




**Point**

Use DHSCS if the output should be given when the counting result becomes equivalent to the comparison value regardless of the scan time of the CPU module.
When the number of instructions that can be simultaneously used is exceeded, use a general-purpose comparison instruction.

## Precautions

The value specified in (s2) should only be the channel of high-speed counter number (1 to 8) set by the parameter. If a channel which is not set by the parameter or a value other than K1 to K8 is specified, an operation error occurs.
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Application).

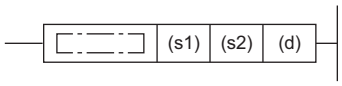
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3780H | The DHSCS, DHSCR, and DHSZ instructions are used exceeding the maximum limit of the number of these instructions. |
| 3405H | A channel number outside the range, the LC device, or the device (I) number is specified. |
| 3600H | A channel number for which the channel setting is not set is specified in the operand for channel number specification of the high-speed counter. |

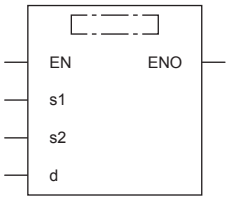
# Reset 32-bit data comparison

## DHSCR

This instruction compares the value counted by a high-speed counter with a specified value, and immediately resets a bit device if the two values are equivalent to each other, or resets the high speed counter.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHSCR(EN,s1,s2,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data to be compared with the current value of a high-speed counter or word device number storing the data to be compared | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Channel number of a high-speed counter | K1 to 8 | 32-bit signed binary | ANY32 |
| (d) | Bit device number to be reset (set to OFF) when both values become equivalent to each other, or channel number of self-reset high speed counter | — | Bit/32-bit signed binary | ANY_ELEMENTARY |

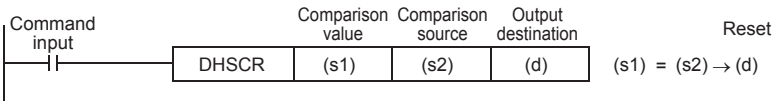
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○*1 | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | — | — | ○ | — | ○ | ○ | — | — | — |

*1 Enable the FX3 compatible function, and specify which devices between LC35 and 55 are to be designated as the FX3 compatible high-speed counter. For FX3 compatible function, refer to MELSEC iQ-F FX5 User's Manual (Application).

### Processing details

- When the current value of a high-speed counter of the channel specified in (s2) becomes the comparison value (s1) (for example, when the current value changes from "199" to "200" or from "201" to "200" if the comparison value is K200), the bit device (d) is reset to OFF regardless of the scan time. For details, refer to MELSEC iQ-F FX5 User's Manual (Application).




**Point**

Use DHSCR if the output should be given when the counting result becomes equivalent to the comparison value regardless of the scan time of the CPU module.
When the number of instructions that can be simultaneously used is exceeded, use a general-purpose comparison instruction.

## Precautions

The value specified in (s2) should only be the channel of high-speed counter number (1 to 8) set by the parameter. If a channel which is not set by the parameter or a value other than K1 to K8 is specified, an operation error occurs.
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Application).

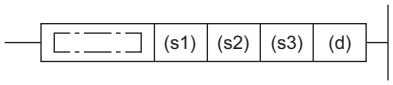
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3780H | The DHSCS, DHSCR, and DHSZ instructions are used exceeding the maximum limit of the in number of these instructions. |
| 3405H | A channel number outside the range or the LC device is specified. |
| 3600H | A channel number for which the channel setting is not set is specified in the operand in channel number specification of the high-speed counter. |

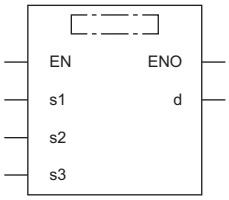
# Comparison of 32-bit data band

## DHSZ

This instruction compares the current value of a high-speed counter with two values (one zone), and outputs the comparison result (refresh).

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHSZ(EN,s1,s2,s3,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Data to be compared with the current value of a high-speed counter or word device number storing data to be compared (comparison value 1) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Data to be compared with the current value of a high-speed counter or word device number storing data to be compared (comparison value 2) | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s3) | Channel number of a high-speed counter or the device number of the current value of a high-speed counter | K1 to 8 | 32-bit signed binary | ANY32 |
| (d) | Head bit device number to which the comparison result is output based on the comparison value 1 and the comparison value 2 | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

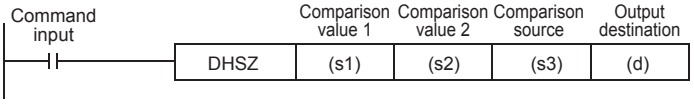
#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○*1 | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | — | — | — | — | — | — | — | — | — | — |

*1 Enable the FX3 compatible function, and specify which devices between LC35 and 55 are to be designated as the FX3 compatible high-speed counter. For FX3 compatible function, refer to MELSEC iQ-F FX5 User's Manual (Application).

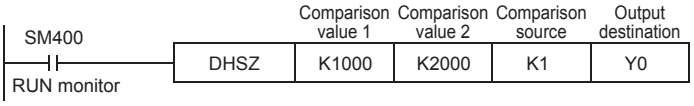
### Processing details

- The current value of a high-speed counter specified in (s3) is compared with two comparison points (comparison value 1 and comparison value 2). Based on the zone comparison result, "smaller than the lower comparison value", "inside the comparison zone" or "larger than the upper comparison value", one among (d), (d)+1 and (d)+2 is set to ON regardless of the scan time. For details, refer to MELSEC iQ-F FX5 User's Manual (Application).




| | Set |
|---|---|
| (s1) > (s3) | → (d) |
| (s1) ≤ (s3) ≤ (s2) | → (d) +1 |
| (s3) > (s2) | → (d) +2 |

- Make sure that the comparison value 1 and the comparison value 2 have the following relationship: [Comparison value 1] ≤ [Comparison value 2]. When the setting differs from the above, an operation error occurs and the DHSZ instruction will not perform any operation.
- When the current value of the high-speed counter CH1 changes (counts) as shown below, the comparison result is turn on to one of the outputs Y0, Y1 or Y2.




ON
K1000 > CH1 current value → Y0
K1000 ≤ CH1 current value ≤ K2000 → Y1
CH1 current value > K2000 → Y2

| Comparison pattern | Current value of CH1 (s3) | Change of output contact (Y) | | |
|---|---|---|---|---|
| | | Y0 | Y1 | Y2 |
| (s1) > (s3) | 1000>(s3) | ON | OFF | OFF |
| | 999→1000 | ON→OFF | OFF→ON | OFF |
| | 1000→999 | OFF→ON | ON→OFF | OFF |
| (s1) ≤ (s3) ≤ (s2) | 999→1000 | ON→OFF | OFF→ON | OFF |
| | 1000→999 | OFF→ON | ON→OFF | OFF |
| | 1000 ≤ (s3) ≤ 2000 | OFF | ON | OFF |
| | 2000→2001 | OFF | ON→OFF | OFF→ON |
| | 2001→2000 | OFF | OFF→ON | ON→OFF |
| (s3) > (s2) | 2000→2001 | OFF | ON→OFF | OFF→ON |
| | 2001→2000 | OFF | OFF→ON | ON→OFF |
| | (s3) > 2000 | OFF | OFF | ON |

**Point**

It is used when the output should be given when the counting result becomes equivalent to the comparison value regardless of the scan time of the CPU module.
When the number of instructions that can be simultaneously used is exceeded, use a general-purpose comparison instruction.

## Precautions

- If a channel which is not set to (s) by the parameter or a value other than K1 to 8 is specified, an operation error occurs.
- Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls. When designating a Y device, do not set a module or a device that crosses over a 16-fold device No.

Example: Designating Y36 or Y37 when using FX5U-64M□.

- For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Application).

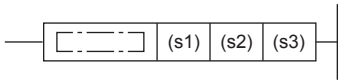
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3780H | The DHSCS, DHSCR, and DHSZ instructions are used exceeding the maximum limit of the number of these instructions. |
| 3405H | A channel number outside the range or the LC device is specified. |
| | The comparison value 1 is greater than the comparison value 2. |
| 2820H | The number of devices is insufficient. |
| 3600H | A channel number for which the channel setting is not set is specified in the operand for channel number specification of the high-speed counter. |

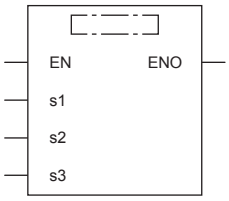
# Start/stop of the 16-bit data high-speed I/O function

## HIOEN(P)

These instructions control the start and stop operations of a high-speed I/O function.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=HIOEN(EN,s1,s2,s3);<br>ENO:=HIOENP(EN,s1,s2,s3); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Function number to be started or stopped | Refer to Function number (☞ Page 722) | 16-bit signed binary | ANY16 |
| (s2) | Set the bit of the channel number where the function is started. | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s3) | Set the bit of the channel number where the function is stopped. | -32768 to +32767 | 16-bit signed binary | ANY16 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

Specify the number of the function to be started or stopped in (s1), the bit of the channel to be started in (s2), and the bit of the channel to be stopped in (s3).

The following table shows the function numbers which can be specified in (s1).

#### ■Function number

| Function number | Function name |
|---|---|
| K0 | High-speed counter |
| K10*1 | Pulse density/rotation speed measurement |
| K20*1 | High-speed comparison table |
| K30*1*2 | Multi-output high-speed comparison table |
| K40 | Pulse width measurement |
| K50 | PWM |

*1 When high-speed counter (function number: K0) is stopped during function operation, the function continues to operate, but nothing will be processed.

*2 When multi-output high-speed comparison table (function number: K30) is stopped, high-speed counter of the same ch is also stopped.

The following table shows the values which can be specified in (s2) and (s3) for each function number.

• Function number K0

The counting start and stop of a high-speed counter can be controlled for each channel of high-speed counter.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | CH8 | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 |

Ex.

To start CH3, set 04H in (s2). To stop it, set 04H in (s3).

To start CH1, CH4, and CH5, set 19H in (s2). To stop them, set 19H in (s3).

To start CH1 and CH4 and to stop CH5, set 09H in (s2) and set 10H in (s3).

• Function number K10

The measuring start and stop of the pulse density (rotation speed measurement) can be controlled for each channel of the high-speed counter.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | CH8 | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 |

• Function number K20

Set the value to turn on the bit of the high-speed comparison table number which is to be started or stopped.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

• Function number K30

For the multi-output high-speed comparison table, specification of a channel is not required. To start the multi-output high-speed comparison table, set 01H in (s2). To stop it, set 01H in (s3).

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | | | | | | | | Valid |

• Function numbers K40 and K50

The measuring start and stop of pulse width measurement and PWM can be controlled for each channel.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | | | | | CH4 | CH3 | CH2 | CH1 |

## Precautions

- When values that turn on the same channel are set for start and stop, the stop operation is prioritized.
- To start the multi-output high-speed comparison table (function number: K30), the high-speed counter must be started using the HIOEN instruction in advance.
- When the high-speed comparison table is used with the HIOEN instruction, the total number of high-speed comparisons, including the DHSCS instruction, DHSCR instruction, DHSZ instruction, and interrupt input of built-in positioning, must be 32 or less.
- For the high-speed comparison table numbers and total number of high-speed comparisons, refer to ☞ Page 981 Adding and Changing Functions.

• The high-speed input/output instructions operate according to the following parameters.

| Function number | Function specified by the HIOEN instruction | Parameter setting |
|---|---|---|
| K0 | High-speed counter | Channel setting of the high-speed counter |
| K10 | Pulse density (rotation speed measurement) | Channel setting of the pulse density/rotation speed measurement<br>High-speed counter |
| K20 | High-speed comparison table | Output setting of the high-speed counter |
| K30 | Multi-output high-speed comparison table | Output setting of the high-speed counter |
| K40 | Pulse width measurement | Channel setting of the pulse width measurement |
| K50 | PWM | Channel setting of PWM |

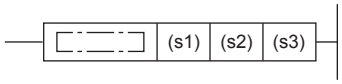
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | A channel number which is used in another instruction is specified. |
| 3405H | An invalid function number is specified in (s). |
| 3600H | A channel number which is not selected in the parameter setting is executed. |
| 3781H | When ring length ≤ preset value is specified and executed with a channel for which ring length is set and preset input is enabled. |

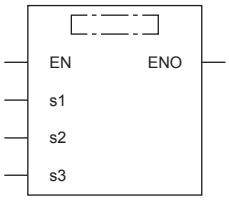
# Start/stop of the 32-bit data high-speed I/O function

## DHIOEN(P)

These instructions control the start and stop operations of a high-speed I/O function.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHIOEN(EN,s1,s2,s3);<br>ENO:=DHIOENP(EN,s1,s2,s3); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Function number to be started or stopped | Refer to Function number (☞ Page 725) | 16-bit signed binary | ANY16 |
| (s2) | Set the bit of the channel number where the function is started. | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s3) | Set the bit of the channel number where the function is stopped. | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |

### Processing details

Specify the number of the function to be started or stopped in (s1), the bit of the channel to be started in (s2), and the bit of the channel to be stopped in (s3).

The following table shows the function numbers which can be specified in (s1).

#### ■Function number

| Function number | Function name |
|---|---|
| K0 | High-speed counter |
| K10*1 | Pulse density/rotation speed measurement |
| K20*1 | High-speed comparison table |
| K30*1*2 | Multi-output high-speed comparison table |
| K40 | Pulse width measurement |
| K50 | PWM |

*1 When high-speed counter (function number: K0) is stopped during function operation, the function continues to operate, but nothing will be processed.

*2 When multi-output high-speed comparison table (function number: K30) is stopped, high-speed counter of the same ch is also stopped.

The following table shows the values which can be specified in (s2) and (s3) for each function number.

• Function number K0

The counting start and stop of a high-speed counter can be controlled for each channel of the high-speed counter.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | CH8 | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 |

Ex.

To start CH3, set 04H in (s2). To stop it, set 04H in (s3).

To start CH1, CH4, and CH5, set 19H in (s2). To stop them, set 19H in (s3).

To start CH1 and CH4 and to stop CH5, set 09H in (s2) and set 10H in (s3).

• Function number K10

The measuring start and stop of the pulse density (rotation speed measurement) can be controlled for each channel of the high-speed counter.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | CH8 | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 |

• Function number K20

Set the value to turn on the bit of the high-speed comparison table number which is to be started or stopped.

| Low-order bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

| High-order bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 | b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
| 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 |

• Function number K30

For the multi-output high-speed comparison table, specification of a channel is not required. To start the multi-output high-speed comparison table, set 01H in (s2). To stop it, set 01H in (s3).

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | | | | | | | | Valid |

• Function numbers K40 and K50

The measuring start and stop of pulse width measurement and PWM can be controlled for each channel.

| Bit position | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | | | | | | | | | | | | CH4 | CH3 | CH2 | CH1 |

## Precautions

- When the same channel is simultaneously turned on for start and stop, the stop operation is prioritized.
- When the high-speed comparison table is used with the DHIOEN instruction, the total number of high-speed comparisons, including the DHSCS instruction, DHSCR instruction, DHSZ instruction, and interrupt input of built-in positioning, must be 32 or less.
- For the high-speed comparison table numbers and total number of high-speed comparisons, refer to ☞ Page 981 Adding and Changing Functions.
- To start the multi-output high-speed comparison table (function number: K30), start the high-speed counter using the DHIOEN instruction in advance.
- The high-speed input/output instructions operate according to the following parameters.

| Function number | Function specified by the DHIOEN instruction | Parameter setting |
|---|---|---|
| K0 | High-speed counter | Channel setting of the high-speed counter |
| K10 | Pulse density (rotation speed measurement) | Channel setting of the pulse density/rotation speed measurement High-speed counter |
| K20 | High-speed comparison table | Output setting of the high-speed counter |
| K30 | Multi-output high-speed comparison table | Output setting of the high-speed counter |
| K40 | Pulse width measurement | Channel setting of the pulse width measurement |
| K50 | PWM | Channel setting of PWM |

## Operation error

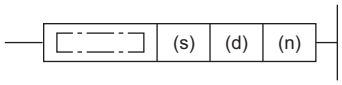
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | A channel number which is used in another instruction is specified. |
| 3405H | An invalid function number is specified in (s). |
| 3600H | A channel number which is not selected in the parameter setting is executed. |
| 3781H | When ring length ≤ preset value is specified and executed with a channel for which ring length is set and preset input is enabled. |

# 12.2 High-speed Current Value Transfer Instruction

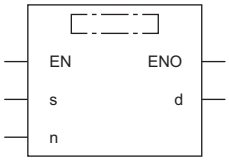
## High-speed current value transfer of 16-bit data

### HCMOV(P)

These instructions read and write (updates) special register for high-speed counter, pulse width measurement, PWM, and positioning.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=HCMOV(EN,s,n,d);<br>ENO:=HCMOVP(EN,s,n,d); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Transfer source device number | — | Bit/16-bit signed binary | ANY_ELEMENTARY |
| (d) | Transfer destination device number | — | Bit/16-bit signed binary | ANY_ELEMENTARY |
| (n) | Specification to clear the device value of the transfer source after the transfer | K0, K1 | 16-bit unsigned binary | ANY16_U |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

#### Processing details

These instructions transfer the data in the device specified by (s) to the device specified by (d). At this time, if the value of (n) is K0, the value of (s) is not cleared. If the value of (n) is K1, the value of (s) is cleared to "0" after the transfer. The value is cleared only for special devices (SD8099) supporting high-speed transfer.

**Point**

When (s) is a device supporting high-speed transfer

- When the HCMOV instruction is executed, the latest value is acquired such as the current value of a high-speed counter and transferred to (d).

When (d) is a device supporting high-speed transfer

- When the HCMOV instruction is executed, value such as the current value of a high-speed counter is changed.

### ■Effect of HCMOV instruction

- By using both input interrupt and HCMOV instruction, the current value of a high-speed counter can be received at the rising edge or falling edge of an external input.
- When HCMOV instruction is used just before a comparison instruction (CMP, ZCP or comparison contact instruction), the latest value of the high-speed counter is used in comparison.

## Precautions

- When it is necessary to execute comparison and outputting as soon as the current value of a high-speed counter changes, use the high-speed comparison table, multi-output high-speed comparison table, or one of the DHSCS, DHSCR, and DHSZ instructions.
- If 32-bit binary data special device which supports the high-speed transfer (such as the current value of a high-speed counter) is read using the HCMOV instruction, the operation is the same as that when the MOV instruction is used.
- Do not overwrite the current value of a high-speed counter using the HCMOV instruction while executing the pulse density (rotation speed measurement) or the SPD instruction.
- If (s) is SD8099, and the (n) value is K1, SD8099 is cleared at the timing the instruction is executed (after the SD8099 current value is transferred). Do not clear the SD8099 current value with an application instruction such as a MOV instruction as the scanning could be affected.

**Point**

The HCMOV instruction is mainly used to read the current value of the high-speed counter/pulse width measurement and change the current address (in the user units) or the current address (in the pulse unit) of positioning.

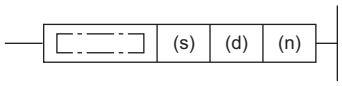
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | A value outside the data range is set in (n). |
| 2821H | When an operand that executes transmission between an SM supporting high-speed transfer and an SD supporting high-speed transfer is designated. |

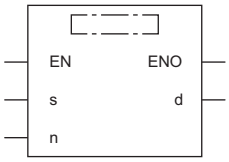
# High-speed current value transfer of 32-bit data

## DHCMOV(P)

These instructions read and write (updates) special register for high-speed counter, pulse width measurement, PWM, and positioning.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DHCMOV(EN,s,n,d);<br>ENO:=DHCMOVP(EN,s,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Transfer source device number | — | Bit/32-bit signed binary | ANY_ELEMENTARY |
| (d) | Transfer source device number | — | Bit/32-bit signed binary | ANY_ELEMENTARY |
| (n) | Specification to clear the device value of the transfer source after the transfer | K0, K1 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

These instructions transfer the data in the device specified by (s) to the device specified by (d). At this time, if the value of (n) is K0, the value of (s) is not cleared. If the value of (n) is K1, the value of (s) is cleared to "0" after the transfer. The value is cleared only for SD devices of the current value of the high-speed counter or LC devices used as a high-speed counter when using the FX3 compatible high-speed counter.

**Point**

When (s) is a device supporting high-speed transfer

- When the DHCMOV instruction is executed, the latest value is acquired such as the current value of a high-speed counter and transferred to (d).

When (d) is a device supporting high-speed transfer

- When the DHCMOV instruction is executed, value such as the current value of a high-speed counter is changed.

#### ■Effect of DHCMOV instruction

- By using both input interrupt and DHCMOV instruction, the current value of a high-speed counter can be received at the rising edge or falling edge of an external input.
- When DHCMOV instruction is used just before a comparison instruction (DCMP, DZCP or comparison contact instruction), the latest value of the high-speed counter is used in comparison.

## Precautions

- When it is necessary to execute comparison and outputting as soon as the current value of a high-speed counter changes, use the high-speed comparison table, multi-output high-speed comparison table, or one of the DHSCS, DHSCR, and DHSZ instructions.
- Do not overwrite the current value of a high-speed counter using the DHCMOV instruction while executing the pulse density (rotation speed measurement) or the DSPD instruction.
- Transfer is not possible between an SM supporting high-speed transfer and an SD supporting high-speed transfer.
- When the device supporting high-speed transfer is set as the transfer source (s) by the DHCMOV instruction while the high-speed I/O function is stopped, the previous value before stop is read out. However, if the function is not executed even once, the initial value is read out.



Ex.

SD5303, SD5302 (PWM pulse width) is set as the transfer source (s), the operation is executed as follows.

| | |
|---|---|
| When the PWM function is not executed | "0" is read out. (This is not the value of the parameter that is set by the GX Works3.) |
| When the PWM function was executed but it is currently stopped | The value when the PWM function was stopped is read out. |
| When the PWM function is executed | The latest value that is currently operating is read out. |

- When the high-speed counter's SD devices (current value, maximum value, minimum value) are read out individually, only the read SD device will be updated. Thus, there may be cases when the high-speed counter's SD device does not establish the relation of minimum value ≤ current value ≤ maximum value temporarily. Refer to the MELSEC iQ-F FX5 User's Manual (Application) for details on the timing that the high-speed counter's SD device is updated.

**Point**

The DHCMOV instruction is mainly used to read the current value of the high-speed counter/pulse width measurement and change the current address (in the user units) or the current address (in the pulse unit) of positioning.

## Operation error

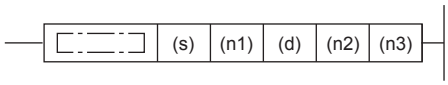
| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | A value outside the data range is set in (n). |
| 2821H | When an operand that executes transmission between an SM supporting high-speed transfer and an SD supporting high-speed transfer is designated. |

# 13 EXTERNAL DEVICE COMMUNICATION INSTRUCTION

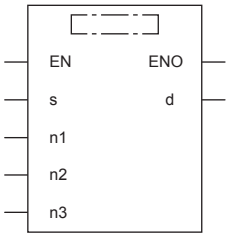
## 13.1 Serial Communication 2

### RS2

This instruction sends or receives data by non-protocol communication via serial ports of RS-232C or RS-485.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RS2(EN,s,n1,n2,n3,d); |

**FBD/LD**




#### Setting data

**■Descriptions, ranges, and data types**

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device storing send data | — | 16-bit signed binary/ character string | ANY16 |
| (n1) | Number of bytes of data to be sent | 0 to 4096 | 16-bit unsigned binary | ANY16_U |
| (d) | Head device storing receive data | — | 16-bit signed binary/ character string | ANY16 |
| (n2) | Number of bytes to be received | 0 to 4096 | 16-bit unsigned binary | ANY16_U |
| (n3) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |

**■Applicable devices**

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○*1 | — | — | — | — | ○ | — | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n3) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |

*1 T, ST, C cannot be used.

#### Processing details

This instruction sends or receives data by non-protocol communication via the built-in RS-485 port or the optional RS-232C or RS-485 serial ports installed in the CPU module. This instruction specifies the head device storing the sent data from the CPU module, the number of data bytes, the head device storing the received data, and the maximum number of bytes that can be received. For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

## Precautions

- It is not permitted to use instructions for external device communication instruction on the same port.
- While this instruction is being driven, the communication format cannot be changed. Set this instruction to OFF before changing the communication format.
- When using the header and terminator, set them before driving this instruction. Do not change the values of the header and terminator while this instruction is being driven.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | Data outside the allowable range was input. |
| 2820H | The device specified by (s) and (d) exceeds the corresponding device range. |
| 1810H | Channel number which is used in another instruction is specified. |
| 3600H | Channel number specified by (n3) is not set by parameters. |

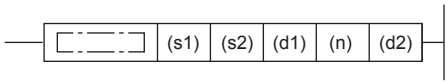
For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

# 13.2 Inverter Communication Instruction

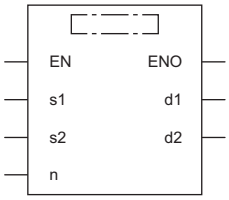
## Inverter operation monitoring(Status check)

### IVCK

This instruction reads the operation status of an inverter to the CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVCK(EN,s1,s2,n,d1,d2); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Inverter instruction codes | *1 | 16-bit signed binary | ANY16 |
| (d1) | Device number storing the read value | — | 16-bit signed binary | ANY16 |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d2) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication) or respective inverter manual.

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

#### Processing details

The operation status corresponding to the instruction code specified in (s2) of an inverter connected to the communication channel (n) whose station number is specified in (s1) is read and transferred to (d1). For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication). (For the instruction codes, refer to the each inverter manual.)

#### Precautions

Three devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

## Operation error

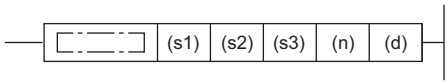
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

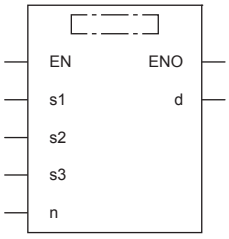
# Inverter operations control(Drive)

## IVDR

This instruction writes a control value necessary for inverter operation to a CPU module using the computer link operation function of the inverter.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVDR(EN,s1,s2,s3,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Inverter instruction codes | *1 | 16-bit signed binary | ANY16 |
| (s3) | Set value to be written to the inverter parameter or device number storing the data to be set | — | 16-bit signed binary | ANY16 |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication) or respective inverter manual.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | — | — | — | ○ | ○ | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

The control value specified in (s3) is written to the instruction code specified in (s2) of an inverter connected to the communication channel (n) whose station number is specified in (s1). For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication). (For the instruction codes, refer to the each inverter manual.)

### Precautions

Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

## Operation error

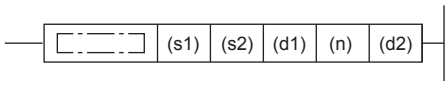
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

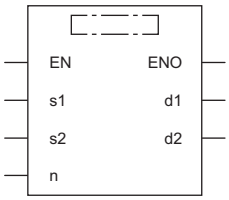
# Inverter parameter read

## IVRD

This instruction reads a parameter of an inverter to the CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVRD(EN,s1,s2,n,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Inverter parameter number | *1 | 16-bit signed binary | ANY16 |
| (d1) | Device number storing the read value | — | 16-bit signed binary | ANY16 |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d2) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication) or respective inverter manual.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○ | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

The value of the parameter (s2) is read from an inverter connected to the communication channel (n) whose station number is (s1), and output to (d1). For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication). (For the parameter numbers, refer to the each inverter manual.)

### Precautions

Three devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

## Operation error

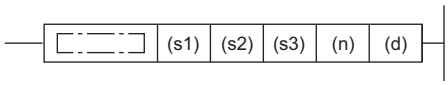
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (s2) is outside the allowable range. (Less than K0, K3000 to 9999, or K13000 to 32767) |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

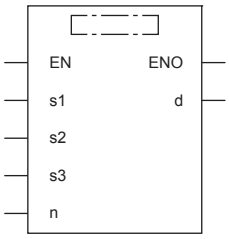
# Inverter parameter write

## IVWR

This instruction writes a parameter of an inverter from the CPU module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVWR(EN,s1,s2,s3,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Inverter parameter number | *1 | 16-bit signed binary | ANY16 |
| (s3) | Set value to be written to the inverter parameter or device number storing the data to be set | — | 16-bit signed binary | ANY16 |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication) or respective inverter manual.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | ○ | — | — | ○ | ○ | — | — | — | ○ | ○ | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

A value specified in (s3) is written to a parameter (s2) in an inverter connected to the communication channel (n) whose station number is (s1). For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication). (For the parameter numbers, refer to the each inverter manual.)

### Precautions

Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

## Operation error

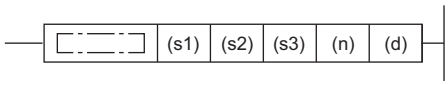
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (s2) is outside the allowable range. (Less than K0, K3000 to 9999, or K13000 to 32767) |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

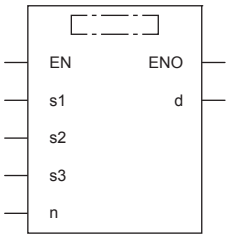
# Inverter parameter block write

## IVBWR

This instruction writes parameters of an inverter from the CPU module in a batch.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVBWR(EN,s1,s2,s3,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Number of parameters in an inverter to be written at one time | *1 | 16-bit signed binary | ANY16 |
| (s3) | Start device of a parameter table to be written to an inverter | — | 16-bit signed binary | ANY16 |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication) or respective inverter manual.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

A data table specified in (s2) and (s3) is written to an inverter connected to the communication channel (n) whose station number is (s1) in batch. For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication). (For the parameter numbers, refer to the each inverter manual.)

### Precautions

Three devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (s2) is K0 or less. |
| | The value specified by (s3) is outside the allowable range. (Less than K0, K3000 to 9999, or K13000 to 32767) |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

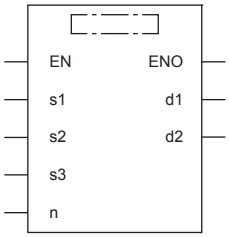
# Inverter multi command

## IVMC

This instruction writes 2 types of settings (operation command and set frequency) to the inverter, and reads 2 types of data (inverter status monitor, output frequency, etc.) from the inverter at the same time.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=IVMC(EN,s1,s2,s3,n,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Inverter station number | K0 to 31 | 16-bit signed binary | ANY16 |
| (s2) | Multiple instructions for inverter: Send/receive data type specification | *1 | 16-bit signed binary | ANY16 |
| (s3) | Head device which stores data to be written to the inverter | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 2) |
| (d1) | Head device which stores values to be read from the inverter | — | 16-bit signed binary | ANY16_ARRAY (Number of elements: 2) |
| (n) | Communication channel | K1 to 4 | 16-bit unsigned binary | ANY16_U |
| (d2) | Head bit device to which the execution status of the instruction is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

*1 Refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (d1) | — | — | — | ○ | ○ | — | — | — | ○ | — | — | — | — |
| (n) | — | — | — | — | — | — | — | — | — | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes multiple commands of an inverter connected to the communication channel (n) whose station number is specified in (s1). Specify the send/receive data type using (s2), the head device which stores data to be written to the inverter using (s3), and the head device which stores values to be read from the inverter using (d1). For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

## Precautions

- If a device number outside the range due to indexing, etc. is specified in (d1), the receive data from the inverter is not stored in (d1). However, values set in (s3) and (s3)+1 may be written to the inverter.
- If any unspecified value is set in (s2), unexpected data may be written to and read from the inverter, and values of (d1) and (d1)+1 may be updated.
- The IVMC instruction reads the inverter status at the time of communication with the inverter, and stores it in (d1). Accordingly, the inverter status written by the IVMC instruction can be read when the next reading instruction (IVCK, IVMC, etc.) is executed.
- Two devices are occupied from the device specified in (s3) and (d1). Make sure that these devices are not used in other controls.
- Three devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

## Operation error

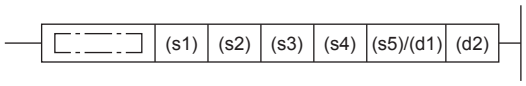
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel number specified by (d) is used by another instruction. |
| 2820H | The specified device exceeds the range of the corresponding device. |
| 3405H | The value specified by (s1) is other than any of K0 to 31. |
| | The value specified by (n) is other than any of K1 to 4. |
| 3600H | Channel number specified by (d) is not set by parameters. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

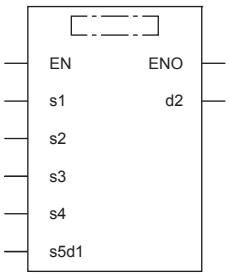
# 13.3 MODBUS Communication Instruction

## ADPRW

This instruction allows the MODBUS Master to communicate (read/write data) with the Slaves.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=ADPRW(EN,s1,s2,s3,s4,s5d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Slave node address | 0 to 20H | 16-bit signed binary | ANY16 |
| (s2) | Function code | 01H to 06H, 0FH, 10H | 16-bit signed binary | ANY16 |
| (s3) | Function parameters depending on the function code | 0 to FFFFH | 16-bit signed binary | ANY16 |
| (s4) | Function parameters depending on the function code | 1 to 2000 | 16-bit signed binary | ANY16 |
| (s5)/(d1) | Function parameters depending on the function code | — | Bit/16-bit signed binary | ANY_ELEMENTARY |
| (d2) | Head bit device number to which the execution status of the communication is output | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s3) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s4) | — | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s5)/(d1) | ○ | — | — | ○*1 | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

- Function code (s2) is operated on Slave node address (s1) according to Parameters (s3), (s4), and (s5)/(d1). Use 0 as the Slave Node Address for Broadcast commands. For details, refer to MELSEC iQ-F FX5 User's Manual (MODBUS Communication).
- The communication execution status (d2) is output according to the status of the ADPRW instruction such as communicating/completed normally/completed with an error.

## Precautions

Three devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

## Operation error

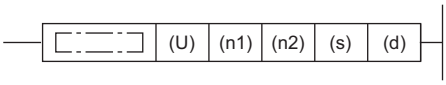
| Error code (SD0/SD8067) | Description |
|---|---|
| 1810H | Channel used by the instruction is used by other instruction. |
| 3600H | Invalid parameter setup. |
| 2822H | Device that cannot be used by this instruction is specified. |
| 3405H | Data outside the allowable range was input. |
| 2820H | The specified device exceeds the range of the corresponding device. |

For communication errors, refer to MELSEC iQ-F FX5 User's Manual (MODBUS Communication).

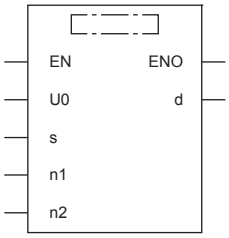
# 13.4 Predefined Protocol Support Function Instruction

## S(P).CPRTCL

This instruction executes the communication protocol registered using the engineering tool.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=S_CPRTCL(EN,U0,s,n1,n2,d);<br>ENO:=SP_CPRTCL(EN,U0,s,n1,n2,d); |

**FBD/LD**




("S_CPRTCL", "SP_CPRTCL" enters □.)

### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U)*1 | Dummy (Input the character string ['U0'].) | — | Character string | ANYSTRING_SINGLE |
| (n1) | Communication channel | 1 to 4 | 16-bit unsigned binary | ANY16_U |
| (n2) | Number of protocols to be executed continuously | 1 to 8 | 16-bit unsigned binary | ANY16_U |
| (s) | Head device number for storing the control data | Refer to Control data (☞ Page 748) | Word | ANY16_ARRAY (Number of elements: 18) |
| (d) | Head device number which turns ON when the execution of the instruction is completed and remains on for 1 scan. If the instruction is completed with an error, (d)+1 is also turned on. | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

*1 In the case of the ST language and the FBD/LD language, U displays as U0.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U) | — | — | — | ○ | — | — | — | — | ○ | — | — | ○ | — |
| (n1) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (n2) | — | — | — | ○ | — | — | — | — | ○ | ○ | — | — | — |
| (s) | — | — | — | ○ | — | — | — | — | ○ | — | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

#### ■Control data

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s)+0 | Completion status | The completion status is stored upon completion of the instruction.<br>When two or more protocols are executed, the execution result of the protocol executed last is stored.<br>• 0: Normal completion<br>• Other than 0: Error completion (error code) | — | System |
| (s)+1 | Resulting number of executed protocols | The number of protocols executed by the SP.ECPRTCL instruction is stored.<br>Any protocol where an error occurred is also included in the execution number. If the setting of setting data or control data contains an error, "0" is stored. | 0, 1 to 8 | |

| Device | Item | Description | Setting range | Set by*1 |
|---|---|---|---|---|
| (s)+2 | Execution protocol number 1 | Specify the number of the protocol to be executed first. | 1 to 64 | User |
| (s)+3 | Execution protocol number 2 | Specify the number of the protocol to be executed second. | 0, 1 to 64 | |
| (s)+4 | Execution protocol number 3 | Specify the number of the protocol to be executed third. | 0, 1 to 64 | |
| (s)+5 | Execution protocol number 4 | Specify the number of the protocol to be executed fourth. | 0, 1 to 64 | |
| (s)+6 | Execution protocol number 5 | Specify the number of the protocol to be executed fifth. | 0, 1 to 64 | |
| (s)+7 | Execution protocol number 6 | Specify the number of the protocol to be executed sixth. | 0, 1 to 64 | |
| (s)+8 | Execution protocol number 7 | Specify the number of the protocol to be executed seventh. | 0, 1 to 64 | |
| (s)+9 | Execution protocol number 8 | Specify the number of the protocol to be executed eighth. | 0, 1 to 64 | |
| (s)+10 | Collation match Receive packet number 1 | If receiving is included in the communication type of the protocol that has been executed first, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the first protocol, "0" is stored. | 0, 1 to 16 | System |
| (s)+11 | Collation match Receive packet number 2 | If receiving is included in the communication type of the protocol that has been executed second, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the second protocol, "0" is stored. If the number of protocols executed is less than 2, "0" is stored. | 0, 1 to 16 | |
| (s)+12 | Collation match Receive packet number 3 | If receiving is included in the communication type of the protocol that has been executed third, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the third protocol, "0" is stored. If the number of protocols executed is less than 3, "0" is stored. | 0, 1 to 16 | |
| (s)+13 | Collation match Receive packet number 4 | If receiving is included in the communication type of the protocol that has been executed fourth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fourth protocol, "0" is stored. If the number of protocols executed is less than 4, "0" is stored. | 0, 1 to 16 | |
| (s)+14 | Collation match Receive packet number 5 | If receiving is included in the communication type of the protocol that has been executed fifth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fifth protocol, "0" is stored. If the number of protocols executed is less than 5, "0" is stored. | 0, 1 to 16 | |
| (s)+15 | Collation match Receive packet number 6 | If receiving is included in the communication type of the protocol that has been executed sixth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the sixth protocol, "0" is stored. If the number of protocols executed is less than 6, "0" is stored. | 0, 1 to 16 | |
| (s)+16 | Collation match Receive packet number 7 | If receiving is included in the communication type of the protocol that has been executed seventh, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the seventh protocol, "0" is stored. If the number of protocols executed is less than 7, "0" is stored. | 0, 1 to 16 | |
| (s)+17 | Collation match Receive packet number 8 | If receiving is included in the communication type of the protocol that has been executed eighth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the eighth protocol, "0" is stored. If the number of protocols executed is less than 8, "0" is stored. | 0, 1 to 16 | |

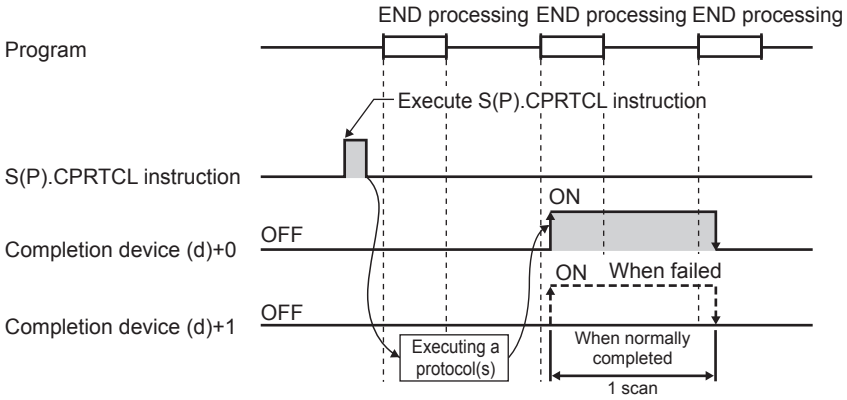
*1 User: Data to be set before the execution of the instruction. System: The CPU module stores the execution result of the instruction.

## Processing details

This instruction executes the protocol registered using the engineering tool. Using CH specified by (n1), the instruction executes the protocol in accordance with the control data stored in the device specified by (s) and later. The instruction continuously executes as many protocols as specified by (n2) (a maximum of 8 protocols) at one time.
The number of executed protocols is stored in the device specified by (s)+1.
The completion of the S(P).CPRTCL instruction can be checked using the completion devices (d)+0 and (d)+1.

- Completion device (d)+0: Turns ON during the END processing for the scan in which the S(P).CPRTCL instruction is completed, and turns OFF during the next END processing.
- Completion device (d)+1: Turns ON or OFF depending on the status when the S(P).CPRTCL instruction is completed.

| Status | Description |
|---|---|
| When completed normally | The device does not change (remains OFF). |
| When completed with an error | The device turns ON during the END processing for the scan in which the S(P).CPRTCL instruction is completed, and turns OFF during the next END processing. |




For details, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

## Precautions

- If an error occurs in the mth protocol while multiple protocols are being executed, the instruction does not execute the "m+1"th protocol and after and is completed with an error.
- The communication CH for which the S(P).CPRTCL instruction can be executed are only those for which "Predefined protocol support function" is specified for the communication protocol.
- If a cancel request is received during execution of the mth protocol while multiple protocols are executed continuously, the following is stored in (s).

| Device | Item | Description |
|---|---|---|
| (s)+0 | Completion status | The error codes. |
| (s)+1 | Resulting number of executed protocols | The executed protocol number. |
| (s)+10 | Collation match<br>Receive packet number 1 | The receive packet number successful in collation match for the already executed protocol. |
| ⋮ | ⋮ | |
| (s)+m+8 | Collation match<br>Receive packet number m-1 | |

- If same instructions are executed for the same CH, the subsequent instruction is ignored and is not executed until the preceding instruction is completed.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Serial Communication).

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | The device used exceeded the specified range. |
| 2821H | The device used to store data are overlapping. |
| 2822H | Device that cannot be specified is specified. |
| 3405H | The input data was out of range. |

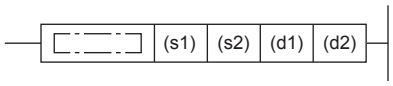
# 14 POSITIONING INSTRUCTION

## 14.1 Positioning Instruction

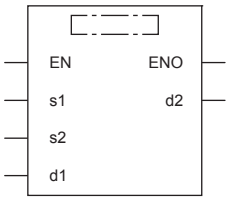
## Zero return(OPR) with 16-bit data DOG search

### DSZR [For the FX3 compatible operand specification]

This instruction executes mechanical zero return.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSZR(EN,s1,s2,d1,d2); |

**FBD/LD**




#### Setting data

##### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Bit device number to which the near-point dog signal is input | — | Bit | ANY_ELEMENTARY (BOOL) |
| (s2) | Bit device number to which the zero-phase signal is input | — | Bit | ANY_ELEMENTARY (BOOL) |
| (d1) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

##### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (s2) | ○*1*2 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d1) | ○*3 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*4 | — | — | — | — | — | — | — | — | — | — | — | — |

*1 When using X, always specify a device that has been set by parameter.
*2 Specify the device set with a parameter or same as the one set in (s1).
*3 Only Y can be used.
*4 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

## Processing details

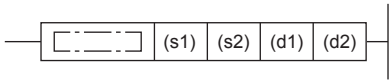
This instruction executes mechanical zero return. The values of special devices are applied as the zero return speed and creep speed. With the forward limit or reverse limit, zero return with the dog search function can be executed.

- For (s1), specify the near-point dog signal input device number. When an X device is specified, the near-point dog signal functions follow the logic set by parameter. When other than X device is specified, the device functions follow the positive logic.
- For (s2), specify the zero-phase signal input device number. When an X device is specified, the zero-phase signal functions follow the logic set by parameter. When other than X device is specified, the device functions follow the positive logic.
- For (d1), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (d2), specify the bit device from which the rotation direction signal is output. Only the device specified with a parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

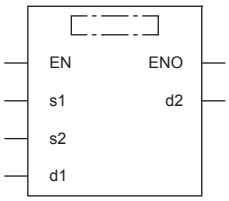
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DSZR [For the FX5 operand specification]

This instruction executes mechanical zero return.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DSZR(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Zero return speed | 1 to 65535 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (s2) | Creep speed | 1 to 65535 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the zero return complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes mechanical zero return. The near-point dog signal and zero-phase signal function follow the device set with parameters. With the forward limit or reverse limit, zero return with the dog search function can be executed.

- For (s1), specify the zero return speed in the user units. (The speed must be 200 Kpps or lower in frequency.)
- For (s2), specify the creep speed in the user units. Set the creep speed equal to or slower than the zero return speed set in (s1). (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number for which zero return is performed.
- For (d2), specify the bit device of the zero return complete flag or abnormal end flag.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

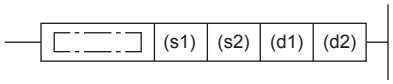
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

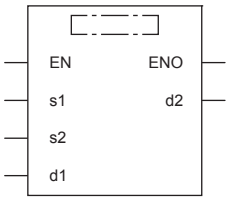
# Zero return(OPR) with 32-bit data DOG search

## DDSZR

This instruction executes mechanical zero return.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDSZR(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Zero return speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Creep speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY16 |
| (d2) | Bit device number of the zero return complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes mechanical zero return. The near-point dog signal and zero-phase signal function follow the device set with parameters. With the forward limit or reverse limit, zero return with the dog search function can be executed.

- For (s1), specify the zero return speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (s2), specify the creep speed in user units. Set the creep speed equal to or slower than the zero return speed set in (s1). (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number for which zero return is performed.
- For (d2), specify the bit device of the zero return complete flag or abnormal end flag.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

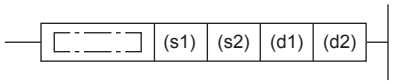
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

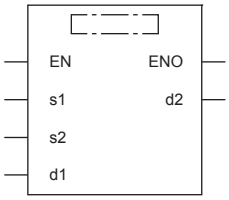
# 16-bit data interrupt positioning

## DVIT [For the FX3 compatible operand specification]

This instruction executes interrupt 1-speed constant quantity feed.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DVIT(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address after an interrupt input | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

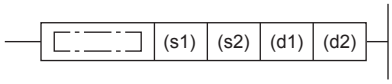
This instruction executes interrupt 1-speed constant quantity feed.

- For (s1), specify the transfer distance that is output after an interrupt, in user units. (The distance must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (d2), specify the device from which the rotation direction signal is output. Only the device specified with a parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

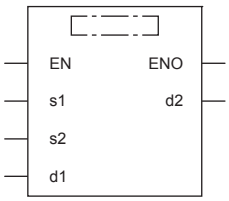
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DVIT [For the FX5 operand specification]

This instruction executes interrupt 1-speed constant quantity feed.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DVIT(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address after an interrupt input | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Specify the axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes interrupt 1-speed constant quantity feed.

- For (s1), specify the transfer distance that is output after an interrupt, in user units. (The distance must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DVIT instruction.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

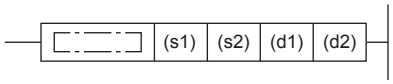
### Precautions

Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

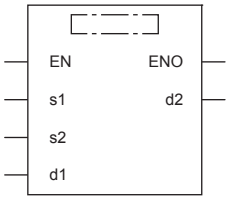
# 32-bit data interrupt positioning

## DDVIT [For the FX3 compatible operand specification]

This instruction executes interrupt 1-speed constant quantity feed.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDVIT(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address after an interrupt input | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

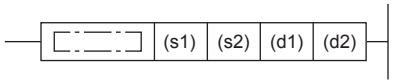
This instruction executes interrupt 1-speed constant quantity feed.

- For (s1), specify the transfer distance that is output after an interrupt, in user units. (The distance must be within the range of -2147483647 to +2147483647 in the number of pulses.)
- For (s2), specify the speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (d2), specify the device from which the rotation direction signal is output. Only the device specified with a parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

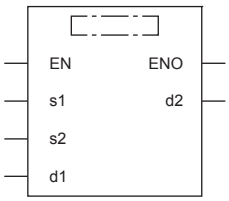
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DDVIT [For the FX5 operand specification]

This instruction executes interrupt 1-speed constant quantity feed.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDVIT(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address after an interrupt input | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Specify the axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes interrupt 1-speed constant quantity feed.

- For (s1), specify the transfer distance that is output after an interrupt, in user units. (The distance must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DDVIT instruction.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

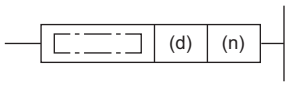
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

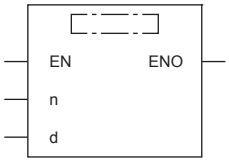
# Positioning by one table operation

## TBL [For the FX3 compatible operand specification]

This instruction executes one specified table operation from the instructions set in the data table using the engineering tool etc.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TBL(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (n) | Table number to be executed | 1 to 100 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

*1 Only Y can be used.

### Processing details

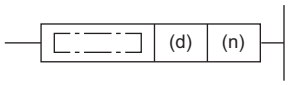
This instruction operates one table of the positioning table that is set with parameters in the engineering tool.

- For (d), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (n), specify the table number to be executed according to the output specified in (d).

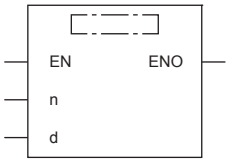
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## TBL [For the FX5 operand specification]

This instruction executes one specified table operation from the instructions set in the data table using the engineering tool etc.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=TBL(EN,n,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (n) | Table number to be executed | 1 to 100 | 16-bit unsigned binary | ANY16_U |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

### Processing details

This instruction operates one table of the positioning table that is set with parameters in the engineering tool.

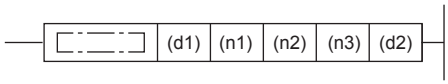
- For (d), specify the axis number from which pulses are output.
- For (n), specify the table number to be executed according to the output specified in (d).

For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

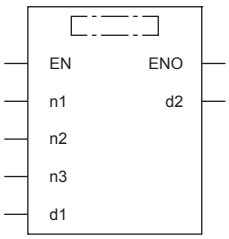
# Positioning by multiple table operation

## DRVTBL

This instruction executes positioning operation set in multiple data tables with the engineering tool in continuous operation or stepping operation. To execute such operation, this instruction needs to be executed only once.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVTBL(EN,n1,n2,n3,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY16 |
| (n1) | Start table number to be executed | 1 to 100 | 16-bit unsigned binary | ANY16_U |
| (n2) | Last table number to be executed | 1 to 100 | 16-bit unsigned binary | ANY16_U |
| (n3) | Table execution method | K0, K1 | 16-bit unsigned binary | ANY16_U |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANYBIT_ARRAY (Number of elements: 2) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (d1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes positioning operation set in multiple data tables with the engineering tool in the continuous operation or stepping operation. To execute such operation, this instruction needs to be executed only once.

- For (d1), specify the axis number from which pulses are output.
- For (n1), specify the start table to be executed according to the output specified in (d1).
- For (n2), specify the last table. When (n1) and (n2) are the same, only one table is executed. The table operation keeps executing until the last table or a table that is not set with parameters is executed.
- For (n3), specify the table execution method. (K0 = Stepping operation, K1 = Continuous operation)
- For (d2), specify the bit device of the normal complete flag or abnormal end flag.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

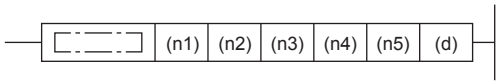
## Precautions

Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

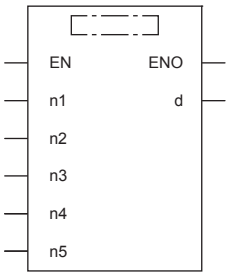
# Multiple axes concurrent drive positioning

## DRVMUL

This instruction executes tables of multiple axes of one module simultaneously.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVMUL(EN,n1,n2,n3,n4,n5,d); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (n1) | Start axis number | K1 | 16-bit unsigned binary | ANY16_U |
| (n2) | Table number of the axis 1 | K0 to 100 | 16-bit unsigned binary | ANY16_U |
| (n3) | Table number of the axis 2 | K0 to 100 | 16-bit unsigned binary | ANY16_U |
| (n4) | Table number of the axis 3 | K0 to 100 | 16-bit unsigned binary | ANY16_U |
| (n5) | Table number of the axis 4 | K0 to 100 | 16-bit unsigned binary | ANY16_U |
| (d) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANYBIT_ARRAY (Number of elements: 8) |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n3) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n4) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n5) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

This instruction executes tables of multiple axes of one module simultaneously.

- For (n1), specify the start axis number. When the built-in positioning function of the CPU module is used, the start axis is the axis 1. Thus, specify K1.
- For (n2), specify the table number that is executed with the axis (n1). When not executing the axis (n1), specify K0.
- For (n3), specify the table number that is executed with the axis (n1)+1. When not executing the axis (n1)+1, specify K0.
- For (n4), specify the table number that is executed with the axis (n1)+2. When not executing the axis (n1)+2, specify K0.
- For (n5), specify the table number that is executed with the axis (n1)+3. When not executing the axis (n1)+3, specify K0.
- For (d), specify the device of the instruction execution complete flag for each axis. Eight devices are occupied from (d), and function as follows.

| Device | Description |
|---|---|
| (d) | Instruction execution complete flag for the axis (n1) |
| (d)+1 | Instruction execution abnormal end flag for the axis (n1) |
| (d)+2 | Instruction execution complete flag for the axis (n1)+1 |
| (d)+3 | Instruction execution abnormal end flag for the axis (n1)+1 |
| (d)+4 | Instruction execution complete flag for the axis (n1)+2 |
| (d)+5 | Instruction execution abnormal end flag for the axis (n1)+2 |
| (d)+6 | Instruction execution complete flag for the axis (n1)+3 |
| (d)+7 | Instruction execution abnormal end flag for the axis (n1)+3 |

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

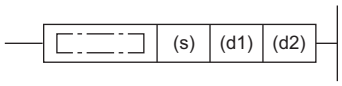
## Precautions

Eight devices are occupied from the device specified in (d). Make sure that these devices are not used in other controls.
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

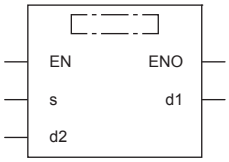
# 32-bit data ABS current value read

## DABS

This instruction reads the absolute position (ABS) data when a servo amplifier (equipped with the absolute position detection function) is connected. The data is converted into pulse when read.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DABS(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Head device number that inputs the output signal for absolute position (ABS) data from the servo amplifier | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |
| (d1) | Head device number that outputs the absolute position (ABS) data control signal to the servo amplifier | — | Bit | ANYBIT_ARRAY (Number of elements: 3) |
| (d2) | Absolute position (ABS) data (32-bit value) | — | 32-bit signed binary | ANY32 |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (d1) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |
| (d2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction reads the absolute position (ABS) data when a servo amplifier (equipped with the absolute position detection function) is connected. The data is converted into pulse when being read.

- For (s), specify the head device number that inputs the output signal for absolute position (ABS) data from the servo amplifier.
- For (d1), specify the head device number that outputs the absolute position (ABS) data control signal to the servo amplifier. Be sure to use transistor outputs for the CPU module outputs.
- For (d2), specify the device that stores the absolute position (ABS) data read from the servo amplifier.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

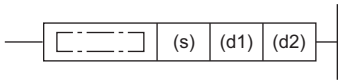
Three devices are occupied from the device specified in (s) and (d1). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

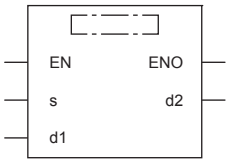
# 16-bit data variable speed pulse

## PLSV [For the FX3 compatible operand specification]

This instruction outputs variable speed pulses with an assigned rotation direction output.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLSV(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d1) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

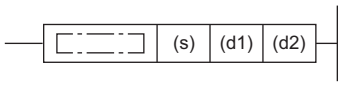
This instruction outputs variable speed pulses with an assigned rotation direction output.

- For (s), specify the command speed to be output. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (d2), specify the device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

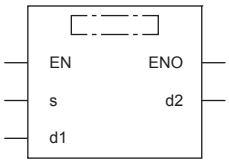
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

# PLSV [For the FX5 operand specification]

This instruction outputs variable speed pulses with an assigned rotation direction output.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=PLSV(EN,s,d1,d2); |

**FBD/LD**




## Setting data

### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

This instruction outputs variable speed pulses with an assigned rotation direction output.

- For (s), specify the command speed to be output. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the abnormal end flag for the PLSV instruction. (This device does not have the normal complete status, and only has the abnormal end status ((d2)+1).

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## Precautions

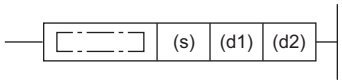
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

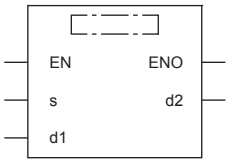
# 32-bit data variable speed pulse

## DPLSV [For the FX3 compatible operand specification]

This instruction outputs variable speed pulses with an assigned rotation direction output.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPLSV(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

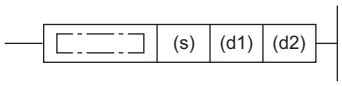
This instruction outputs variable speed pulses with an assigned rotation direction output.

- For (s), specify the command speed to be output. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the output devices (Y) having positioning parameters can be specified.
- For (d2), specify the device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

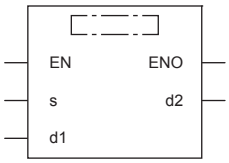
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DPLSV [For the FX5 operand specification]

This instruction outputs variable speed pulses with an assigned rotation direction output.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DPLSV(EN,s,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s) | Command speed | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction outputs variable speed pulses with an assigned rotation direction output.

- For (s), specify the command speed to be output. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the abnormal end flag for the DPLSV instruction. (This device does not have the normal complete status, and only has the abnormal end status ((d2)+1).

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

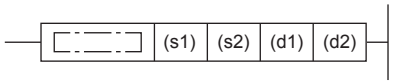
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

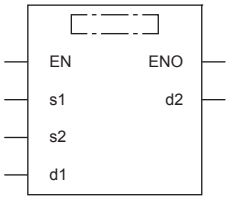
# 16-bit data relative positioning

## DRVI [For the FX3 compatible operand specification]

This instruction executes one-speed positioning by incremental drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVI(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Output bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

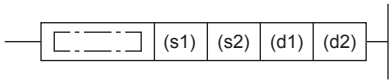
This instruction executes one-speed positioning by incremental drive. Specify the positioning address in the incremental system, in which the transfer direction and transfer distance from the current position (relative address) are specified for positioning.

- For (s1), specify the relative positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the Y devices having positioning parameters can be specified.
- For (d2), specify the bit device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

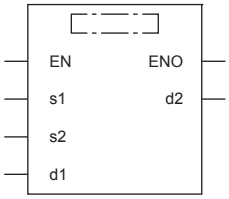
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DRVI [For the FX5 operand specification]

This instruction executes one-speed positioning by incremental drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVI(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes one-speed positioning by incremental drive. Specify the positioning address in the incremental system, in which the transfer direction and transfer distance from the current position (relative address) are specified for positioning.

- For (s1), specify the relative positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DRVI instruction.

For details on the function and error code, refer to 📖MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

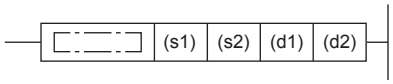
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to 📖MELSEC iQ-F FX5 User's Manual (Positioning Control).

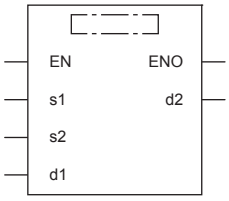
# 32-bit data relative positioning

## DDRVI [For the FX3 compatible operand specification]

This instruction executes one-speed positioning by incremental drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDRVI(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Output bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

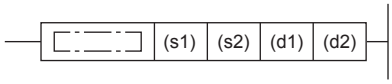
This instruction executes one-speed positioning by incremental drive. Specify the positioning address in the incremental system, in which the transfer direction and transfer distance from the current position (relative address) are specified for positioning.

- For (s1), specify the relative positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the Y devices having positioning parameters can be specified.
- For (d2), specify the device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

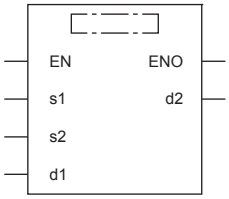
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DDRVI [For the FX5 operand specification]

This instruction executes one-speed positioning by incremental drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDRVI(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes one-speed positioning by incremental drive. Specify the positioning address in the incremental system, in which the transfer direction and transfer distance from the current position (relative address) are specified for positioning.

- For (s1), specify the relative positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DDRVI instruction.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

### Precautions

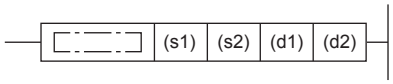
Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

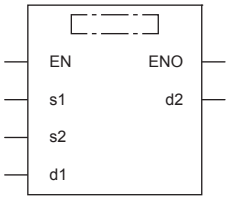
# 16-bit data absolute positioning

## DRVA [For the FX3 compatible operand specification]

This instruction executes one-speed positioning by absolute drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVA(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Output bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

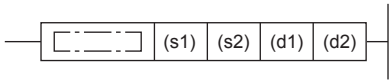
This instruction executes one-speed positioning by absolute drive. Specify the positioning address in the absolute system, in which the transfer distance from the origin (absolute address) is specified for positioning.

- For (s1), specify the absolute positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the Y devices having positioning parameters can be specified.
- For (d2), specify the bit device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

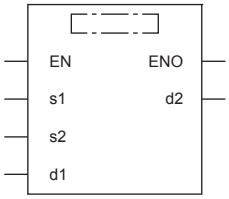
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## DRVA [For the FX5 operand specification]

This instruction executes one-speed positioning by absolute drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DRVA(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -32768 to +32767 | 16-bit signed binary | ANY16 |
| (s2) | Command speed | 1 to 65535 | 16-bit unsigned binary | ANY16 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

### Processing details

This instruction executes one-speed positioning by absolute drive. Specify the positioning address in the absolute system, in which the transfer distance from the origin (absolute address) is specified for positioning.

- For (s1), specify the absolute positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DRVA instruction.

For details on the function and error code, refer to 📖MELSEC iQ-F FX5 User's Manual (Positioning Control).

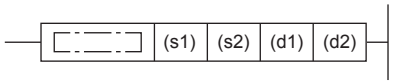
### Precautions

Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.
For other precautions, refer to 📖MELSEC iQ-F FX5 User's Manual (Positioning Control).

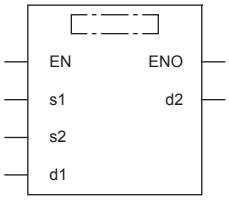
# 32-bit data absolute positioning

## DDRVA [For the FX3 compatible operand specification]

This instruction executes one-speed positioning by absolute drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDRVA(EN,s1,s2,d1,d2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Output bit device number (Y) from which pulses are output | 0 to 3 | Bit | ANY_ELEMENTARY (BOOL) |
| (d2) | Bit device number from which the rotation direction is output | — | Bit | ANY_BOOL |

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | ○*1 | — | — | — | — | — | — | — | — | — | — | — | — |
| (d2) | ○*2 | — | — | ○*3 | — | — | — | — | — | — | — | — | — |

*1 Only Y can be used.

*2 When the output mode is CW/CCW, specify the CCW axis. When the output mode is PULSE/SIGN and using Y, only the SIGN output or general-purpose output of the self-axis can be specified.

*3 T, ST, C cannot be used.

### Processing details

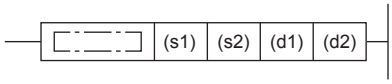
This instruction executes one-speed positioning by absolute drive. Specify the positioning address in the absolute system, in which the transfer distance from the origin (absolute address) is specified for positioning.

- For (s1), specify the absolute positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the device from which pulses are output. Only the Y devices having positioning parameters can be specified.
- For (d2), specify the bit device from which the rotation direction signal is output. Only the device specified with the parameter or general-purpose outputs can be specified. When the output devices (Y) is executed by another function (PWM, positioning PULSE axis, or CW/CCW axis etc.), the device does not function and causes an error.

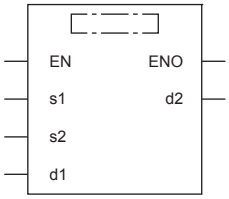
For details on the function, precautions, and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

# DDRVA [For the FX5 operand specification]

This instruction executes one-speed positioning by absolute drive.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=DDRVA(EN,s1,s2,d1,d2); |

**FBD/LD**




## Setting data

### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (s1) | Positioning address | -2147483648 to +2147483647 | 32-bit signed binary | ANY32 |
| (s2) | Command speed | 1 to 2147483647 | 32-bit signed binary | ANY32 |
| (d1) | Axis number from which pulses are to be output | K1 to 4 | 16-bit unsigned binary | ANY_ELEMENTARY (WORD) |
| (d2) | Bit device number of the positioning complete flag or abnormal end flag | — | Bit | ANY_BOOL |

### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (s1) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (s2) | ○ | — | — | ○ | ○ | ○ | ○ | ○ | ○ | ○ | — | — | — |
| (d1) | — | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d2) | ○ | — | — | ○*1 | — | — | — | — | — | — | — | — | — |

*1 T, ST, C cannot be used.

## Processing details

This instruction executes one-speed positioning by absolute drive. Specify the positioning address in the absolute system, in which the transfer distance from the origin (absolute address) is specified for positioning.

- For (s1), specify the absolute positioning address in user units. (The address must be within the range of -2147483647 to +2147483647 number of pulses.)
- For (s2), specify the command speed in user units. (The speed must be 200 Kpps or lower in frequency.)
- For (d1), specify the axis number from which pulses are output.
- For (d2), specify the bit device of the normal complete flag or abnormal end flag for the DDRVA instruction.

For details on the function and error code, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

## Precautions

Two devices are occupied from the device specified in (d2). Make sure that these devices are not used in other controls.

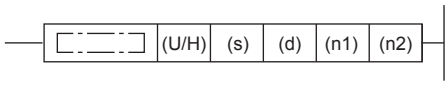
For other precautions, refer to MELSEC iQ-F FX5 User's Manual (Positioning Control).

# 15 DIVIDED DATA READ/WRITE FROM/TO BFM INSTRUCTION

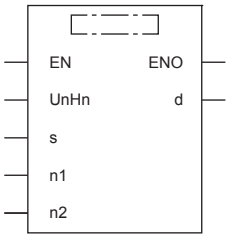
## 15.1 Divided BFM Read

### RBFM

This instruction reads data from continuous buffer memory areas in an FX3 intelligent function module

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=RBFM(EN,UnHn,s,n1,n2,d); |

**FBD/LD**






### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U/H)*1 | Module number | K1 to 16 | 16-bit unsigned binary | ANY16_U |
| (s) | Head buffer memory number | 0 to 32767 | 16-bit unsigned binary | ANY16_U |
| (d) | Head device number storing data to be read from buffer memory | — | 16-bit signed binary | ANY16 |
| (n1) | Number of all buffer memory areas to be read | 1 to 32768 | 16-bit unsigned binary | ANY16_U |
| (n2) | Not used | — | 16-bit unsigned binary | ANY16_U |

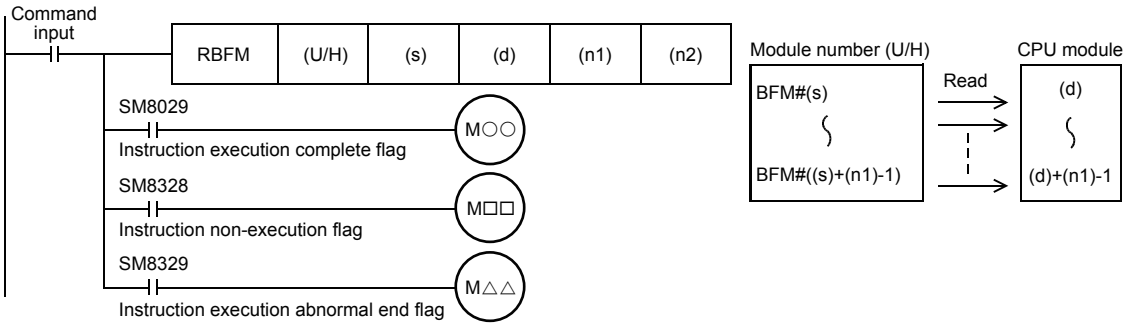
*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (d) | — | — | — | ○ | — | ○ | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- This instruction reads (n1) points of buffer memory starting from (s) inside the intelligent function module number (U/H) to (d) in the CPU module. When (n1) exceeds 64 points, it divides and reads by several scans. (64 points are read in one scan)



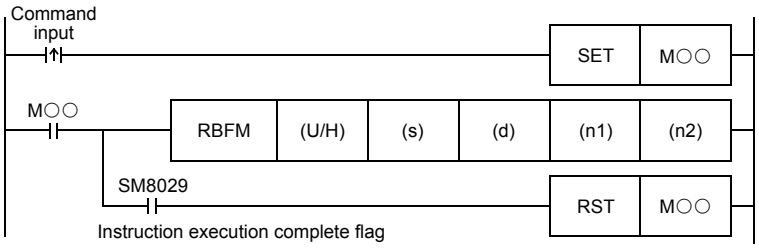

- When this instruction is finished normally, instruction execution complete flag (SM8029) turns on. When this instruction is finished abnormally, instruction execution abnormal end flag (SM8329) turns on.
- When this instruction or the WBFM instruction is executed in the same scan, instruction non-execution flag (SM8328) is set to on, and execution of such an instruction is paused. When execution of the other target instruction is complete, the paused instruction resumes.

### ■Related devices

| Device | Name | Description |
|---|---|---|
| SM8029 | Instruction execution complete | Turns ON when an instruction is finished normally. |
| SM8328 | Instruction non-execution | Turns ON when the RBFM instruction or WBFM instruction in another step is executed for the same module number. |
| SM8329 | Instruction execution abnormal end | Turns ON when an instruction is finished abnormally. |

## Precautions

- Do not stop the instruction while it is being executed. If driving is stopped, the buffer memory reading processing is suspended, but the data that is already read is stored in (d) onwards. Stop the instruction after execution completes as in the following program.




- When indexing is executed, the contents of index registers at the beginning of execution are used. Even if the contents of index registers are changed after the instruction, such changes do not affect the process of the instruction.
- The contents of (n1) points starting from (d) update (change) every scan while this instruction is executed. Use the data after the instruction is completed.
- Do not update (change) the contents of (n1) buffer memory areas starting from the buffer memory (s) while this instruction is executed. If the contents are updated, the intended data may not be read.
- This instruction cannot be used in FX5 intelligent function modules.
- This instruction cannot be used while a interrupt routine program is being executed.

### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Communication procedure with module is not completed correctly when this instruction is executed. |
| 2801H | Module with the module number specified by (U/H) does not exist. |
| 2823H | The number of transfer points specified by (n1) and the buffer memory number specified by (s) is beyond the buffer memory area range. |
| 2820H | The number of transfer points specified by (n1) and the device number specified by (d) is beyond the specified device range. |
| 3580H | Instructions that cannot be used in an interrupt routine program are being used. |

## Common items between RBFM instruction and WBFM instruction

### ■Specification of module number of FX3 intelligent function module and buffer memory

For FX3 intelligent function module connection method, number of connectable FX3 intelligent function modules and handling of I/O numbers, refer to 📖 manuals of the CPU module and FX3 intelligent function modules.

• Module number of FX3 intelligent function module

Use the module number to specify for which equipment the RBFM/WBFM instruction is used. (Setting range: K1 to K16)

| | | Module No. 1 | Module No. 2 | Module No. 3 |
|---|---|---|---|---|
| CPU module | I/O module | Bus conversion module | Intelligent function module | Intelligent function module |

A module number is automatically assigned to each intelligent function module connected to the CPU module. The module number is assigned as No.1 → No.2 → No.3… starting from the equipment nearest the CPU module.

• Buffer memory number

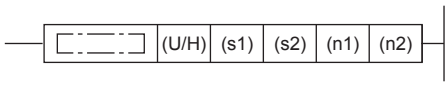
The intelligent function module incorporates a RAM memory. The RAM memory is called buffer memory. Buffer memory numbers range from #0 to #32767 and their contents vary depending on the function of the extension equipment. (Setting range: K0 to K32767)

For the contents of buffer memory areas, refer to 📖 manuals of intelligent function modules.

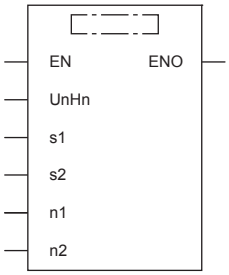
# 15.2 Divided BFM Write

## WBFM

This instruction writes data to continuous buffer memory areas in an FX3 intelligent function module.

| Ladder diagram | Structured text |
|---|---|
|  | ENO:=WBFM(EN,UnHn,s1,s2,n1,n2); |

**FBD/LD**




### Setting data

#### ■Descriptions, ranges, and data types

| Operand | Description | Range | Data type | Data type (label) |
|---|---|---|---|---|
| (U/H)*1 | Module number | K1 to 16 | 16-bit unsigned binary | ANY16_U |
| (s1) | Head buffer memory number | 0 to 32767 | 16-bit unsigned binary | ANY16_U |
| (s2) | Head device number storing data to be written to buffer memory | — | 16-bit signed binary | ANY16 |
| (n1) | Number of all buffer memory areas to be written | 1 to 32768 | 16-bit unsigned binary | ANY16_U |
| (n2) | Not used | — | 16-bit unsigned binary | ANY16_U |

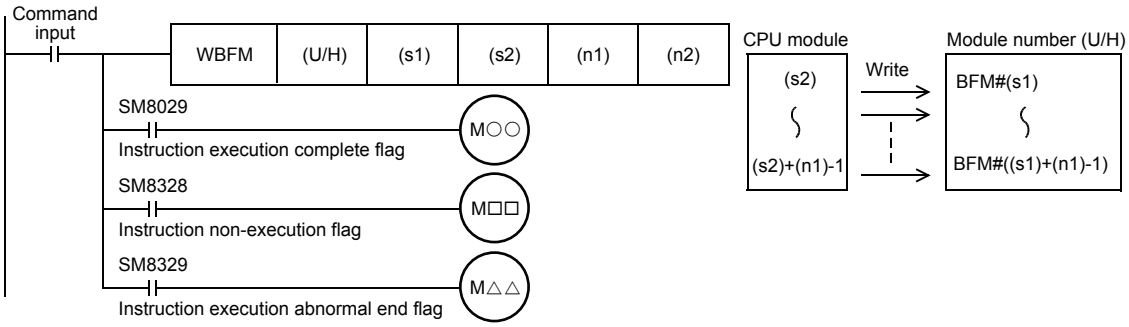
*1 In the case of the ST language and the FBD/LD language, U/H displays as UnHn.

#### ■Applicable devices

| Operand | Bit | | | Word | | | Double word | | Indirect specification | Constant | | | Others |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X, Y, M, L, SM, F, B, SB, S | U□\G□ | T, ST, C, LC | T, ST, C, D, W, SD, SW, R | U□\G□ | Z | LC | LZ | | K, H | E | $ | |
| (U/H) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (s2) | — | — | — | ○ | — | ○ | — | — | ○ | — | — | — | — |
| (n1) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |
| (n2) | ○ | — | — | ○ | ○ | ○ | — | — | ○ | ○ | — | — | — |

## Processing details

- This instruction writes (n1) points of buffer memory starting from (s1) inside the intelligent function module number (U/H) to (s2) in the CPU module. When (n1) exceeds 64 points, it divides and writes by several scans. (64 points are read in one scan)




- When this instruction is finished normally, instruction execution complete flag (SM8029) turns on. When this instruction is finished abnormally, instruction execution abnormal end flag (SM8329) turns on.
- When this instruction or the RBFM instruction is executed in the same scan, instruction non-execution flag (SM8328) is set to on, and execution of such an instruction is paused. When execution of the other target instruction is complete, the paused instruction resumes.

### ■Related devices

| Device | Name | Description |
|---|---|---|
| SM8029 | Instruction execution complete | Turns ON when an instruction is finished normally. |
| SM8328 | Instruction non-execution | Turns ON when the RBFM instruction or WBFM instruction in another step is executed for the same module number. |
| SM8329 | Instruction execution abnormal end | Turns ON when an instruction is finished abnormally. |

## Precautions

- Do not stop the instruction while it is being executed. If driving is stopped, the buffer memory write processing is suspended, but the data that is already written is stored in (m2) onwards.
- When indexing is executed, the contents of index registers at the beginning of execution are used. Even if the contents of index registers are changed after the instruction, such changes do not affect the process of the instruction.
- Do not update (change) the contents of (n1) points starting from (s2) while this instruction is executed. If the contents are updated, the intended data may not be written to the buffer memory areas.
- This instruction cannot be used in FX5 intelligent function modules.
- This instruction cannot be used while a interrupt routine program is being executed.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2441H | Communication procedure with module is not completed correctly when this instruction is executed. |
| 2801H | Module with the module number specified by (U/H) does not exist, or the specified module is not supported. |
| 2823H | The number of transfer points specified by (n1) and the buffer memory number specified by (s1) is beyond the buffer memory range. |
| 2820H | The number of transfer points specified by (n1) and the device number specified by (s2) is beyond the specified device range. |
| 3580H | Instructions that cannot be used in an interrupt routine program are being used. |

# MEMO

# PART 5 STANDARD FUNCTIONS

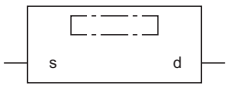
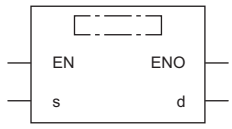
This part consists of the following chapters.

# 16 TYPE CONVERSION FUNCTIONS

## 16.1 Converting BOOL to WORD

### BOOL_TO_WORD(_E)

These functions convert BOOL type data to WORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=BOOL_TO_WORD(s);<br>[With EN/ENO]<br>d:=BOOL_TO_WORD_E(EN,ENO,s); |

#### Setting data

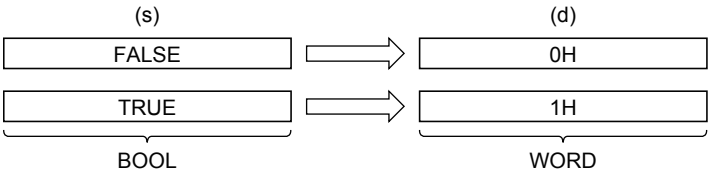
##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_WORD(_E)) | Output | Output variable | WORD |

#### Processing details

##### ■Operation processing

- These functions convert the BOOL type data input to (s) to WORD type data and output from (d).
- When the input value is "FALSE", these functions output 0H as the WORD type data value.
- When the input value is "TRUE", these functions output 1H as the WORD type data value.




- A value input to (s) is the BOOL type data value.

##### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

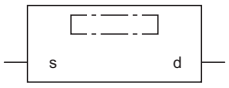
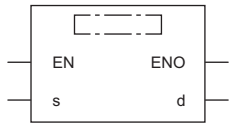
#### Operation error

There is no operation error.

# 16.2 Converting BOOL to DWORD

## BOOL_TO_DWORD(_E)

These functions convert BOOL type data to DWORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=BOOL_TO_DWORD(s);<br>[With EN/ENO]<br>d:=BOOL_TO_DWORD_E(EN,ENO,s); |

### Setting data

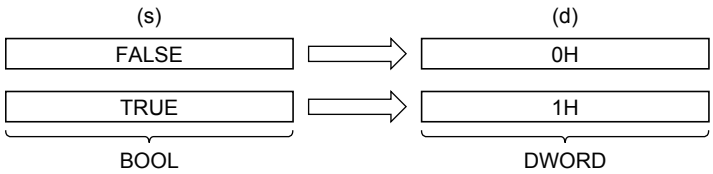
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_DWORD(_E)) | Output | Output variable | DWORD |

### Processing details

#### ■Operation processing

- These functions convert the BOOL type data input to (s) to DWORD type data and output from (d).
- When the input value is "FALSE", these functions output 0H as the DWORD type data value.
- When the input value is "TRUE", these functions output 1H as the DWORD type data value.




- A value input to (s) is the BOOL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

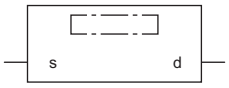
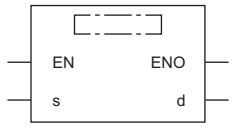
### Operation error

There is no operation error.

# 16.3 Converting BOOL to INT

## BOOL_TO_INT(_E)

These functions convert BOOL type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=BOOL_TO_INT(s);<br>[With EN/ENO]<br>d:=BOOL_TO_INT_E(EN,ENO,s); |
|  |  | |

### Setting data

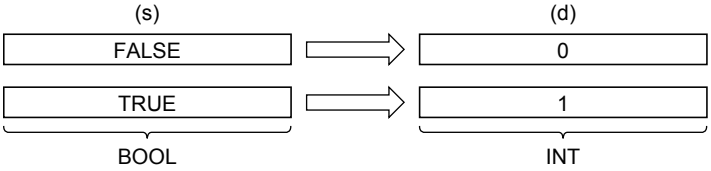
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the BOOL type data input to (s) to INT type data and output from (d).
- When the input value is "FALSE", these functions output 0 as the INT type data value.
- When the input value is "TRUE", these functions output 1 as the INT type data value.




- A value input to (s) is the BOOL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

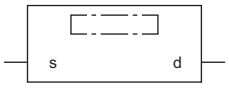
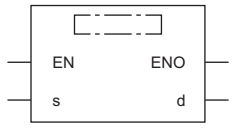
### Operation error

There is no operation error.

# 16.4 Converting BOOL to DINT

## BOOL_TO_DINT(_E)

These functions convert BOOL type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=BOOL_TO_DINT(s);<br>[With EN/ENO]<br>d:=BOOL_TO_DINT_E(EN,ENO,s); |

### Setting data

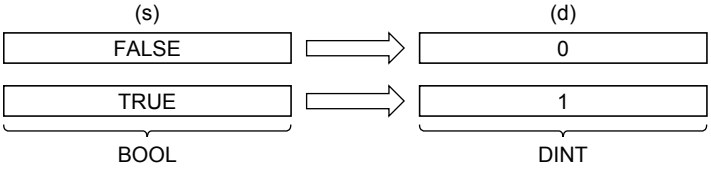
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the BOOL type data input to (s) to DINT type data and output from (d).
- When the input value is "FALSE", these functions output 0 as the DINT type data value.
- When the input value is "TRUE", these functions output 1 as the DINT type data value.




- A value input to (s) is the BOOL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

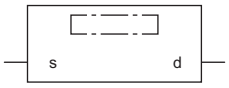
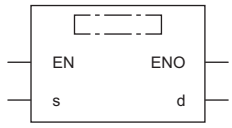
### Operation error

There is no operation error.

# 16.5 Converting BOOL to TIME

## BOOL_TO_TIME(_E)

These functions convert BOOL type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=BOOL_TO_TIME(s);<br>[With EN/ENO]<br>d:=BOOL_TO_TIME_E(EN,ENO,s); |

### Setting data

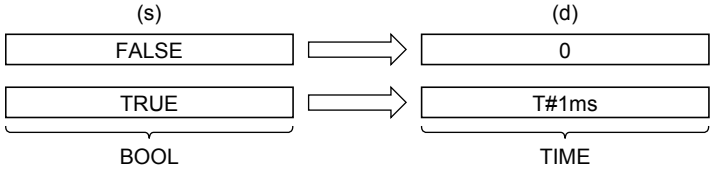
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the BOOL type data input to (s) to TIME type data and output from (d).
- When the input value is "FALSE", these functions output 0 as the TIME type data value.
- When the input value is "TRUE", these functions output 1 as the TIME type data value.




- A value input to (s) is the BOOL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

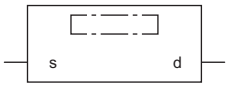
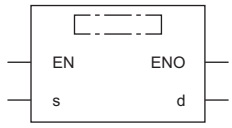
### Operation error

There is no operation error.

# 16.6 Converting BOOL to STRING

## BOOL_TO_STRING(_E)

These functions convert BOOL type data to STRING type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=BOOL_TO_STRING(s);<br>[With EN/ENO]<br>d:=BOOL_TO_STRING_E(EN,ENO,s); |

### Setting data

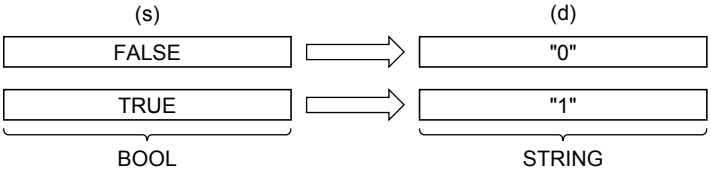
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BOOL_TO_STRING(_E)) | Output | Output variable | STRING |

### Processing details

#### ■Operation processing

- These functions convert the BOOL type data input to (s) to STRING type data and output from (d).
- When the input value is "FALSE", these functions output 0 as the STRING type data value.
- When the input value is "TRUE", these functions output 1 as the STRING type data value.




- A value input to (s) is the BOOL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

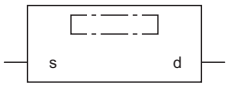
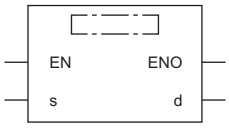
### Operation error

There is no operation error.

# 16.7 Converting WORD to BOOL

## WORD_TO_BOOL(_E)

These functions convert WORD type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=WORD_TO_BOOL(s);<br>[With EN/ENO]<br>d:=WORD_TO_BOOL_E(EN,ENO,s); |

### Setting data

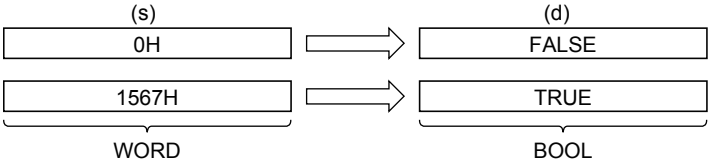
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(WORD_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the WORD type data input to (s) to BOOL type data and output from (d).
- When the input value is 0H, these functions output "FALSE".
- When the input value is any value other than 0H, these functions output "TRUE".




- A value input to (s) is the WORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

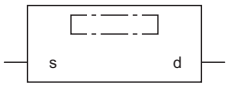
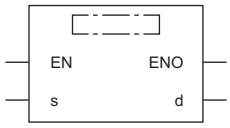
### Operation error

There is no operation error.

# 16.8 Converting WORD to DWORD

## WORD_TO_DWORD(_E)

These functions convert WORD type data to DWORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=WORD_TO_DWORD(s);<br>[With EN/ENO]<br>d:=WORD_TO_DWORD_E(EN,ENO,s); |

### Setting data

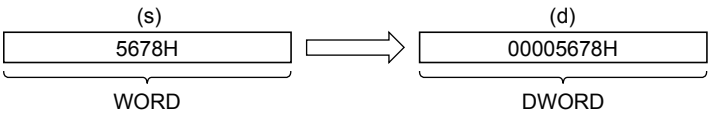
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(WORD_TO_DWORD(_E)) | Output | Output variable | DWORD |

### Processing details

#### ■Operation processing

- These functions convert the WORD type data input to (s) to DWORD type data and output from (d).
- Each of high-order 16 bits becomes "0" after data conversion.




- A value input to (s) is the WORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

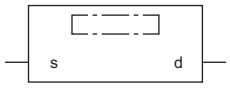
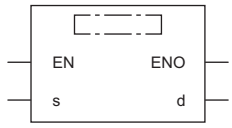
### Operation error

There is no operation error.

# 16.9 Converting WORD to INT

## WORD_TO_INT(_E)

These functions convert WORD type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=WORD_TO_INT(s);<br>[With EN/ENO]<br>d:=WORD_TO_INT_E(EN,ENO,s); |

### Setting data

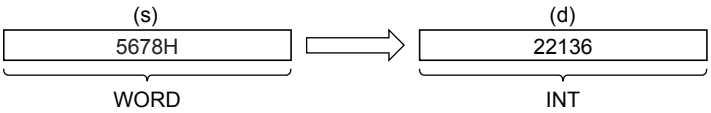
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(WORD_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the WORD type data input to (s) to INT type data and output from (d).




- A value input to (s) is the WORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

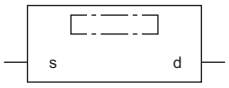
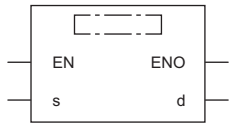
### Operation error

There is no operation error.

# 16.10 Converting WORD to DINT

## WORD_TO_DINT(_E)

These functions convert WORD type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=WORD_TO_DINT(s);<br>[With EN/ENO]<br>d:=WORD_TO_DINT_E(EN,ENO,s); |

### Setting data

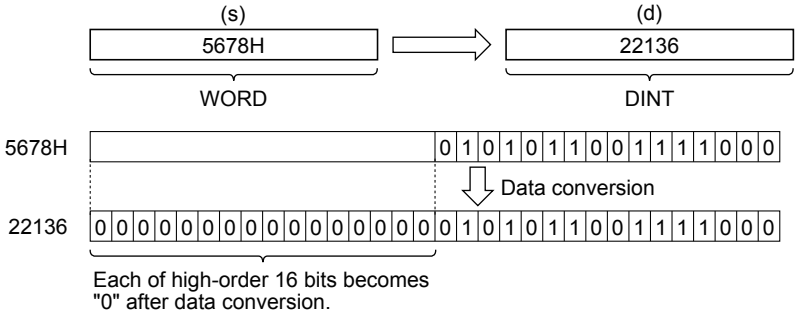
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(WORD_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the WORD type data input to (s) to DINT type data and output from (d).
- Each of high-order 16 bits becomes "0" after data conversion.




- A value input to (s) is the WORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

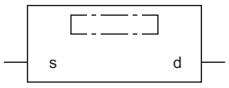
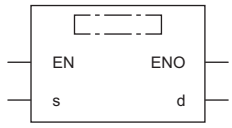
### Operation error

There is no operation error.

# 16.11 Converting WORD to TIME

## WORD_TO_TIME(_E)

These functions convert WORD type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=WORD_TO_TIME(s);<br>[With EN/ENO]<br>d:=WORD_TO_TIME_E(EN,ENO,s); |

### Setting data

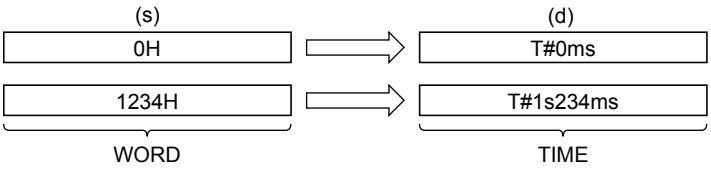
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(WORD_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the WORD type data input to (s) to TIME type data and output from (d).




- A value input to (s) is the WORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

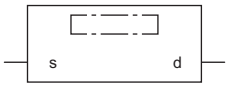
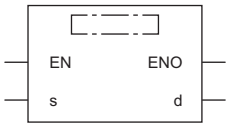
### Operation error

There is no operation error.

# 16.12 Converting DWORD to BOOL

## DWORD_TO_BOOL(_E)

These functions convert DWORD type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DWORD_TO_BOOL(s);<br>[With EN/ENO]<br>d:=DWORD_TO_BOOL_E(EN,ENO,s); |

### Setting data

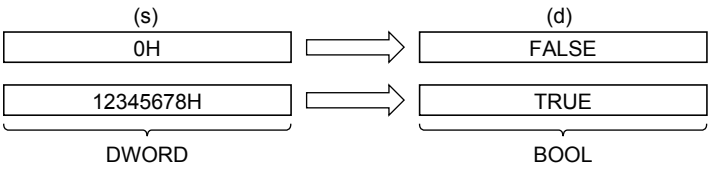
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DWORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DWORD_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the DWORD type data input to (s) to BOOL type data and output from (d).
- When the input value is 0H, these functions output "FALSE".
- When the input value is any value other than 0H, these functions output "TRUE".




- A value input to (s) is the DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

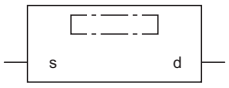
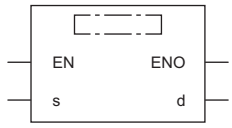
### Operation error

There is no operation error.

# 16.13 Converting DWORD to WORD

## DWORD_TO_WORD(_E)

These functions convert DWORD type data to WORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DWORD_TO_WORD(s);<br>[With EN/ENO]<br>d:=DWORD_TO_WORD_E(EN,ENO,s); |

### Setting data

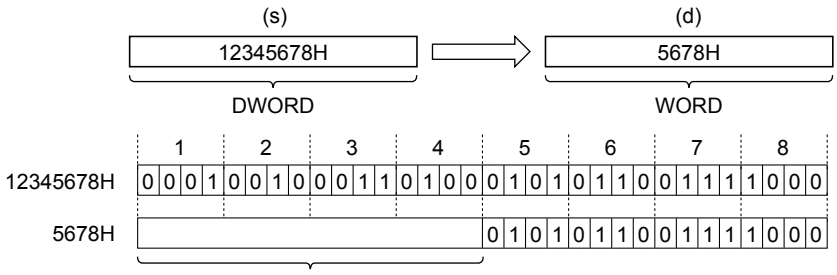
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DWORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DWORD_TO_WORD(_E)) | Output | Output variable | WORD |

### Processing details

#### ■Operation processing

- These functions convert the DWORD type data input to (s) to WORD type data and output from (d).
- The information stored in high-order 16 bits of an input value is discarded.




The information stored in high-order 16 bits is discarded.

- A value input to (s) is the DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

**Point**

When DWORD_TO_WORD(_E) is executed, the information stored in high-order 16 bits of the DWORD type data value input from (s) is discarded.

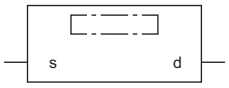
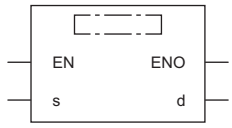
## Operation error

There is no operation error.

# 16.14 Converting DWORD to INT

## DWORD_TO_INT(_E)

These functions convert DWORD type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DWORD_TO_INT(s);<br>[With EN/ENO]<br>d:=DWORD_TO_INT_E(EN,ENO,s); |

### Setting data

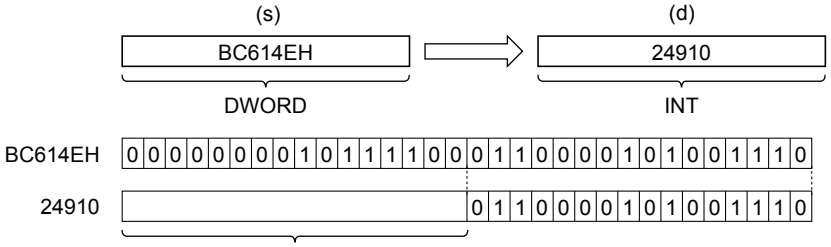
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DWORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DWORD_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the DWORD type data input to (s) to INT type data and output from (d).
- The information stored in high-order 16 bits of an input value is discarded.




The information stored in high-order 16 bits is discarded.

- A value input to (s) is the DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

**Point**

When DWORD_TO_INT(_E) is executed, the information stored in high-order 16 bits of the DWORD type data value input from (s) is discarded.

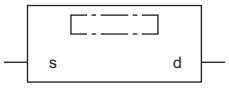
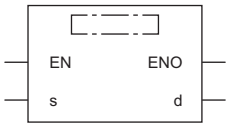
## Operation error

There is no operation error.

# 16.15 Converting DWORD to DINT

## DWORD_TO_DINT(_E)

These functions convert DWORD type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DWORD_TO_DINT(s);<br>[With EN/ENO]<br>d:=DWORD_TO_DINT_E(EN,ENO,s); |

### Setting data

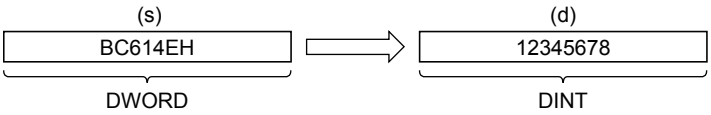
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DWORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DWORD_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the DWORD type data input to (s) to DINT type data and output from (d).




- A value input to (s) is the DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

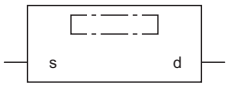
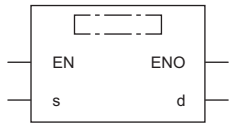
### Operation error

There is no operation error.

# 16.16 Converting DWORD to TIME

## DWORD_TO_TIME(_E)

These functions convert DWORD type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DWORD_TO_TIME(s);<br>[With EN/ENO]<br>d:=DWORD_TO_TIME_E(EN,ENO,s); |

### Setting data

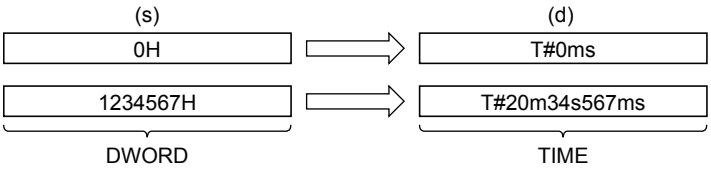
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DWORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DWORD_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the DWORD type data input to (s) to TIME type data and output from (d).




- A value input to (s) is the DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

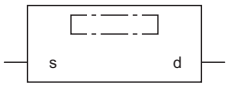
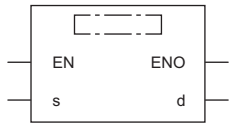
### Operation error

There is no operation error.

# 16.17 Converting INT to BOOL

## INT_TO_BOOL(_E)

These functions convert INT type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=INT_TO_BOOL(s);<br>[With EN/ENO]<br>d:=INT_TO_BOOL_E(EN,ENO,s); |

### Setting data

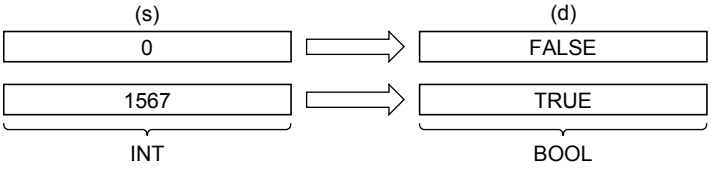
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to BOOL type data and output from (d).
- When the input value is 0, these functions output "FALSE".
- When the input value is any value other than 0, these functions output "TRUE".




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

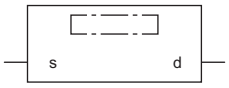
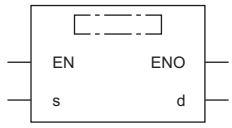
### Operation error

There is no operation error.

# 16.18 Converting INT to WORD

## INT_TO_WORD(_E)

These functions convert INT type data to WORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=INT_TO_WORD(s);<br>[With EN/ENO]<br>d:=INT_TO_WORD_E(EN,ENO,s); |

### Setting data

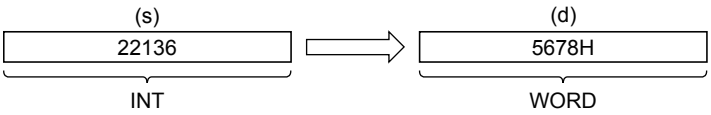
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_WORD(_E)) | Output | Output variable | WORD |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to WORD type data and output from (d).




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

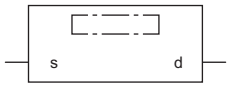
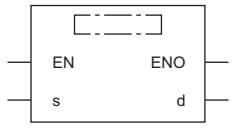
### Operation error

There is no operation error.

# 16.19 Converting INT to DWORD

## INT_TO_DWORD(_E)

These functions convert INT type data to DWORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=INT_TO_DWORD(s);<br>[With EN/ENO]<br>d:=INT_TO_DWORD_E(EN,ENO,s); |

### Setting data

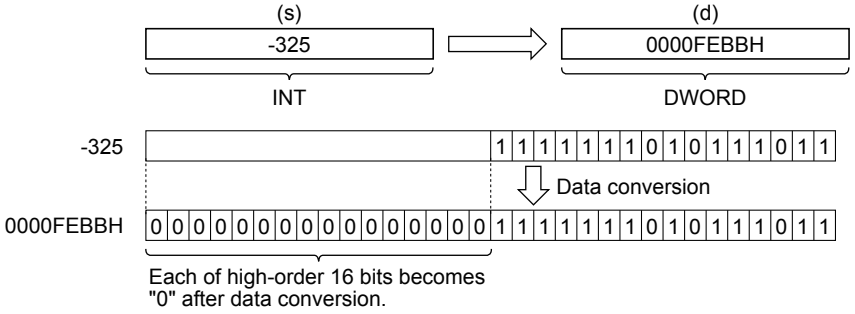
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_DWORD(_E)) | Output | Output variable | DWORD |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to DWORD type data and output from (d).
- Each of high-order 16 bits becomes "0" after data conversion.




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

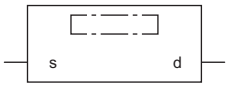
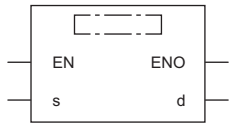
### Operation error

There is no operation error.

# 16.20 Converting INT to DINT

## INT_TO_DINT(_E)

These functions convert INT type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=INT_TO_DINT(s);<br>[With EN/ENO]<br>d:=INT_TO_DINT_E(EN,ENO,s); |
|  |  | |

### Setting data

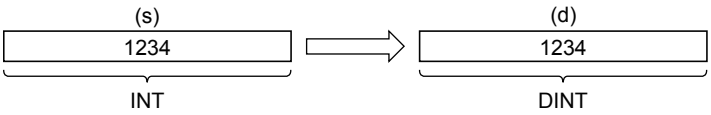
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to DINT type data and output from (d).




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

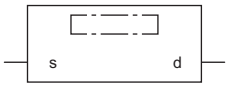
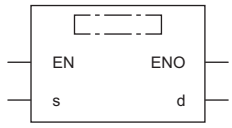
### Operation error

There is no operation error.

# 16.21 Converting INT to BCD

## INT_TO_BCD(_E)

These functions convert INT type data to BCD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=INT_TO_BCD(s);<br>[With EN/ENO]<br>d:=INT_TO_BCD_E(EN,ENO,s); |
|  |  | |

### Setting data

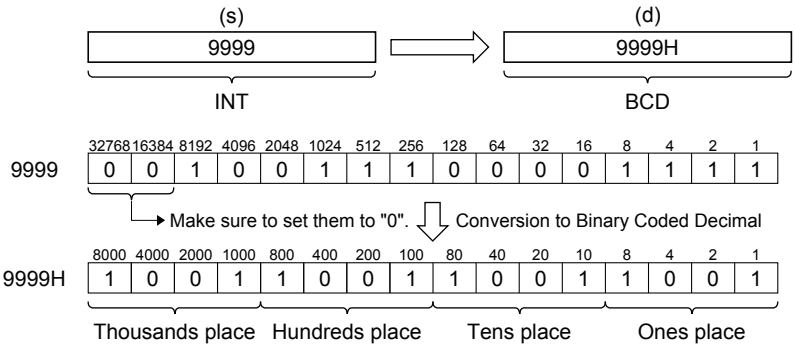
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_BCD(_E)) | Output | Output variable | WORD |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to BCD type data and output from (d).




- A value input to (s) is the INT type data value and within the range from 0 to 9999.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

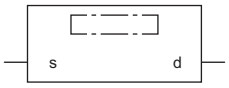
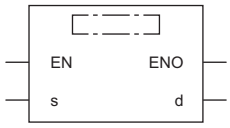
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Data in the device specified by (s) is out of the valid range (0 to 9999). |

# 16.22 Converting INT to REAL

## INT_TO_REAL(_E)

These functions convert INT type data to REAL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=INT_TO_REAL(s);<br>[With EN/ENO]<br>d:=INT_TO_REAL_E(EN,ENO,s); |

### Setting data

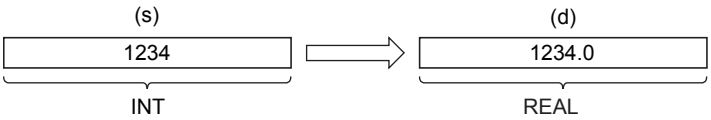
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_REAL(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to REAL type data and output from (d).




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

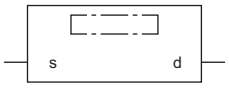
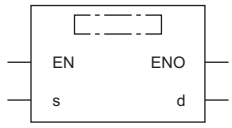
### Operation error

There is no operation error.

# 16.23 Converting INT to TIME

## INT_TO_TIME(_E)

These functions convert INT type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=INT_TO_TIME(s);<br>[With EN/ENO]<br>d:=INT_TO_TIME_E(EN,ENO,s); |

### Setting data

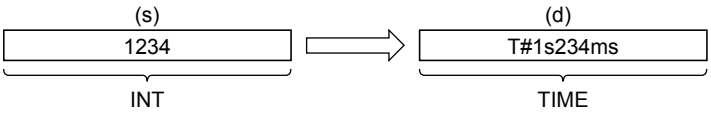
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the INT type data input to (s) to TIME type data and output from (d).




- A value input to (s) is the INT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

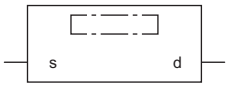
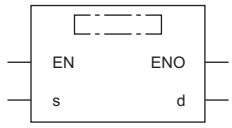
### Operation error

There is no operation error.

# 16.24 Converting INT to STRING

## INT_TO_STRING(_E)

These functions convert INT type data to STRING type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=INT_TO_STRING(s);<br>[With EN/ENO]<br>d:=INT_TO_STRING_E(EN,ENO,s); |
|  |  | |

### Setting data

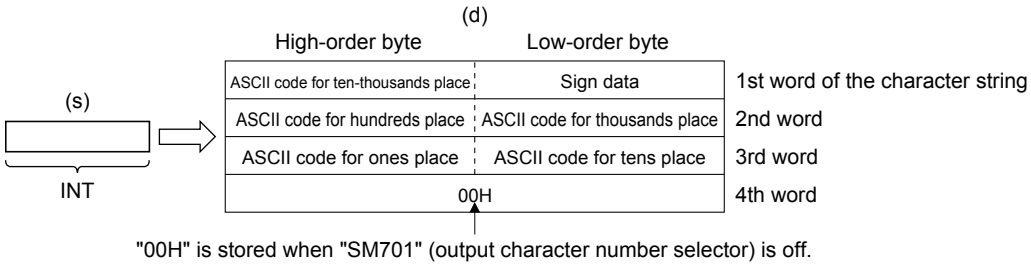
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_STRING(_E)) | Output | Output variable | STRING(6) |

### Processing details

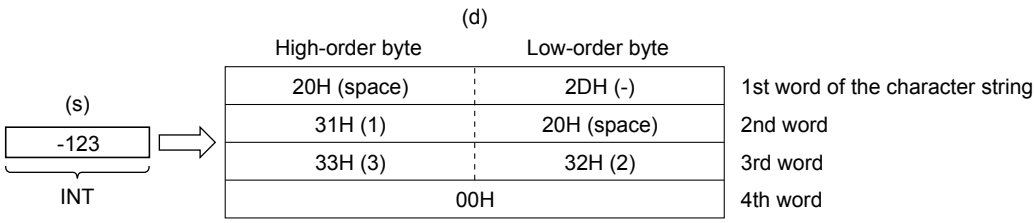
#### ■Operation processing

- These functions convert the INT type data input to (s) to STRING type data and output from (d).




- A value input to (s) is the INT type data value.
- In "Sign data", 20H (space) is stored when the input value is positive, and 2DH (-) is stored when the input value is negative.
- 20H (space) is stored in high-order digits when the number of significant figures is small.

Ex.
When "-123" is input




- 00H is stored at the end (4th word) of the character string when SM701 (output character number selector signal) is off.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

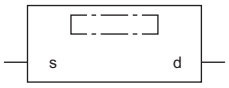
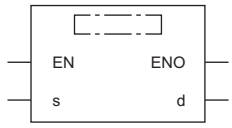
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "0000H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 16.25 Converting DINT to BOOL

## DINT_TO_BOOL(_E)

These functions convert DINT type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DINT_TO_BOOL(s);<br>[With EN/ENO]<br>d:=DINT_TO_BOOL_E(EN,ENO,s); |

### Setting data

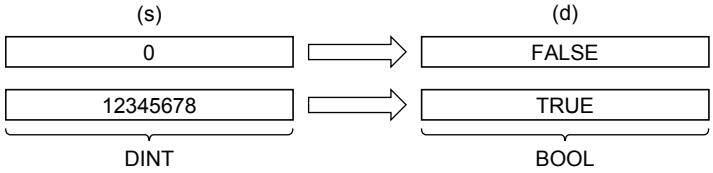
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to BOOL type data and output from (d).
- When the input value is 0, these functions output "FALSE".
- When the input value is any value other than 0, these functions output "TRUE".




- A value input to (s) is the DINT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

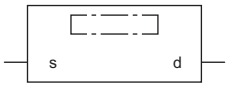
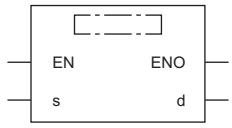
### Operation error

There is no operation error.

# 16.26 Converting DINT to WORD

## DINT_TO_WORD(_E)

These functions convert DINT type data to WORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DINT_TO_WORD(s);<br>[With EN/ENO]<br>d:=DINT_TO_WORD_E(EN,ENO,s); |
|  |  | |

### Setting data

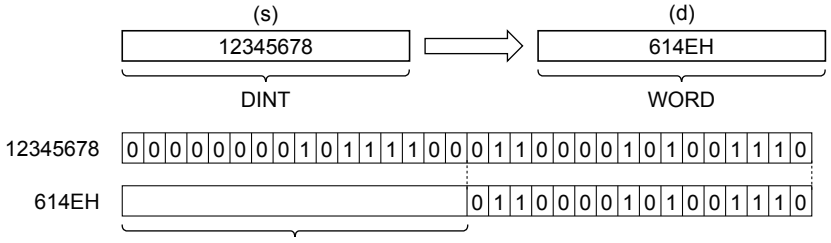
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_WORD(_E)) | Output | Output variable | WORD |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to WORD type data and output from (d).
- The information stored in high-order 16 bits of an input value is discarded.




The information stored in high-order 16 bits is discarded.

- A value input to (s) is the DINT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

**Point**

When DINT_TO_WORD(_E) is executed, the information stored in high-order 16 bits of the DINT type data value input from (s) is discarded.

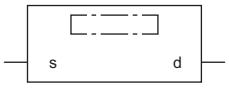
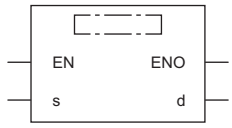
## Operation error

There is no operation error.

# 16.27 Converting DINT to DWORD

## DINT_TO_DWORD(_E)

These functions convert DINT type data to DWORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DINT_TO_DWORD(s);<br>[With EN/ENO]<br>d:=DINT_TO_DWORD_E(EN,ENO,s); |

### Setting data

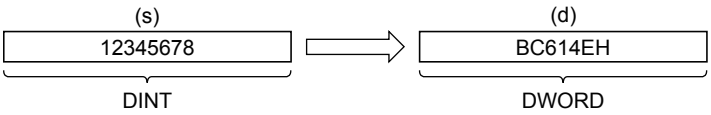
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_DWORD(_E)) | Output | Output variable | DWORD |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to DWORD type data and output from (d).




- A value input to (s) is the DINT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

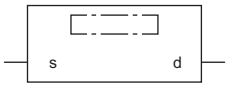
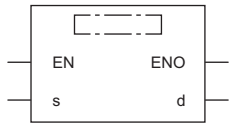
### Operation error

There is no operation error.

# 16.28 Converting DINT to INT

## DINT_TO_INT(_E)

These functions convert DINT type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DINT_TO_INT(s);<br>[With EN/ENO]<br>d:=DINT_TO_INT_E(EN,ENO,s); |
|  |  | |

### Setting data

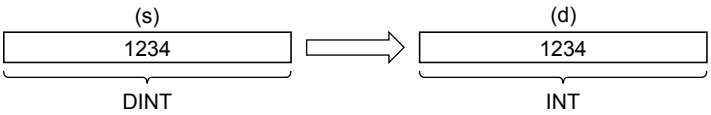
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to INT type data and output from (d).




- A value input to (s) is the DINT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

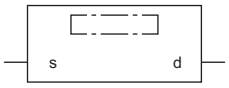
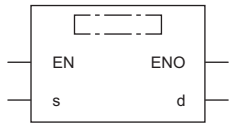
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The 32-bit signed binary data in the device specified by (s) is out of the valid range (-32768 to 32767). |

# 16.29 Converting DINT to BCD

## DINT_TO_BCD(_E)

These functions convert DINT type data to BCD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DINT_TO_BCD(s);<br>[With EN/ENO]<br>d:=DINT_TO_BCD_E(EN,ENO,s); |
|  |  | |

### Setting data

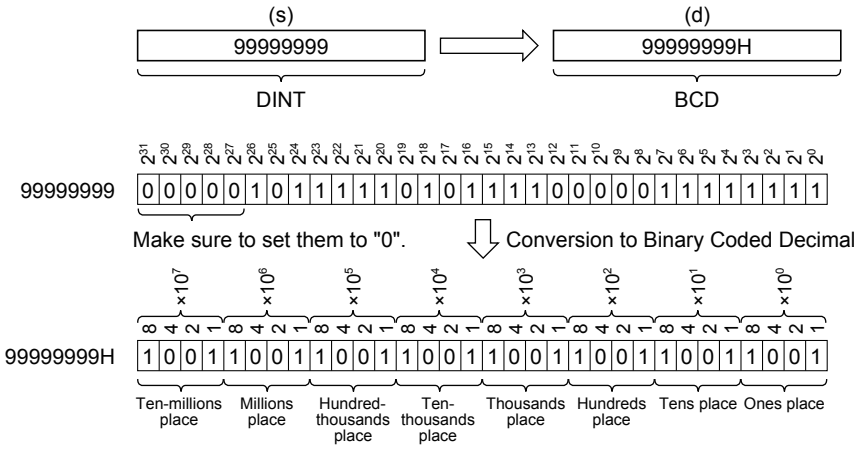
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_BCD(_E)) | Output | Output variable | ANY_BIT |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to BCD type data and output from (d).




- A value input to (s) is the DINT type data value. When (d) is WORD, the input value is within the range from 0 to 9999. When (d) is DWORD, the input value is within the range from 0 to 99999999.
- WORD or DWORD can be specified to (d). BOOL cannot be specified.

## ■Operation result

### 1. Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

- When (d) is WORD

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The 32-bit signed binary data in the device specified by (s) is out of the valid range (-32768 to 32767). |
| | Data in the device specified by (s) is out of the valid range (0 to 9999). |

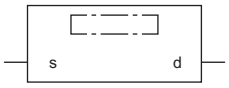
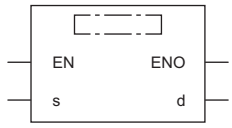
- When (d) is DWORD

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Data in the device specified by (s) is out of the valid range (0 to 99999999). |

# 16.30 Converting DINT to REAL

## DINT_TO_REAL(_E)

These functions convert DINT type data to REAL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=DINT_TO_REAL(s);<br>[With EN/ENO]<br>d:=DINT_TO_REAL_E(EN,ENO,s); |

### Setting data

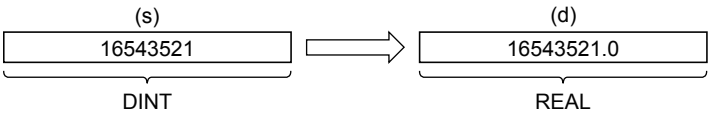
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_REAL(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to REAL type data and output from (d).




- A value input to (s) is the DINT type data value.
- The number of significant figures of the REAL type data is approximately 7 since the data is processed in 32-bit single precision.
- The converted data includes an error (rounding error) if an integer value is outside the range of -16777216 to 16777215.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

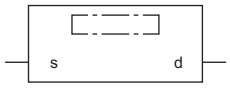
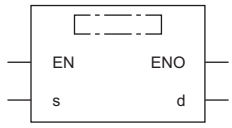
### Operation error

There is no operation error.

# 16.31 Converting DINT to TIME

## DINT_TO_TIME(_E)

These functions convert DINT type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DINT_TO_TIME(s);<br>[With EN/ENO]<br>d:=DINT_TO_TIME_E(EN,ENO,s); |
|  |  | |

### Setting data

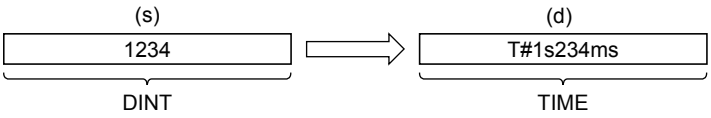
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to TIME type data and output from (d).




- A value input to (s) is the DINT type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

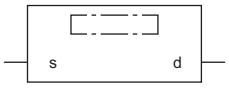
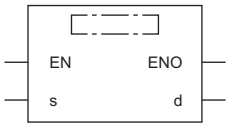
### Operation error

There is no operation error.

# 16.32 Converting DINT to STRING

## DINT_TO_STRING(_E)

These functions convert DINT type data to STRING type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DINT_TO_STRING(s);<br>[With EN/ENO]<br>d:=DINT_TO_STRING_E(EN,ENO,s); |

### Setting data

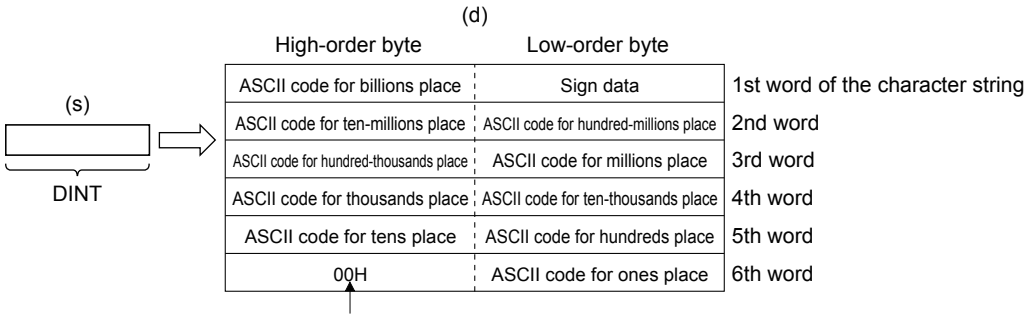
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | DINT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_STRING(_E)) | Output | Output variable | STRING(11) |

### Processing details

#### ■Operation processing

- These functions convert the DINT type data input to (s) to STRING type data and output from (d).

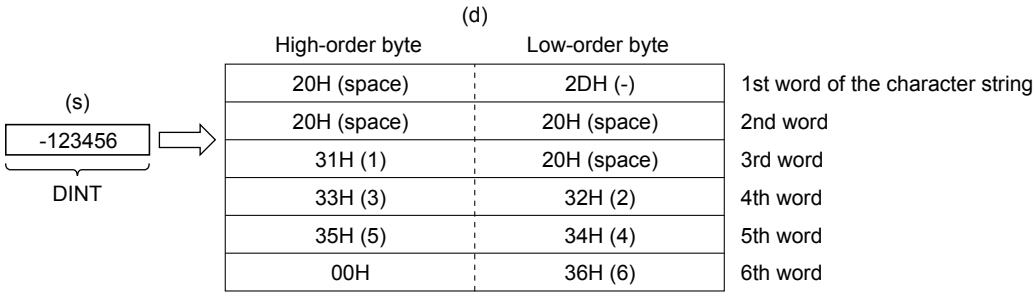



"00H" is stored when "SM701" (output character number selector) is off.

- A value input to (s) is the DINT type data value.
- In "Sign data", 20H (space) is stored when the input value is positive, and 2DH (-) is stored when the input value is negative.
- 20H (space) is stored in high-order digits when the number of significant figures is small.

Ex.

When "-123456" is input




- 00H is stored at the end (high-order byte of the 6th word) of the character string when SM701 (output character number selector signal) is off.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

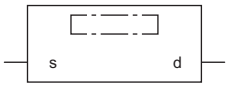
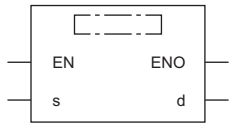
## Operation error

There is no operation error.

# 16.33 Converting BCD to INT

## BCD_TO_INT(_E)

These functions convert BCD type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=BCD_TO_INT(s);<br>[With EN/ENO]<br>d:=BCD_TO_INT_E(EN,ENO,s); |
|  |  | |

### Setting data

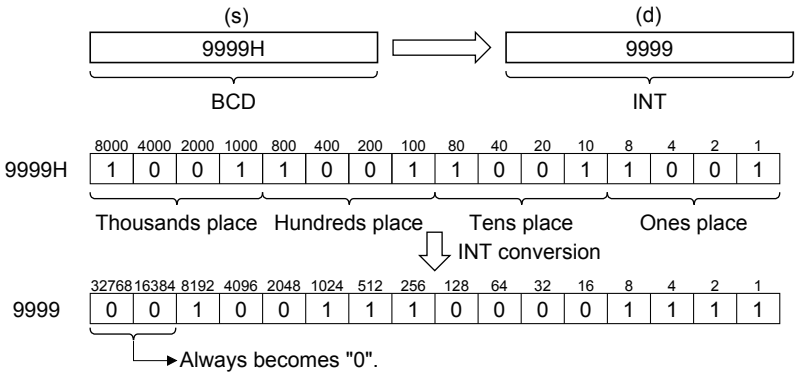
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | WORD |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BCD_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the BCD type data input to (s) to INT type data and output from (d).




- A value input to (s) is the WORD type data value and within the range from 0H to 9999H (from 0 to 9 for each digit).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

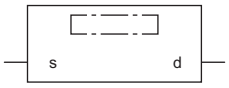
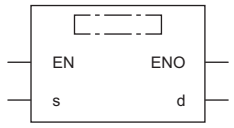
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | A value other than 0 to 9 exists in each digit of (s). |

# 16.34 Converting BCD to DINT

## BCD_TO_DINT(_E)

These functions convert BCD type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=BCD_TO_DINT(s);<br>[With EN/ENO]<br>d:=BCD_TO_DINT_E(EN,ENO,s); |
|  |  | |

### Setting data

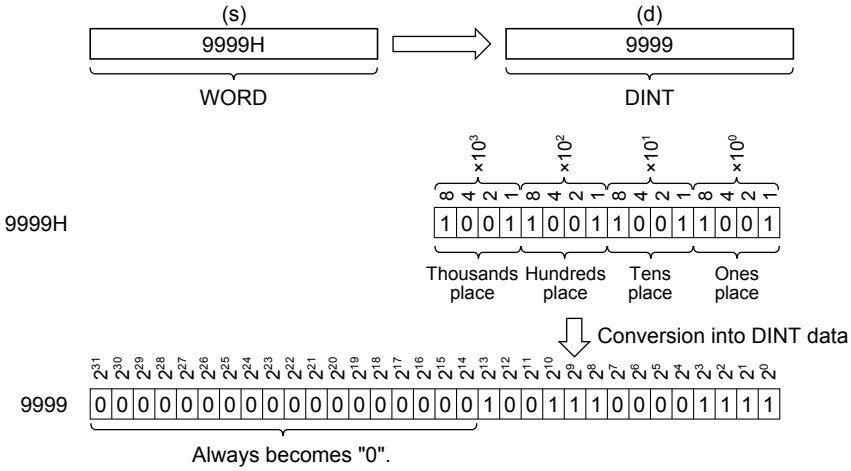
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BCD_TO_DINT(_E)) | Output | Output variable | DINT |

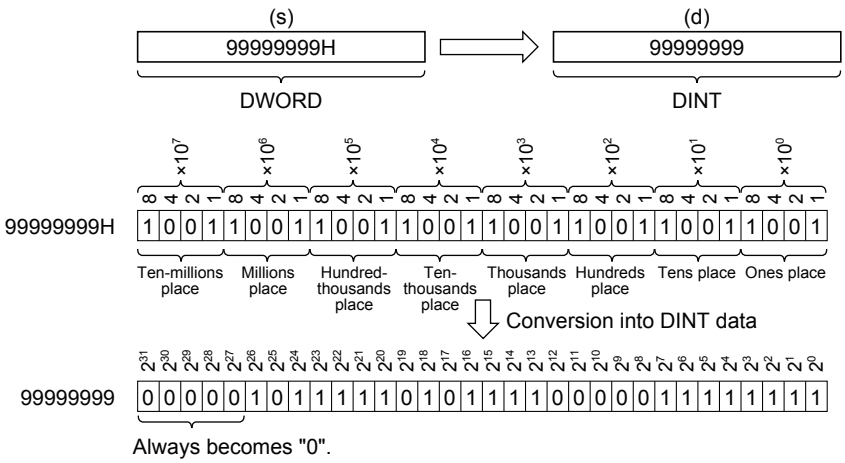
### Processing details

#### ■Operation processing

- These functions convert the BCD type data input to (s) to DINT type data and output from (d).
  - When WORD is specified to (s)




  - When DWORD is specified to (s)




16

- A value input to (s) is within the range from 0H to 9999H (from 0 to 9 for each digit) for the WORD type data value and from 0H to 99999999H (from 0 to 9 for each digit) for the DWORD type data value.
- WORD or DWORD can be specified to (s). BOOL cannot be specified.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

- When (s) is WORD

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | A value other than 0 to 9 exists in each digit of (s). |

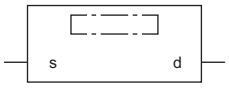
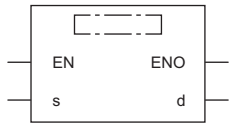
- When (s) is DWORD

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | A value other than 0 to 9 exists in each digit of (s). |

# 16.35 Converting REAL to INT

## REAL_TO_INT(_E)

These functions convert REAL type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=REAL_TO_INT(s);<br>[With EN/ENO]<br>d:=REAL_TO_INT_E(EN,ENO,s); |
|  |  | |

### Setting data

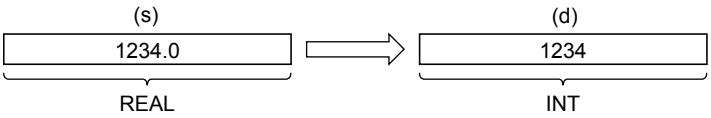
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(REAL_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the REAL type data input to (s) to INT type data and output from (d).




- A value input to (s) is the REAL type data value and within the range from -32768 to 32767.
- After conversion, the first digit after the decimal point of the REAL type data value is rounded off.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

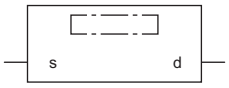
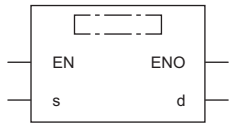
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (-32768 to 32767). |
| 3402H | • A special number is set to (s).<br>• The set single-precision real number is not located within the following range.<br>0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$<br>• The set device or label value is -0, denormalized number, NaN (not a number), or $\pm\infty$. |

# 16.36 Converting REAL to DINT

## REAL_TO_DINT(_E)

These functions convert REAL type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=REAL_TO_DINT(s);<br>[With EN/ENO]<br>d:=REAL_TO_DINT_E(EN,ENO,s); |

### Setting data

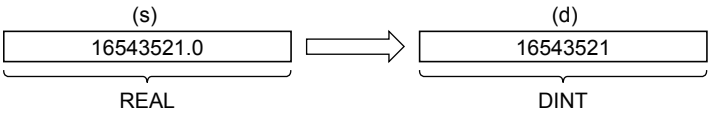
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(REAL_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the REAL type data input to (s) to DINT type data and output from (d).




- A value input to (s) is the REAL type data value and within the range from -2147483648 to 2147483647.
- After conversion, the first digit after the decimal point of the REAL type data value is rounded off.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

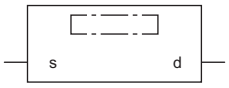
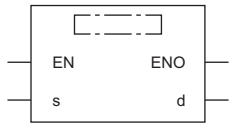
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The single-precision real number in the device specified by (s) is out of the valid range (-2147483648 to 2147483647). |
| 3402H | A special number is set to (s).<br>• The set single-precision real number is not located within the following range.<br>0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$<br>• The set device or label value is -0, denormalized number, NaN (not a number), or $\pm\infty$. |

# 16.37 Converting REAL to STRING

## REAL_TO_STRING(_E)

These functions convert REAL type data to STRING type data (exponent format).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=REAL_TO_STRING(s);<br>[With EN/ENO]<br>d:=REAL_TO_STRING_E(EN,ENO,s); |

### Setting data

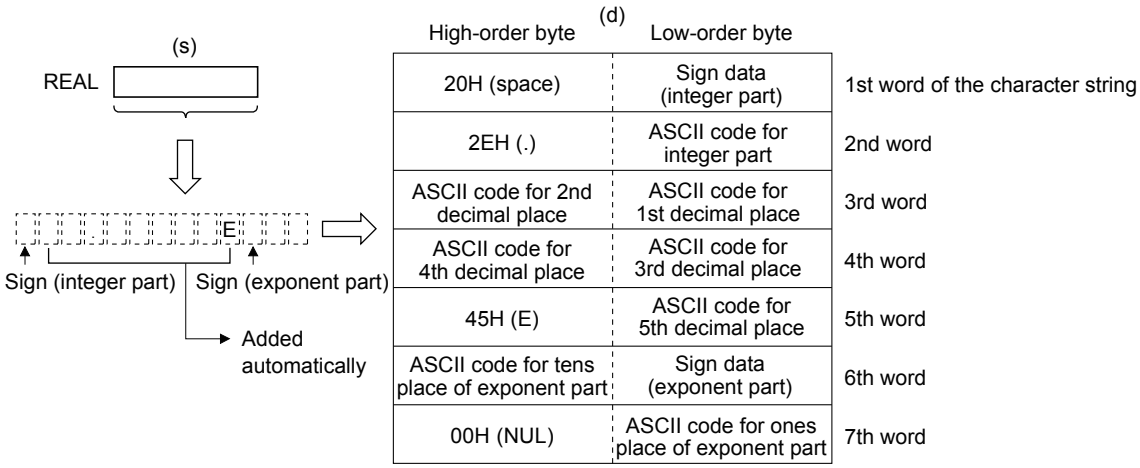
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(REAL_TO_STRING(_E)) | Output | Output variable | STRING(13) |

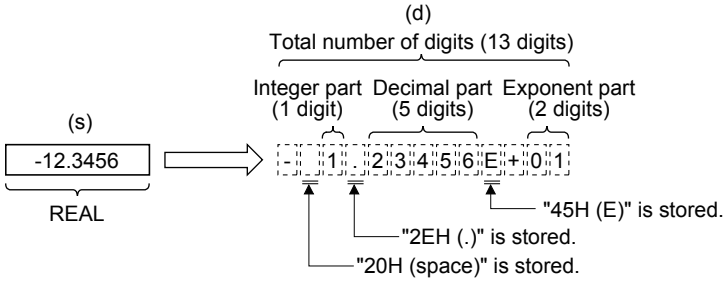
### Processing details

#### ■Operation processing

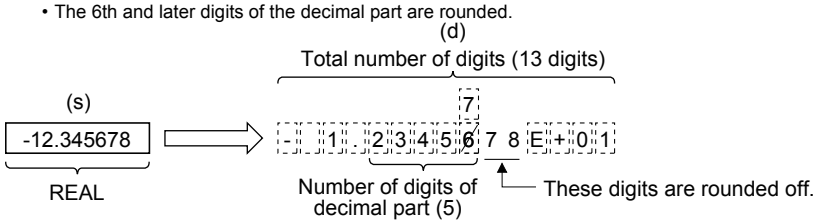
- These functions convert the REAL type data input to (s) to STRING type (exponent format) data and output from (d).



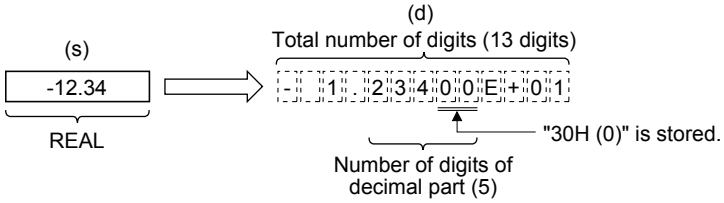

- A value input to (s) is the REAL type data value.
- The string data obtained by conversion is output from (d) as follows:
  - The number of digits is fixed respectively for the integer part, decimal part and exponent part as follows: Integer part: 1, decimal part: 5, exponent part: 2
  - "20H (space)" is stored in the 2nd byte, "2EH (.)" is stored in the 4th byte, and "45H (E)" is stored in the 10th byte automatically.



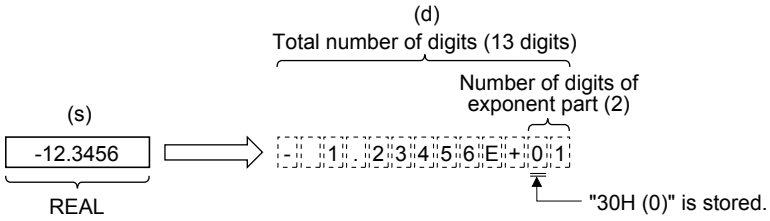
• In "Sign data (integer part)", "20H (space)" is stored when the input value is positive, and "2DH (-)" is stored when the input value is negative.
• The 6th and later digits of the decimal part are rounded.




• "30H (0)" is stored in the decimal part when the number of significant figures is small.




• In "Sign data (exponent part)", "2BH (+)" is stored when the input value is positive, and "2DH (-)" is stored when the input value is negative.
• "30H (0)" is stored in the tens place of the exponent part when the exponent part consists of 1 digit.




• "00H" is automatically stored at the end (7th word) of the character string.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

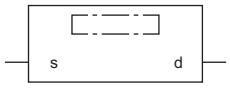
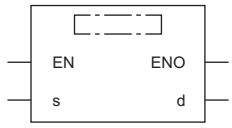
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | (s) is not located within the following range<br>• 0, $2^{-126} \leq$ \|specified device value\| $< 2^{128}$<br>• (s) is -0, denormalized number, NaN (not a number), or $\pm\infty$. |
| 3406H | The whole converted character string cannot be stored in the devices from the device specified by (d) to the last device of the target device. |

# 16.38 Converting TIME to BOOL

## TIME_TO_BOOL(_E)

These functions convert TIME type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=TIME_TO_BOOL(s);<br>[With EN/ENO]<br>d:=TIME_TO_BOOL_E(EN,ENO,s); |

### Setting data

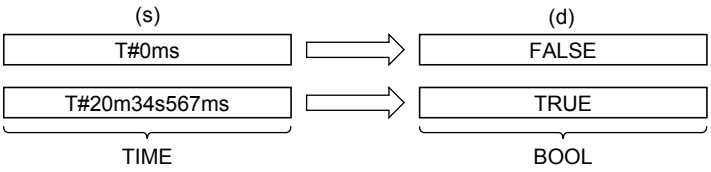
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to BOOL type data and output from (d).




#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

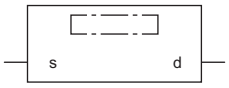
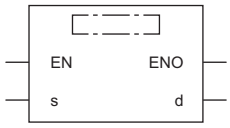
### Operation error

There is no operation error.

# 16.39 Converting TIME to WORD

## TIME_TO_WORD(_E)

These functions convert TIME type data to WORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=TIME_TO_WORD(s);<br>[With EN/ENO]<br>d:=TIME_TO_WORD_E(EN,ENO,s); |

### Setting data

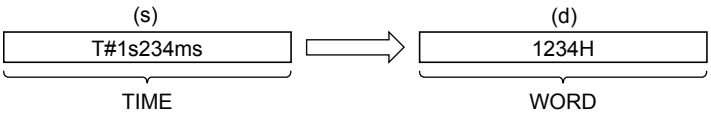
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_WORD(_E)) | Output | Output variable | WORD |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to WORD type data and output from (d).




- A value input to (s) is the TIME type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

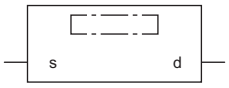
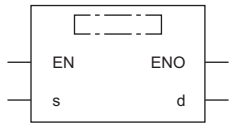
### Operation error

There is no operation error.

# 16.40 Converting TIME to DWORD

## TIME_TO_DWORD(_E)

These functions convert TIME type data to DWORD type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=TIME_TO_DWORD(s);<br>[With EN/ENO]<br>d:=TIME_TO_DWORD_E(EN,ENO,s); |

### Setting data

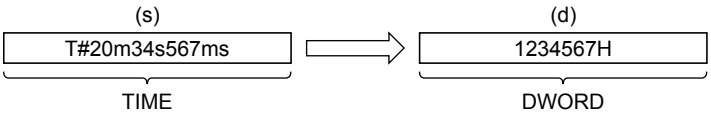
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_DWORD(_E)) | Output | Output variable | DWORD |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to DWORD type data and output from (d).




- A value input to (s) is the TIME type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

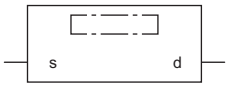
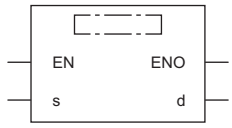
### Operation error

There is no operation error.

# 16.41 Converting TIME to INT

## TIME_TO_INT(_E)

These functions convert TIME type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=TIME_TO_INT(s);<br>[With EN/ENO]<br>d:=TIME_TO_INT_E(EN,ENO,s); |

### Setting data

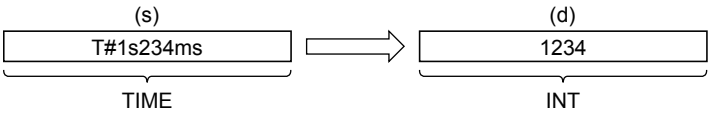
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to INT type data and output from (d).




- A value input to (s) is the TIME type data value.
- When the data is converted to INT, the TIME type data stored in high-order 16 bits (1 word) are ignored.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

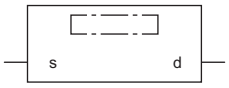
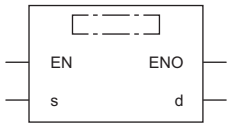
### Operation error

There is no operation error.

# 16.42 Converting TIME to DINT

## TIME_TO_DINT(_E)

These functions convert TIME type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=TIME_TO_DINT(s);<br>[With EN/ENO]<br>d:=TIME_TO_DINT_E(EN,ENO,s); |
|  |  | |

### Setting data

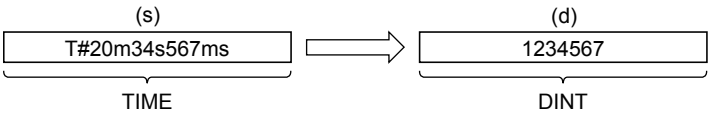
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to DINT type data and output from (d).




- A value input to (s) is the TIME type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

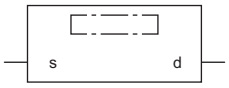
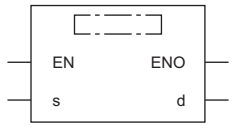
### Operation error

There is no operation error.

# 16.43 Converting TIME to STRING

## TIME_TO_STRING(_E)

These functions convert TIME type data to STRING type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=TIME_TO_STRING(s);<br>[With EN/ENO]<br>d:=TIME_TO_STRING_E(EN,ENO,s); |
|  |  | |

### Setting data

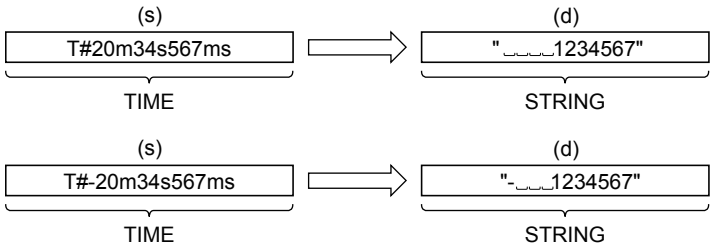
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TIME_TO_STRING(_E)) | Output | Output variable | STRING(11) |

### Processing details

#### ■Operation processing

- These functions convert the TIME type data input to (s) to STRING type data and output from (d).




- A value input to (s) is the TIME type data value.
- 00H is stored at the end of the character string when SM701 (output character number selector signal) is off.
- The following shows the operation result to be stored in the output.
  - As the 1st character, "20H" (space) is stored if the binary data is positive, and "2DH" (-) is stored if the data is negative.
  - "20H" (space) is stored on the left side of the effective digits.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

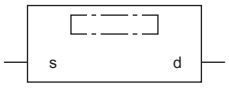
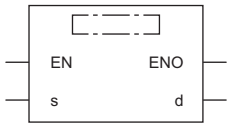
### Operation error

There is no operation error.

# 16.44 Converting STRING to BOOL

## STRING_TO_BOOL(_E)

These functions convert STRING type data to BOOL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=STRING_TO_ BOOL(s);<br>[With EN/ENO]<br>d:=STRING_TO_ BOOL_E(EN,ENO,s); |

### Setting data

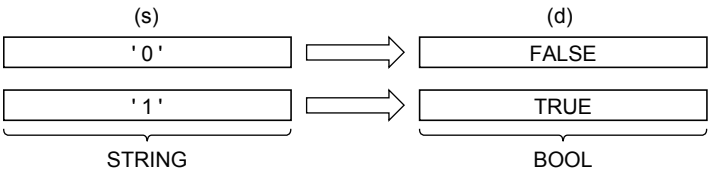
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(1) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(STRING_TO_BOOL(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions convert the STRING type (in the decimal format or exponent format) data input to (s) to BOOL type data and output from (d).




#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

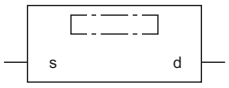
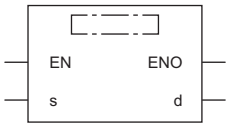
### Operation error

There is no operation error.

# 16.45 Converting STRING to INT

## STRING_TO_INT(_E)

These functions convert STRING type data to INT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=STRING_TO_INT(s);<br>[With EN/ENO]<br>d:=STRING_TO_INT_E(EN,ENO,s); |
|  |  | |

### Setting data

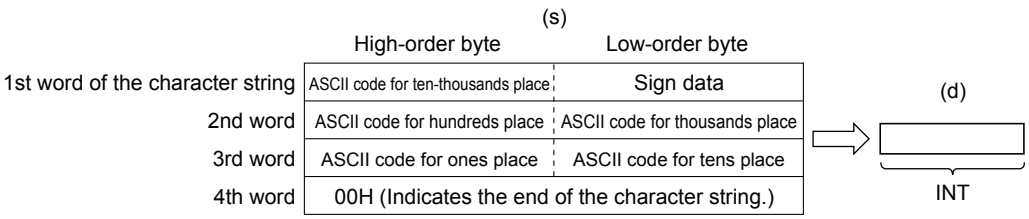
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(6) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(STRING_TO_INT(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions convert the STRING type data input to (s) to INT type data and output from (d).




- A value input to (s) is the STRING type data value and within the following range.
  - Within the range of "30H" to "39H", "20H", "2DH", and "00H" in ASCII code
  - Within the range of "-32768" to "32767" as the STRING type data value

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

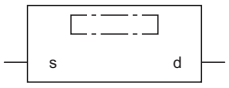
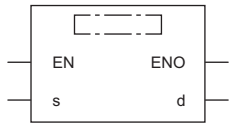
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Invalid data which cannot be converted to (s) are input.<br>• Values for each place of the ASCII code are other than "30H" to "39H", "20H", and "00H".<br>• Values for the ASCII data are other than "-32768" to "32767" when STRING_TO_INT(_E) is used. |

# 16.46 Converting STRING to DINT

## STRING_TO_DINT(_E)

These functions convert STRING type data to DINT type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=STRING_TO_DINT(s);<br>[With EN/ENO]<br>d:=STRING_TO_DINT_E(EN,ENO,s); |

### Setting data

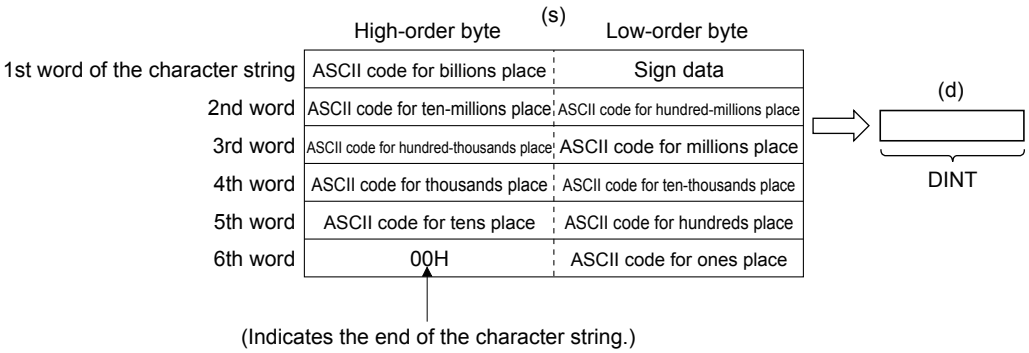
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(11) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(STRING_TO_DINT(_E)) | Output | Output variable | DINT |

### Processing details

#### ■Operation processing

- These functions convert the STRING type data input to (s) to DINT type data and output from (d).




- A value input to (s) is the STRING type data value and within the following range.
  - Within the range of "30H" to "39H", "20H", "2DH", and "00H" in ASCII code
  - Within the range of "-2147483648" to "2147483647" as the STRING type data value

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

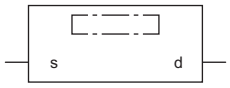
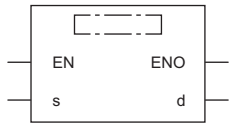
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Invalid data which cannot be converted to (s) are input.<br>• Values for each place of the ASCII code are other than "30H" to "39H", "20H", and "00H".<br>• Values for the ASCII data are other than "-2147483648" to "2147483647" when STRING_TO_DINT(_E) is used. |

# 16.47 Converting STRING to REAL

## STRING_TO_REAL(_E)

These functions convert STRING type data to REAL type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=STRING_TO_REAL(s);<br>[With EN/ENO]<br>d:=STRING_TO_REAL_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(24) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(STRING_TO_REAL(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions convert the STRING type (in the decimal format or exponent format) data input to (s) to REAL type data and output from (d).

(s)

| | High-order byte | Low-order byte |
|---|---|---|
| 1st word of the character string | ASCII code for 1st character | Sign data |
| 2nd word | ASCII code for 3rd character | ASCII code for 2nd character |
| 3rd word | ASCII code for 5th character | ASCII code for 4th character |
| 4th word | ASCII code for 7th character | ASCII code for 6th character |
| 5th word | ASCII code for 9th character | ASCII code for 8th character |
| 6th word | ASCII code for 11th character | ASCII code for 10th character |
| 7th word | 00H (Indicates the end of the character string.) | |

(d)

REAL

- The conversion source STRING type data can be in the decimal format or exponent format.
  - Decimal point format

(s)

| | High-order byte | Low-order byte |
|---|---|---|
| 1st word of the character string | 31H (1) | 2DH (-) |
| 2nd word | 33H (3) | 2EH (.) |
| 3rd word | 30H (0) | 35H (5) |
| 4th word | 34H (4) | 33H (3) |
| 5th word | 00H | |

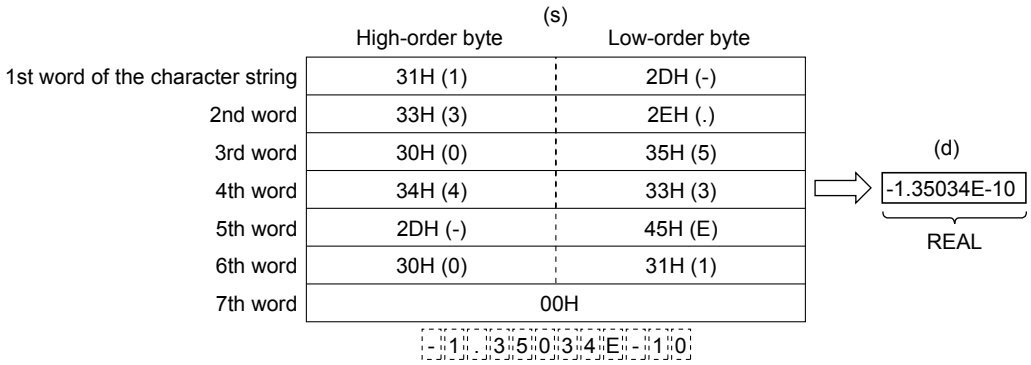
- 1 . 3 5 0 3 4

(d)

-1.35034

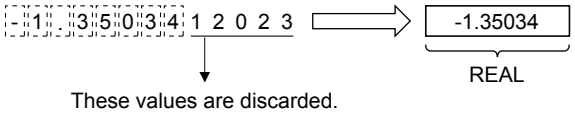
REAL

• Exponent format

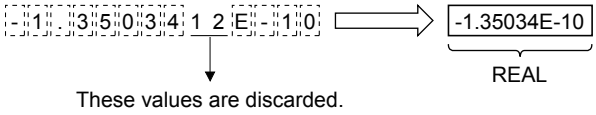



- With regard to STRING type data, six digits excluding the sign, decimal point and exponent part are valid, and the 7th and later digits are discarded during conversion.

• Decimal point format

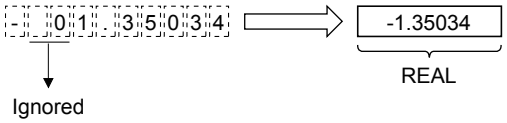



• Exponent format

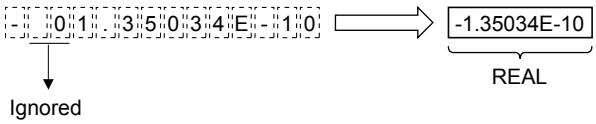



- When 2BH (+) is specified as the sign in the floating point format or when the sign is omitted, a character string is converted into a positive value. It is handled as negative value during conversion when the sign is set to 2DH (-).
- String data in the exponent format is handled as positive value during conversion when the sign of the exponent part is set to 2BH (+) or when the sign is omitted. When 2DH (-) is specified as the sign, a character string is converted into a negative value.
- When 20H (space) or 30H (0) exists between numbers except the first 0 in STRING type data, 20H or 30H is ignored during conversion.

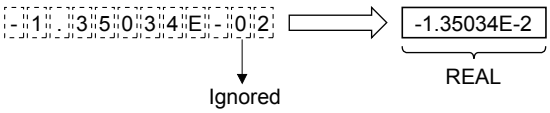
• Decimal point format




• Exponent format




- When 30H (0) exists between a number and "E" in STRING type data (exponent format), 30H is ignored during conversion.




- When 20H (space) is contained in character string, 20H is ignored during conversion.
- Up to 24 characters can be input as STRING type data. 20H (space) and 30H (0) in a character string are counted as one character respectively.
- A value input to (s) is the STRING type data value and within the following range.
  - Within the range of "30H" to "39H", "45H", "2BH", "2DH", "2EH", "20H" and "00H" in ASCII code

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

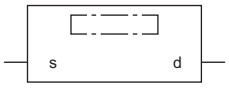
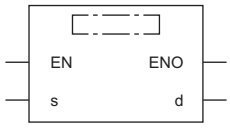
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | 00H does not exist in the corresponding device range starting from (s) |
| 3401H | Invalid data which cannot be converted to (s) are set.<br>• Any character other than "30(0)" to "39(9)" exists in the integer part or decimal part.<br>• 2EH (.) exists in two or more positions in the specified character string.<br>• Any character other than 45H (E), 65(e), 2B(+) , or 2D(-) exists in the specified exponent part.<br>• Two or more exponent parts of 45H (E) or 65(e) exist in the specified character string.<br>• Three or more digits of numerical values in the exponent parts are described in the specified character string.<br>• Two or more signs of exponent parts of 2B(+) or 2D(-) exist in the specified character string.<br>• Two or more signs of 2B(+) or 2D(-) exist in the integral part for the decimal point format and exist in the mantissa part for the exponent format in the specified character string.<br>• The number of characters after (s) is 0 or more than 24 |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 16.48 Converting STRING to TIME

## STRING_TO_TIME(_E)

These functions convert STRING type data to TIME type data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=STRING_TO_TIME(s);<br>[With EN/ENO]<br>d:=STRING_TO_TIME_E(EN,ENO,s); |

### Setting data

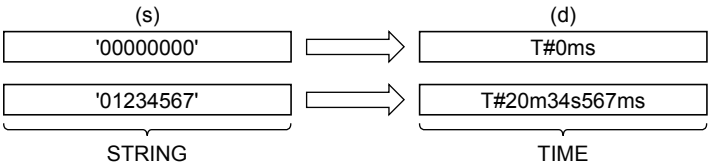
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(11) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(STRING_TO_TIME(_E)) | Output | Output variable | TIME |

### Processing details

#### ■Operation processing

- These functions convert the STRING type data input to (s) to TIME type data and output from (d).




- A value input to (s) is the STRING type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

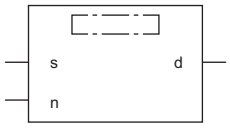
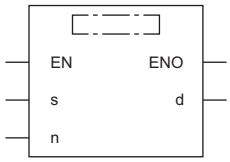
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | Values for each place of the ASCII code for input are other than "30H" to "39H", "20H", and "00H". |
| | Value of the ASCII code for input are outside the following range.<br>-2147483648 to 4147483647 |

# 16.49 Converting Bit Array to INT

## BITARR_TO_INT(_E)

These functions convert a bit array to INT type data for a specified number of bits.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=BITARR_TO_INT(s,n);<br>[With EN/ENO]<br>d:=BITARR_TO_INT_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(BitArr) | Input (Variables are available for element specification.) | Input variable | BOOL array element |
| n | Only a constant 4, 8, 12 or 16 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal , FALSE: Abnormal) | Output variable | BOOL |
| d(BITARR_TO_INT(_E)) | Output | Output variable | ANY16 |

### Processing details

#### ■Operation processing

- These functions convert the data for bits specified by (n) starting from the bit array element input to (s) to ANY 16 type data and output from (d).
- "0" is set to output bits beyond the specified number of bits.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

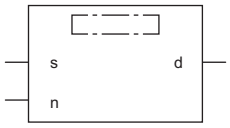
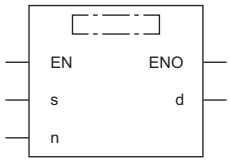
### Operation error

There is no operation error.

# 16.50 Converting Bit Array to DINT

## BITARR_TO_DINT(_E)

These functions convert a bit array to DINT type data for a specified number of bits.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=BITARR_TO_DINT(s,n)<br>[With EN/ENO]<br>d:=BITARR_TO_DINT_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(BitArr) | Input (Variables are available for element specification.) | Input variable | BOOL array element |
| n | Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(BITARR_TO_DINT(_E)) | Output | Output variable | ANY32 |

### Processing details

#### ■Operation processing

- These functions convert the data for bits specified by (n) starting from the bit array element input to (s) to ANY 32 type data and output from (d).
- "0" is set to output bits beyond the specified number of bits.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

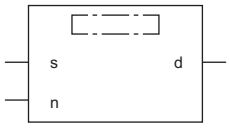
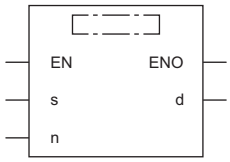
### Operation error

There is no operation error.

# 16.51 Converting INT to Bit Array

## INT_TO_BITARR(_E)

These functions output low-order (n) bits of INT type data to a bit array.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=INT_TO_BITARR(s,n);<br>[With EN/ENO]<br>d:=INT_TO_BITARR_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s | Input | Input variable | ANY16 |
| n | Only a constant 4, 8, 12 or 16 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INT_TO_BITARR(_E)) | Output (Variables are available for element specification.) | Output variable | BOOL array element |

### Processing details

#### ■Operation processing

- These functions output low-order (n) bits of ANY 16 type data specified to (s).
- Output bits beyond the specified number of bits are not changed.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

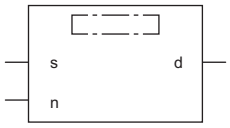
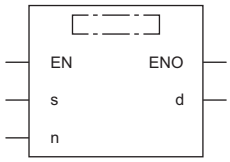
### Operation error

There is no operation error.

# 16.52 Converting DINT to Bit Array

## DINT_TO_BITARR(_E)

These functions output low-order (n) bits of DINT type data to a bit array.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=DINT_TO_BITARR(s,n);<br>[With EN/ENO]<br>d:=DINT_TO_BITARR_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s | Input | Input variable | ANY32 |
| n | Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DINT_TO_BITARR(_E)) | Output (Variables are available for element specification.) | Output variable | BOOL array element |

### Processing details

#### ■Operation processing

- These functions output low-order (n) bits of ANY 32 type data specified to (s) to (d).
- Output bits beyond the specified number of bits are not changed.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

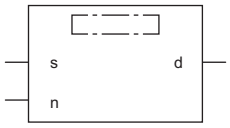
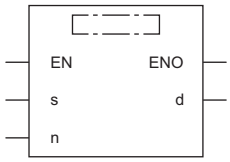
### Operation error

There is no operation error.

# 16.53 Bit Array Copy

## CPY_BITARR(_E)

These functions copy specified number of bits of a bit array.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=CPY_BITARR(s,n);<br>[With EN/ENO]<br>d:=CPY_BITARR_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(BitArrIn) | Input | Input variable | BOOL array element |
| n | Only a constant 4, 8, 12, 16, 20, 24, 28 or 32 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(CPY_BITARR(_E)) | Output | Output variable | BOOL array element |

### Processing details

#### ■Operation processing

- These functions output (n) bits of a bit array specified to (s) to (d).

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

### Operation error

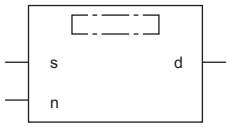
There is no operation error.

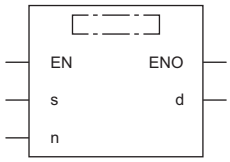
# 16.54 Reading the Specified Bit of Word Label

## GET_BIT_OF_INT(_E)

These functions reads the specified bit of the word label

| Ladder diagram | | Structured text | FBD/LD |
|---|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=GET_BIT_OF_INT(s,n);<br>[With EN/ENO]<br>d:=GET_BIT_OF_INT_E(EN,ENO,s,n); | Not supported |




### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s | Input | Input variable | ANY16 |
| n | Only a constant 0 to 15 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(GET_BIT_OF_INT(_E)) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions output (n)th bit of (s).

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

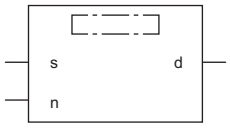
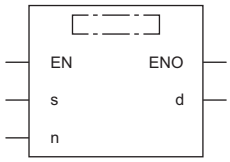
### Operation error

There is no operation error.

# 16.55 Writing the Specified Bit of Word Label

## SET_BIT_OF_INT(_E)

These functions writes the specified bit of the word label.

| Ladder diagram | | Structured text | FBD/LD |
|---|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=SET_BIT_OF_INT(s,n);<br>[With EN/ENO]<br>d:=SET_BIT_OF_INT_E(EN,ENO,s,n); | Not supported |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s | Input | Input variable | BOOL |
| n | Only a constant 0 to 15 can be specified. | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SET_BIT_OF_INT(_E)) | Output | Output variable | ANY16 |

### Processing details

#### ■Operation processing

- These functions write the BOOL value specified by (s) in the (n)th bit of (d).

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

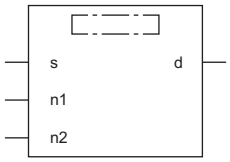
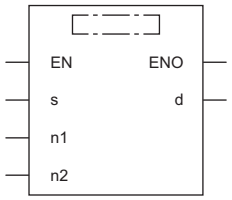
### Operation error

There is no operation error.

# 16.56 Copying the Specified Bit of Word Label

## CPY_BIT_OF_INT(_E)

These functions copy the specified bit of the word label to the one of another word label.

| Ladder diagram | | Structured text | FBD/LD |
|---|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=CPY_BIT_OF_INT(s,n1,n2);<br>[With EN/ENO]<br>d:=CPY_BIT_OF_INT_E(EN,ENO,s,n1,n2); | Not supported |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s | Input | Input variable | ANY16 |
| n1 | Bit specification of input variable (Only a constant 0 to 15 can be specified.) | Input variable | INT |
| n2 | Bit specification of output variable (Only a constant 0 to 15 can be specified.) | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(CPY_BIT_OF_INT(_E)) | Output | Output variable | ANY16 |

### Processing details

#### ■Operation processing

- These function copy the value of the (n1)th bit of the word specified by (s) to the (n2)th bit of (d).

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

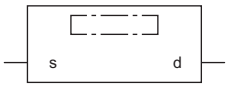
### Operation error

There is no operation error.

# 16.57 Unnecessary of Type Conversion

## GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

These functions output the input variable as the output variable type.

| Ladder diagram, FBD/LD | Structured text |
|---|---|
|  | d:=GET_BOOL_ADDR(s)<br>d:=GET_INT_ADDR(s);<br>d:=GET_WORD_ADDR(s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| s | Input | Input variable | ANY |
| d(GET_BOOL_ADDR / GET_INT_ADDR / GET_WORD_ADDR) | Output | Output variable | BOOL/INT/WORD |

### Processing details

#### ■Operation processing

- These functions output the input data variable as the output variable type according to the following table.

| General function | Input data type | Output data type |
|---|---|---|
| GET_BOOL_ADDR | BOOL<br>ARRAY OF BOOL | BOOL |
| GET_INT_ADDR | INT<br>DINT<br>WORD<br>REAL<br>TIME<br>STRING<br>ARRAY OF INT<br>ARRAY OF DINT<br>ARRAY OF WORD<br>ARRAY OF DWORD<br>ARRAY OF REAL<br>ARRAY OF TIME | INT |
| GET_WORD_ADDR | | WORD |

#### ■Operation result

The operation processing is executed. The operation output value is output from (d).

### Operation error

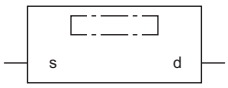
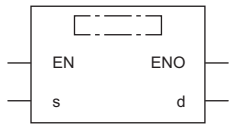
There is no operation error.

# 17 SINGLE NUMBER VARIABLE FUNCTIONS

## 17.1 Absolute Value

### ABS(_E)

These functions output the absolute value of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=ABS(s);<br>[With EN/ENO]<br>d:=ABS_E(EN,ENO,s); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ABS(_E)) | Output | Output variable | ANY_NUM |

#### Processing details

##### ■Operation processing

- These functions output the absolute value of the INT, DINT, or REAL type data input to (s) in the same data type as (s) from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  B=|A|
- A value input to (s) is the INT, DINT, or REAL type data value.
- When -32768 is input while the data type of (s) is INT, -32768 is output from (d).
- When -2147483648 is input while the data type of (s) is DINT, -2147483648 is output from (d). (An operation error does not occur. "ABS_E" outputs "TRUE" from output variable ENO.)

##### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

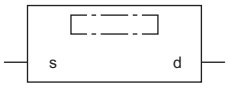
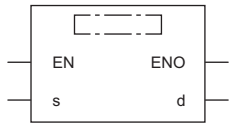
- When (s) is REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The data specified by (s) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 17.2 Square Root

## SQRT(_E)

These functions output the square root of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=SQRT(s);<br>[With EN/ENO]<br>d:=SQRT_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SQRT(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions output the square root of the REAL type data input to (s) from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".

$$B=\sqrt{A}$$

- A value input to (s) is the REAL type data value and within the positive value range.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

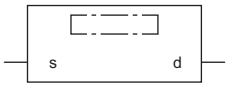
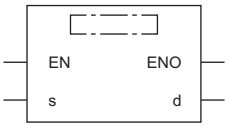
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | A negative value is input. |

# 17.3 Natural Logarithm Operation

## LN(_E)

These functions output the natural logarithm operation result of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=LN(s);<br>[With EN/ENO]<br>d:=LN_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(LN(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the logarithm whose base is "e" of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=\log_e A$
- In the natural logarithm operation, the base "e" is set to "2.71828".

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

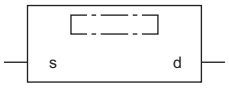
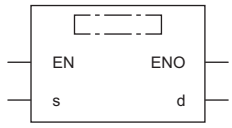
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | A negative value is input. |
| | The data after conversion is other than $-3.40282^{+38}$ to $-1.17549^{-38}$, 0, or $1.17549^{-38}$ to $3.40282^{+38}$. |

# 17.4 Calculating the Common Logarithm

## LOG(_E)

These functions output the operation result of the common logarithm (the logarithm whose base is 10) of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=LOG(s);<br>[With EN/ENO]<br>d:=LOG_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output condition (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(LOG(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the logarithm whose base is "10" of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=\log_{10}A$
- A value input to (s) is the REAL type data value.
- Only a positive value can be set in (s). (The logarithm operation cannot be executed for a negative value).
- When the operation result is -0 or underflow occurs, the operation result is regarded as 0.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

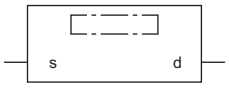
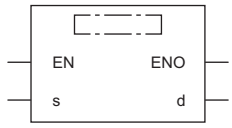
• When (s) is REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value specified in (s) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | The value of (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |
| 3405H | Data outside the allowable range was set to (s).<br>• A negative value is specified.<br>• "0" is specified. |

# 17.5 Exponential Operation

## EXP(_E)

These functions output the exponential operation result of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=EXP(s);<br>[With EN/ENO]<br>d:=EXP_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(EXP(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the exponent of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=e^{A}$
- In the exponential operation, the base "e" is set to "2.71828".
- A value input to (s) is the REAL type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

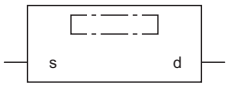
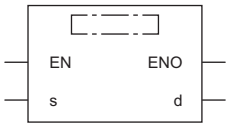
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3403H | The data after conversion is not $-3.40282^{+38}$ to $-1.17549^{-38}$, or $1.17549^{-38}$ to $3.40282^{+38}$. |

# 17.6 Sine Operation

## SIN(_E)

These functions output the sine of the angle of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=SIN(s);<br>[With EN/ENO]<br>d:=SIN_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SIN(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the sine of the angle of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  B=SIN A
- A value (angle) input to (s) is the REAL type data value. Input a value in radians (angle×π/180).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

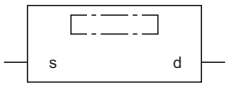
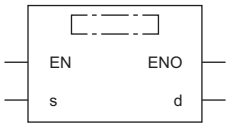
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | A negative value is input. |

# 17.7 Cosine Operation

## COS(_E)

These functions output the cosine of the angle of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=COS(s);<br>[With EN/ENO]<br>d:=COS_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(COS(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the cosine of the angle of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  B=COS A
- A value (angle) input to (s) is the REAL type data value. Input a value in radians (angle×π/180).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

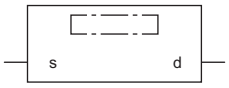
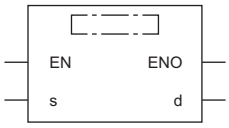
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | A negative value is input. |

# 17.8 Tangent Operation

## TAN(_E)

These functions output the tangent of the angle of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=TAN(s);<br>[With EN/ENO]<br>d:=TAN_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(TAN(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the tangent of the angle data of the REAL type data (angle) input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  B=TAN A
- Even when the input value is π/2 radian or (3/2) π radian, no error occurs because an operation error occurs in a radian value.
- A value (angle) input to (s) is the REAL type data value. Input a value in radians (angle×π/180).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

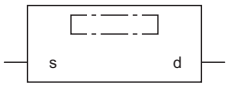
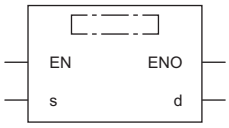
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | A negative value is input. |

# 17.9 Arc Sine Operation

## ASIN(_E)

These functions output the arc sine value of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=ASIN(s);<br>[With EN/ENO]<br>d:=ASIN_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ASIN(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the arc sine of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=SIN^{-1} A$
- A value input to (s) is the REAL type data value and within the following range.
  ASIN(_E): -1.0 to 1.0
- A value (angle) in radians (angle$\times\pi$/180) is output from (d).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

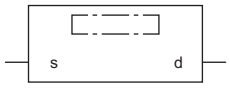
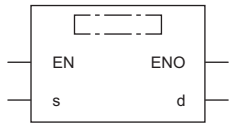
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | A negative value is input. |
| 3405H | A value input by these functions is other than -1.0 to 1.0. |

# 17.10 Arc Cosine Operation

## ACOS(_E)

These functions output the arc cosine value of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=ACOS(s);<br>[With EN/ENO]<br>d:=ACOS_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ACOS(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the arc cosine of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=COS^{-1} A$
- A value input to (s) is the REAL type data value and within the following range.
  ACOS(_E): -1.0 to 1.0
- A value (angle) in radians (angle$\times\pi$/180) is output from (d).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

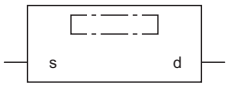
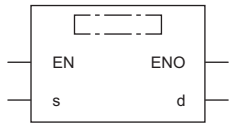
### Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | A negative value is input. |
| 3405H | A value input by these functions is other than -1.0 to 1.0. |

# 17.11 Arc Tangent Operation

## ATAN(_E)

These functions output the arc tangent value of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=ATAN(s);<br>[With EN/ENO]<br>d:=ATAN_E(EN,ENO,s); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | REAL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ATAN(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions calculate the arc tangent value of the REAL type data input to (s), and output from (d).
- These functions are expressed as follows when the input value is "A" and the output operation result is "B".
  $B=TAN^{-1} A$
- A value input to (s) is the REAL type data value and within the following range.
  ATAN(_E): $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$
- A value (angle) in radians (angle$\times\pi$/180) is output from (d).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

### Operation error

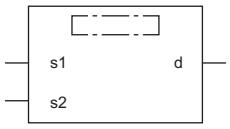
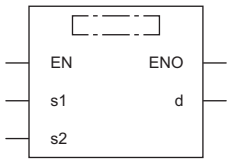
| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | -0 is input. |

# 18 ARITHMETIC OPERATION FUNCTIONS

## 18.1 Addition

### ADD(_E)

These functions output the sum of input values ((s1) + (s2) + ... + (s28)).

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=ADD(s1,s2);<br>[With EN/ENO]<br>d:=ADD_E(EN,ENO,s1,s2); |
|  |  | |

*1 The input variable "s" can be changed in the range of 2 to 28.

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ADD(_E)) | Output | Output variable | ANY_NUM |

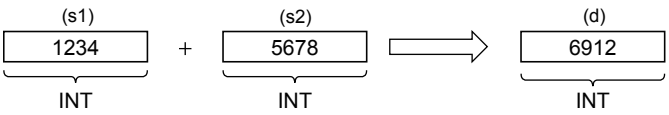
#### Processing details

##### ■Operation processing

- These functions add the INT, DINT, or REAL type data ((s1) + (s2) + ... + (s28)) input to (s1) to (s28), and output from (d) in the same data type as (s).

Ex.

Data type is the INT type




- A value input to (s1) to (s28) is the INT, DINT, or REAL type data value.
- If an underflow and an overflow occur in the operation result, the result will be output as follows from (d).

| Data type is INT | Data type is DINT | Data type is REAL |
|---|---|---|
| • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "ADD_E" outputs "TRUE" from ENO.<br>[Example 1]<br>32767+2=1<br>(7FFFH)+(0002H)=0001H<br>The most significant bit becomes 0, and the carry flags (SM716 and SM8022) turn on.<br>[Example 2]<br>-32768+(-2)=-1<br>(8000H)+(FFFEH)=(FFFFH)<br>The most significant bit becomes 1, and the borrow flag (SM8021) turns on. | • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "ADD_E" outputs "TRUE" from ENO.<br>[Example 1]<br>2147483647+2=1<br>(7FFFFFFFH)+(0002H)=(00000001H)<br>The most significant bit becomes 1, and the carry flags (SM716 and SM8022) turn on.<br>[Example 2]<br>-2147483648+(-2)=-1<br>(80000000H)+(FFFEH)=(7FFFFFFFH)<br>The most significant bit becomes 1, and the borrow flag (SM8021) turns on. | An operation error occurs and an undefined value is output. |

- When the operation result is 0, the zero flag (SM8020) turns on.

## ■Operation result

### 1. Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

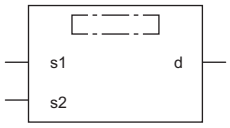
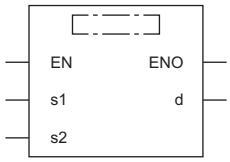
• (s1) to (s28) are REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The data specified by (s1) to (s28) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 18.2 Multiplication

## MUL(_E)

These functions output the product input values ((s1)×(s2)× ... ×(s28)).

| Ladder diagram, FBD/LD[*1] | | Structured text[*1] |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=MUL(s1,s2);<br>[With EN/ENO]<br>d:=MUL_E(EN,ENO,s1,s2); |
|  |  | |

*1 The input variable "s" can be changed in the range of 2 to 28.

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MUL(_E)) | Output | Output variable | ANY_NUM |

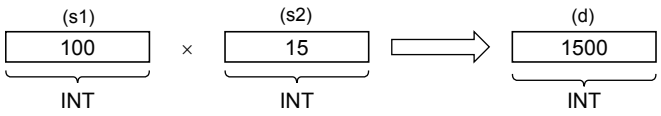
### Processing details

#### ■Operation processing

- These functions multiply the INT, DINT, or REAL type data input to (s1) to (s28) ((s1)×(s2)× ...×(s28)), and output the operation result from (d) in the same data type as (s).

Ex.

Data type is INT




- A value input to (s1) to (s28) is the INT, DINT, or REAL type data value.
- If an underflow occurs in the operation result, the result will be output as follows from (d).

| Data type is INT | Data type is DINT | Data type is REAL |
|---|---|---|
| • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "MUL_E" outputs "TRUE" from ENO.<br>• Even when the operation result exceeds the INT type data range, the INT type data is output. (The operation result is the DINT type, however, the output data is the INT type data with high-order 16 bits deleted.)<br>• When the operation result exceeds the INT type data, convert an input value into the DINT type data by INT_TO_DINT then perform the operation. | • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "MUL_E" outputs "TRUE" from ENO.<br>• Even when the operation result exceeds the DINT type data range, the DINT type data is output. (The operation result is the 64-bit data, however, the output data is the DINT type data with high-order 32 bits deleted.)<br>• When the operation result exceeds the DINT type data, convert an input value into the REAL type data by DINT_TO_REAL then perform the operation. | An operation error occurs and an undefined value is output. |

- When the operation result is 0, the zero flag (SM8020) turns on.

## ■Operation result

### 1. Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

**Point**

When the operation result exceeds the data type range, convert the data type of an input value then perform the operation.

## Operation error

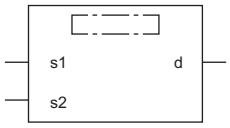
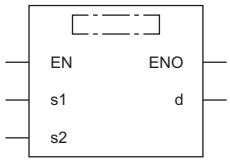
- (s1) to (s28) are REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The data specified by (s1) to (s28) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 18.3 Subtraction

## SUB(_E)

These functions output the difference of input values ((s1) - (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=SUB(s1,s2);<br>[With EN/ENO]<br>d:=SUB_E(EN,ENO,s1,s2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SUB(_E)) | Output | Output variable | ANY_NUM |

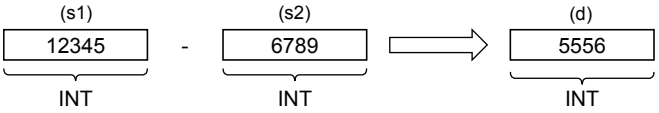
### Processing details

#### ■Operation processing

- These functions subtract the INT, DINT, or REAL type data input to (s1) and (s2) ((s1)-(s2)), and output the operation result from (d) in the same data type as (s).

Ex.

Data type is INT




- A value input to (s1) and (s2) is the INT, DINT, or REAL type data value.
- If an underflow and an overflow occur in the operation result, the result will be output as follows from (d).

| Data type is INT | Data type is DINT | Data type is REAL |
|---|---|---|
| • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "SUB_E" outputs "TRUE" from ENO.<br>[Example 1]<br>32767-(-2)=1<br>(7FFFH)-(0002H)=(0001H)<br>The most significant bit becomes 1, and the carry flags (SM716 and SM8022) turn on.<br>[Example 2]<br>-32768-2=-1<br>(8000H)-(0002H)=(FFFFH)<br>The most significant bit becomes 0, and the borrow flag (SM8021) turns on. | • Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. "SUB_E" outputs "TRUE" from ENO.<br>[Example 1]<br>2147483647-(-2)=-2147483647<br>(7FFFFFFFH)-(FFFEH)=(80000001H)<br>The most significant bit becomes 1, and a negative value is output.<br>[Example 2]<br>-2147483648-2=2147483646<br>(80000000H)-(0002H)=(7FFFFFFEH)<br>The most significant bit becomes 0, and a positive value is output. | An operation error occurs and an undefined value is output. |

- When the operation result is 0, the zero flag (SM8020) turns on.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

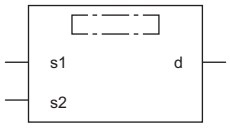
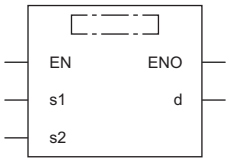
• (s1) and (s2) are REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| | The data specified by (s2) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 18.4 Division

## DIV(_E)

These functions output the quotient of input values ((s1) ÷ (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=DIV(s1,s2);<br>[With EN/ENO]<br>d:=DIV_E(EN,ENO,s1,s2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) | Dividend | Input variable | ANY_NUM |
| s2(IN2) | Divisor | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DIV(_E)) | Output | Output variable | ANY_NUM |

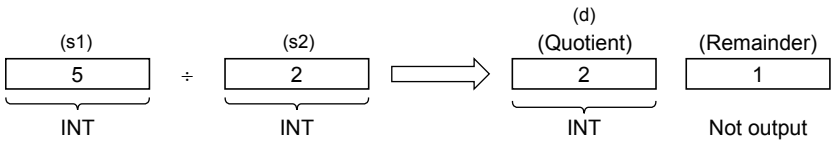
### Processing details

#### ■Operation processing

- These functions divide the INT, DINT, or REAL type data input to (s1) and (s2) ((s1) ÷ (s2)), and output the operation result from (d) in the same data type as (s).

**Ex.**

Data type is INT




- A value input to (s1) and (s2) is the INT, DINT, or REAL type data value. (However, input other than 0 to (s2).)
- When the operation result is 0, the zero flag (SM8020) turns on. When the operation result exceeds "32767" (16-bit operation) or "2147483647" (32-bit operation), the carry flag (SM8022) turns on.

## ■Operation result

### 1. Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

• (s1) and (s2) are INT

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The value (divisor) specified by (s2) is 0. |

• (s1) and (s2) are DINT

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The value (divisor) specified by (s2) is 0. |

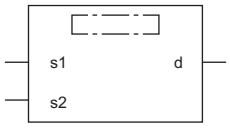
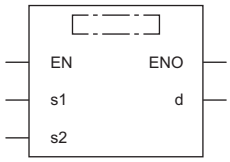
• (s1) and (s2) are REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The value (divisor) specified by (s2) is 0. |
| 3402H | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| | The data specified by (s2) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | (d) exceeds the following range. (An overflow has occurred.)<br>$\|(d)\|<2^{128}$ |

# 18.5 Remainder

## MOD(_E)

These functions output the remainder of input values ((s1) ÷ (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=MOD(s1,s2);<br>[With EN/ENO]<br>d:=MOD_E(EN,ENO,s1,s2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) | Dividend | Input variable | ANY_INT |
| s2(IN2) | Divisor | Input variable | ANY_INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MOD(_E)) | Output | Output variable | ANY_INT |

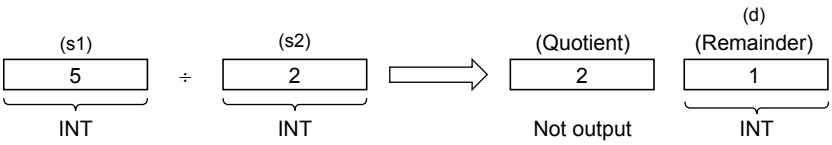
### Processing details

#### ■Operation processing

- These functions divide the INT or DINT type data input to (s1) and (s2) ((s1) ÷ (s2)), and output the remainder from (d) in the same data type as (s).

**Ex.**

Data type is INT




- A value input to (s1) and (s2) is the INT and DINT type data value. (However, input other than 0 to (s2).)

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

- (s1) and (s2) are INT

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The value (divisor) specified by (s2) is 0. |

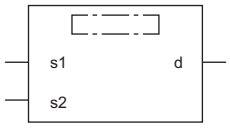
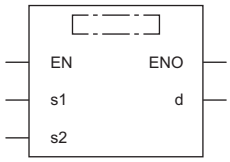
- (s1) and (s2) are DINT

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | The value (divisor) specified by (s2) is 0. |

# 18.6 Exponentiation

## EXPT(_E)

These functions output the exponentiation of an input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=EXPT(s1,s2);<br>[With EN/ENO]<br>d:=EXPT_E(EN,ENO,s1,s2); |

### Setting data

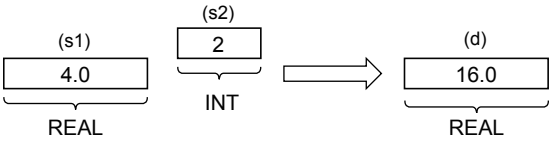
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) | Cardinal number | Input variable | REAL |
| s2(IN2) | Exponent | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(EXPT(_E)) | Output | Output variable | REAL |

### Processing details

#### ■Operation processing

- These functions raise the REAL type data input to (s1) by INT, DINT, or REAL specified by (s2), and output the operation result from (d).




#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

• (s1) is the REAL type and (s2) is the INT type

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value of (s1) is outside the following range.<br>0, $2^{-126} \leq |(s1)| < 2^{128}$ |
| | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | The operation result is within the following range.<br>$2^{128} \leq$ \|operation result\| |

• (s1) is the REAL type and (s2) is the DINT type

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value of (s1) is outside the following range.<br>0, $2^{-126} \leq |(s1)| < 2^{128}$ |
| | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | The operation result is within the following range.<br>$2^{128} \leq$ \|operation result\| |

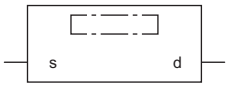
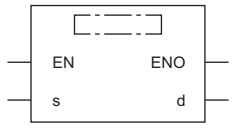
• (s1) and (s2) are REAL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value of (s1) is outside the following range.<br>0, $2^{-126} \leq |(s1)| < 2^{128}$ |
| | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value of (s2) is outside the following range.<br>0, $2^{-126} \leq |(s2)| < 2^{128}$ |
| | The data specified by (s2) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3403H | The operation result is within the following range.<br>$2^{128} \leq$ \|operation result\| |

# 18.7 Move Operation

## MOVE(_E)

These functions output the assignment of input values.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=MOVE(s);<br>[With EN/ENO]<br>d:=MOVE_E(EN,ENO,s); |

### Setting data

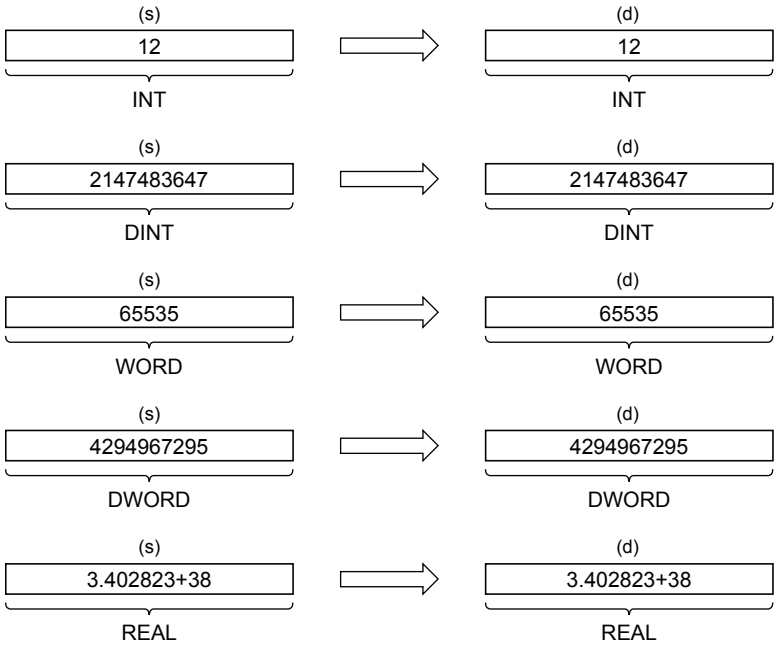
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MOVE(_E)) | Output | Output variable | ANY |

### Processing details

#### ■Operation processing

- These functions assign the value of variable specified to (s) to the variable specified to (d).
- BOOL, INT, DINT, WORD, DWORD, REAL, STRING, TIME, structure, or array type can be specified for (s) and (d). Specify the same data type for (s) and (d).



## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

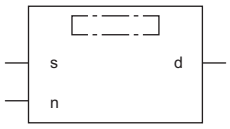
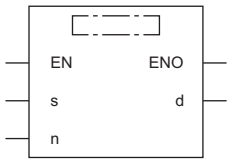
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 19 BIT SHIFT FUNCTIONS

## 19.1 n-bit Left Shift

### SHL(_E)

These functions shift an input value leftward by (n) bits and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=SHL(s,n);<br>[With EN/ENO]<br>d:=SHL_E(EN,ENO,s,n); |
|  |  | |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| n(N) | Number of shift bits | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SHL(_E)) | Output | Output variable | ANY_BIT |

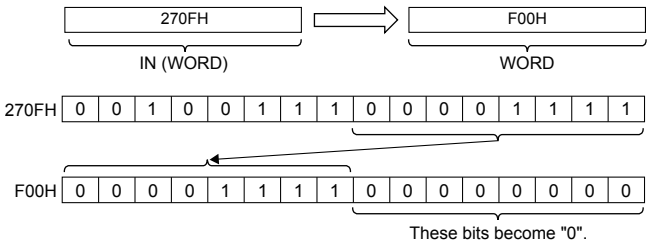
#### Processing details

##### ■Operation processing

- These functions shift the WORD or DWORD type data input to (s) left by (n) bits and output the result in the same data type as (s) from (d).
- The number input in (n) is used as the number of left-shift bits.

Ex.

When the data type of (s) is WORD and 8 is input in (n)




- "0" is set to "n" bits from the least significant bit.
- A value input to (n) is the WORD or DWORD type data value.
- A value input to (n) (Number of shift bits) is the INT type data value and within the following range.

| When the data type of (s) is WORD | When the data type of (s) is DWORD |
|---|---|
| A value in (n) is within 0 to 15.<br>The lower 4-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 6 | A value in (n) is within 0 to 31.<br>The lower 5-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 22 |

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

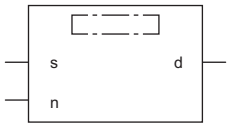
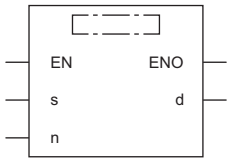
## Operation error

There is no operation error.

# 19.2 n-bit Right Shift

## SHR(_E)

These functions shift an input value rightward by (n) bits and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=SHR(s,n);<br>[With EN/ENO]<br>d:=SHR_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| n(N) | Number of shift bits | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SHR(_E)) | Output | Output variable | ANY_BIT |

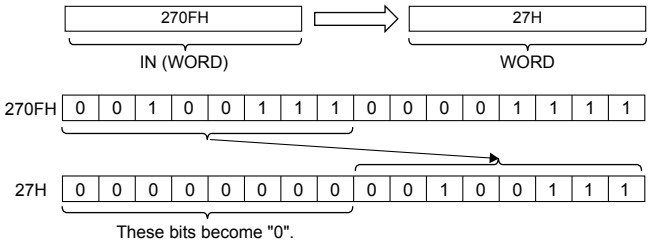
### Processing details

#### ■Operation processing

- These functions shift the WORD or DWORD type data input to (s) right by (n) bits and output the result in the same data type as (s) from (d).
- The number input in (n) is used as the number of right-shift bits.

Ex.

When the data type of (s) is WORD and 8 is input in (n)




- "0" is set to "n" bits from the most significant bit.
- A value input to (n) is the WORD or DWORD type data value.
- A value input to (n) (Number of shift bits) is the INT type data value and within the following range.

| When the data type of (s) is WORD | When the data type of (s) is DWORD |
|---|---|
| A value in (n) is within 0 to 15.<br>The lower 4-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 6 | A value in (n) is within 0 to 31.<br>The lower 5-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 22 |

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

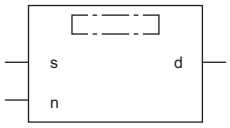
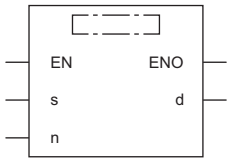
## Operation error

There is no operation error.

# 19.3 n-bit Left Rotation

## ROL(_E)

These functions rotate an input value leftward by (n) bits and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=ROL(s,n);<br>[With EN/ENO]<br>d:=ROL_E(EN,ENO,s,n); |

### Setting data

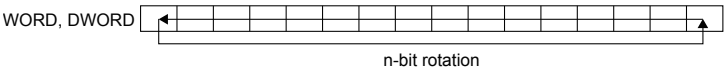
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| n(N) | Number of shift bits | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ROL(_E)) | Output | Output variable | ANY_BIT |

### Processing details

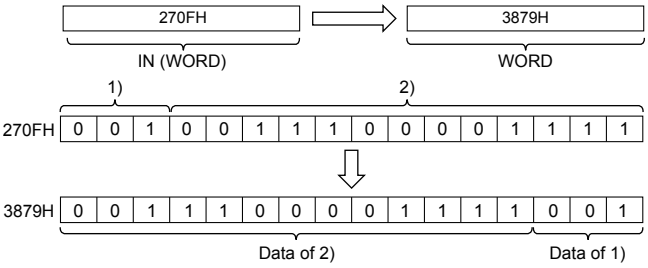
#### ■Operation processing

- These functions rotate the WORD or DWORD type data input to (s) left by (n) bits and output the result in the same data type as (s) from (d).
- The number input in (n) is used as the number of left-rotation bits.




Ex.

When the data type of (s) is WORD and 3 is input in (n) (The bits are rotated left by 3 bits.)




- A value input to (n) is the WORD or DWORD type data value.
- A value input to (n) (Number of shift bits) is the INT type data value and within the following range.

| When the data type of (s) is WORD | When the data type of (s) is DWORD |
|---|---|
| A value in (n) is within 0 to 15.<br>The lower 4-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 6 | A value in (n) is within 0 to 31.<br>The lower 5-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 22 |

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

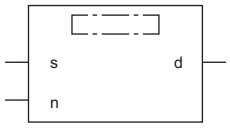
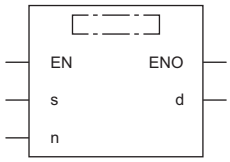
## Operation error

There is no operation error.

# 19.4 n-bit Right Rotation

## ROR(_E)

These functions rotate an input value rightward by (n) bits and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=ROR(s,n);<br>[With EN/ENO]<br>d:=ROR_E(EN,ENO,s,n); |
|  |  | |

### Setting data

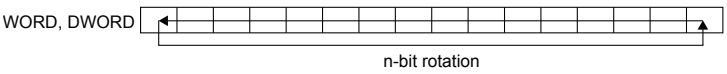
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| n(N) | Number of shift bits | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ROR(_E)) | Output | Output variable | ANY_BIT |

### Processing details

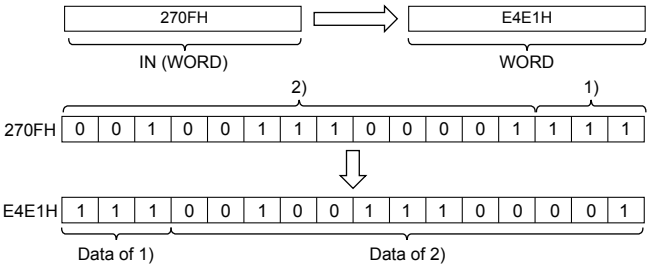
#### ■Operation processing

- These functions rotate the WORD or DWORD type data input to (s) right by (n) bits and output the result in the same data type as (s) from (d).
- The number input in (n) is used as the number of right-rotation bits.




**Ex.**

When the data type of (s) is WORD and 3 is input in (n) (The bits are rotated right by 3 bits.)




- A value input to (n) is the WORD or DWORD type data value.
- A value input to (n) (Number of shift bits) is the INT type data value and within the following range.

| When the data type of (s) is WORD | When the data type of (s) is DWORD |
|---|---|
| A value in (n) is within 0 to 15.<br>The lower 4-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 6 | A value in (n) is within 0 to 31.<br>The lower 5-bit data of the value in (n) is used.<br>[Example]<br>When the input value is 6: 6<br>When the input value is 22: 22 |

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

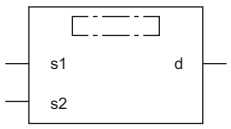
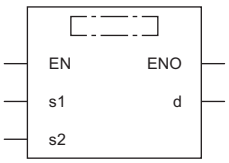
There is no operation error.

# 20 STANDARD BITWISE BOOLEAN FUNCTIONS

## 20.1 AND Operation, OR Operation, XOR Operation

### AND(_E), OR(_E), XOR(_E)

- AND(_E): Outputs the logical product of input values.
- OR(_E): Outputs the logical sum of input values.
- XOR(_E): Outputs the exclusive logical sum of input values.

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=AND(s1,s2);<br>d:=OR(s1,s2);<br>d:=XOR(s1,s2);<br>[With EN/ENO]<br>d:=AND_E(EN,ENO,s1,s2);<br>d:=OR_E(EN,ENO,s1,s2);<br>d:=XOR_E(EN,ENO,s1,s2); |

*1 The input variable "s" can be changed in the range of 2 to 28.

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(AND(_E) / OR(_E) / XOR(_E)) | Output | Output variable | ANY_BIT |

#### Processing details

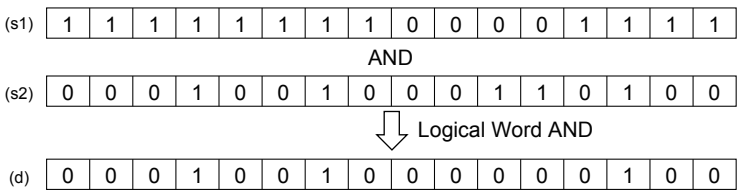
##### ■Operation processing

***1.*** AND(_E)

- These functions perform the logical AND on the BOOL, WORD, or DWORD type data input in (s1) to (s28) bit by bit, and output the operation result from (d) in the same data type as (s).

**Ex.**

When the data type is WORD

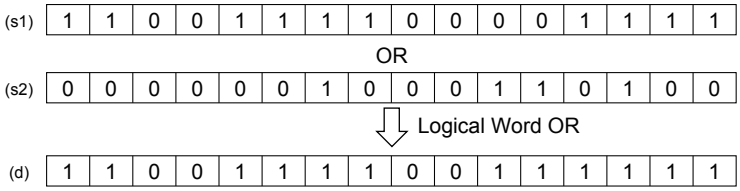


### 2. OR(_E)

- These functions perform the logical OR on the BOOL, WORD, or DWORD type data input in (s1) to (s28) bit by bit, and output the operation result from (d) in the same data type as (s).

**Ex.**

When the data type is WORD

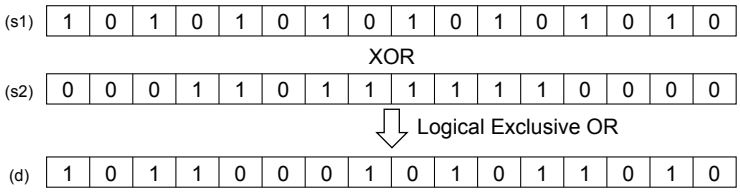



### 3. XOR(_E)

- These functions perform the exclusive logical OR on the BOOL, WORD, or DWORD type data input in (s1) to (s28) bit by bit, and output the operation result from (d) in the same data type as (s).

**Ex.**

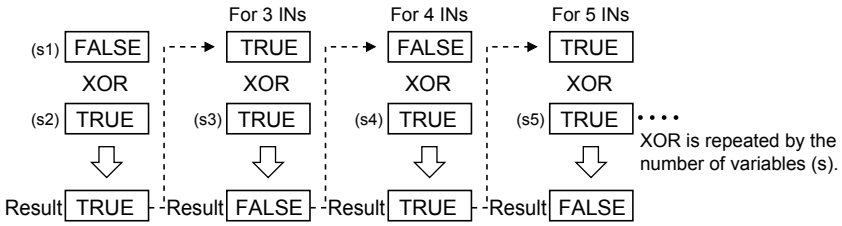
When the data type is WORD




- When three or more variables (s) exist, XOR is performed between (s1) and (s2) first, and XOR is successively performed between the result and (s3). When the expression includes (s4), XOR is performed between the result of XOR with (s3) and (s4). In this manner, XOR is repeated by the number of variables (s) in the order with (s5), (s6), and so on.

**Ex.**

When the data type is BOOL




## ■Operation result

### 1. Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

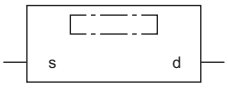
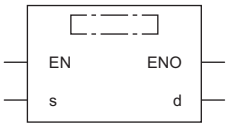
## Operation error

There is no operation error.

# 20.2 Logical Negation

## NOT(_E)

These functions output the logical negation of input values.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=NOT(s);<br>[With EN/ENO]<br>d:=NOT_E(EN,ENO,s); |
|  |  | |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | ANY_BIT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(NOT(_E)) | Output | Output variable | ANY_BIT |

### Processing details

#### ■Operation processing

- These functions calculate the logical negation for each bit of the BOOL, WORD, or DWORD type data input in (s), and output the operation result from (d) in the same data type as (s).

Ex.

When the data type is WORD

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (s) | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | | | | | NOT | | | | | | | | |
| (d) | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

- A value input to (s) is the BOOL, WORD, or DWORD type data value.

#### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

### Operation error

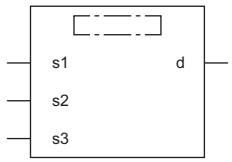
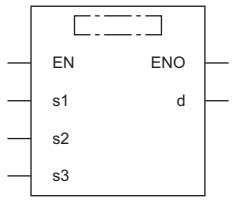
There is no operation error.

# 21 SELECTION FUNCTIONS

## 21.1 Selection

### SEL(_E)

These functions output a selected input value.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=SEL(s1,s2,s3);<br>[With EN/ENO]<br>d:=SEL_E(EN,ENO,s1,s2,s3); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(G) | Output condition (TRUE: Output s3, FALSE: Output s2) | Input variable | BOOL |
| s2(IN0) | Input | Input variable | ANY |
| s3(IN1) | | | |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SEL(_E)) | Output | Output variable | ANY |

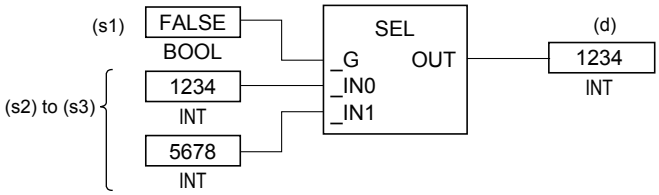
#### Processing details

##### ■Operation processing

- These functions output a value input to (s2) and (s3) according to a value input to (s1) in the same data type as (s2) and (s3) from (d).
- When FALSE(=0) is input to (s1), these functions output an input value of (s2) from (d).
- When TRUE(=1) is input to (s1), these functions output an input value of (s3) from (d).

Ex.

The data type of (s2) and (s3) is the INT type ((s2) and (s3) of an argument correspond to the bit value of (s1) (0 or 1).)




- A value input to (s1) is the BOOL type data value.
- A data value of the BOOL, INT, DINT, WORD, DWORD, REAL, STRING, TIME, structure, or array type can be input to (s2) and (s3).

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

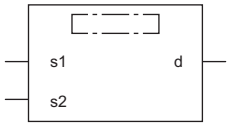
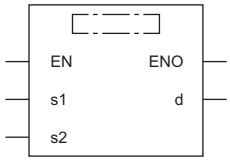
• (s2) and (s3) are the STRING type

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | "00H" is not set to a label specified by (s2) or devices from the device number to end device number of corresponding device. |
| | "00H" is not set to a label specified by (s3) or devices from the device number to the end device number of corresponding device. |
| 3406H | The specified character string cannot be stored in a label specified by (d) or devices from the device number to the end device number of corresponding device. |

# 21.2 Selecting Maximum/Minimum Value

## MAX(_E), MIN(_E)

- MAX(_E): These functions output the maximum value of an input value.
- MIN(_E): These functions output the minimum value of an input value.

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=MAX(s1,s2);<br>d:=MIN(s1,s2);<br>[With EN/ENO]<br>d:=MAX_E(EN,ENO,s1,s2);<br>d:=MIN_E(EN,ENO,s1,s2); |

*1 The input variable "s" can be changed in the range of 2 to 28.



### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | ANY_SIMPLE |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MAX(_E) / MIN(_E)) | Output | Output variable | ANY_SIMPLE |

### Processing details

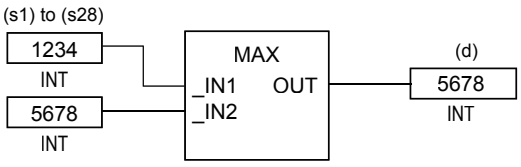
#### ■Operation processing

- MAX(_E)

  These functions output the maximum value of the BOOL, INT, DINT, WORD, DWORD, DWORD, REAL, STRING, or TIME type data input to (s1) to (s28) in the same data type as (s) from (d).

Ex.

Data type is INT

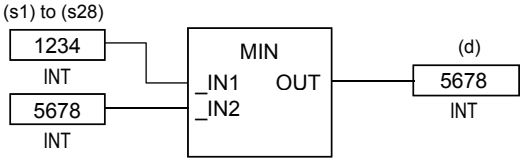



- MIN(_E)

  These functions output the minimum value of the BOOL, INT, DINT, WORD, DWORD, DWORD, REAL, STRING, or TIME type data input to (s1) to (s28) in the same data type as (s) from (d).

Ex.

Data type is INT




- A data value of the BOOL, INT, DINT, WORD, DWORD, REAL, STRING, or TIME type can be input to (s1) to (s28).
- The number of pins for (s) can be changed in the range of 2 to 28.

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

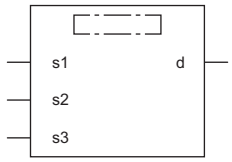
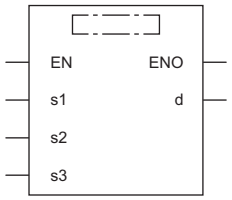
• (s1) to (s28) are STRING

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) to (s28) and later, "00H" does not exist. |
| 3405H | The character string specified by (s1) to (s28) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 21.3 Limit Control

## LIMIT(_E)

These functions output an input value controlled with the upper and lower limits.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=LIMIT(s1,s2,s3);<br>[With EN/ENO]<br>d:=LIMIT_E(EN,ENO,s1,s2,s3); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(MIN) | Lower limit value (minimum output threshold value) | Input variable | ANY_SIMPLE |
| s2(IN) | Input value to be controlled with the upper and lower limits | Input variable | ANY_SIMPLE |
| s3(MX) | Upper limit value (maximum output threshold value) | Input variable | ANY_SIMPLE |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(LIMIT(_E)) | Output | Output variable | ANY_SIMPLE |

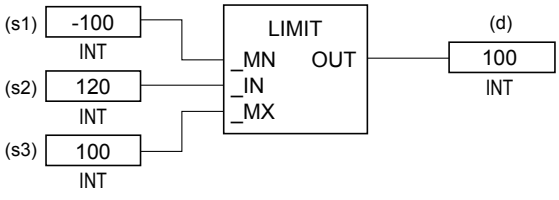
### Processing details

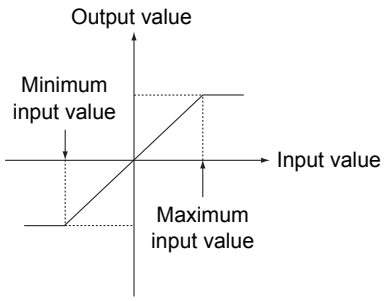
#### ■Operation processing

- These functions output an input value according to the BOOL, INT, DINT, WORD, DWORD, REAL, STRING, or TIME type data input to (s1), (s2), and (s3) in the same data type as (s1), (s2), and (s3) from (d).
  - When the input value of (s2) is larger than the one of (s3), these functions output the input value of (s3) from (d).
  - When the input value of (s2) is smaller than the one of (s1), these functions output the input value of (s1) from (d).
  - When the input value of (s1) ≤ the input value of (s2) ≤ the input value of (s3), these functions output the input value of (s2) from (d).

**Ex.**

Data type is INT




- A data value of the BOOL, INT, DINT, WORD, DWORD, REAL, STRING, or TIME type can be input to (s1), (s2), and (s3). (The input value of (s1) must be smaller than the one of (s3).)

## ■Operation result

### 1. Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### 2. Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

• (s1), (s2), and (s3) are INT or WORD

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

• (s1), (s2), and (s3) are DINT, DWORD, or TIME

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s2). |

• (s1), (s2), and (s3) are BOOL

| Error code (SD0/SD8067) | Description |
|---|---|
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s3). |

• (s1), (s2), and (s3) are the REAL type

| Error code (SD0/SD8067) | Description |
|---|---|
| 3402H | The value of (s1) is outside the following range.<br>0, $2^{-126} \leq \|(s1)\| < 2^{128}$ |
| | The data specified by (s1) is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value of (s2) is outside the following range.<br>0, $2^{-126} \leq \|(s2)\| < 2^{128}$ |
| | The data specified by (s2) is -0, denormalized number, NaN (not a number), or ±∞. |
| | The value of (s3) is outside the following range.<br>0, $2^{-126} \leq \|(s3)\| < 2^{128}$ |
| | The data specified by (s3) is -0, denormalized number, NaN (not a number), or ±∞. |
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s3). |

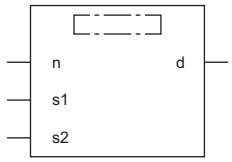
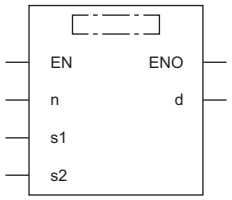
• (s1), (s2), and (s3) are STRING

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | "00H" is not set to a label specified by (s1), (s2), and (s3) or devices from specified device number to the end device number of corresponding device. |
| 3405H | The lower limit value specified by (s1) is greater than the upper limit value specified by (s3). |
| | The character strings specified by (s1), (s2), and (s3) have more than 16383 characters. |
| 3406H | The specified character string cannot be stored in a label specified by (d) or devices from specified device number to the end device number of corresponding device. |

# 21.4 Multiplexer

## MUX(_E)

These functions output one of multiple input values.

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=MUX(n,s1,s2);<br>[With EN/ENO]<br>d:=MUX_E(EN,ENO,n,s1,s2); |
|  |  | |

*1 The input variable "s" can be changed in the range of 2 to 28.

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| n(K) | Output value selection | Input variable | INT |
| s1(IN0) to s28(IN27) | Input | Input variable | ANY |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MUX(_E)) | Output | Output variable | ANY |

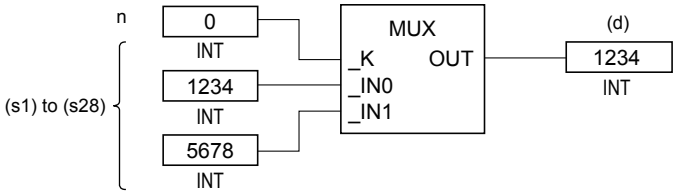
### Processing details

#### ■Operation processing

- These functions output one of values input to (s1) to (s28) according to the input value of (n) in the same data type as (s) from (d).
- When 0 is input to (n), these functions output a value input to (s1) from (d).
- When (n)-1 is input to (n), these functions output a value input to (sn) from (d).

Ex.

Data type is INT




- When a value input to (n) is outside the pin number range for (s), these functions output an indefinite value from (d). (An operation error does not occur. "MUX_E" outputs "FALSE" from ENO).
- A value input to (n) is the INT type data value and within the range from 0 to 27. (The value must be in the pin number range for (s).)
- A data value of the BOOL, INT, DINT, WORD, DWORD, REAL, STRING, TIME, structure, or array type can be input to (s).

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

• (s1) to (s28) are STRING

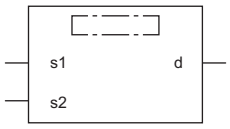
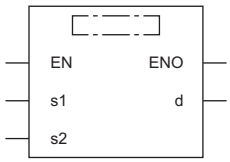
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) to (s28) and later, "00H" does not exist. |
| 3405H | The character string specified by (s1) to (s28) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 22 COMPARISON FUNCTIONS

## 22.1 Compare

### GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)

These functions output the data comparison result of input values.

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=GT(s1,s2);<br>d:=GE(s1,s2);<br>d:=EQ(s1,s2);<br>d:=LE(s1,s2);<br>d:=LT(s1,s2);<br>[With EN/ENO]<br>d:=GT_E(EN,ENO,s1,s2);<br>d:=GE_E(EN,ENO,s1,s2);<br>d:=EQ_E(EN,ENO,s1,s2);<br>d:=LE_E(EN,ENO,s1,s2);<br>d:=LT_E(EN,ENO,s1,s2); |

*1 The input variable "s" can be changed in the range of 2 to 28.

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | ANY_SIMPLE |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(GT(_E) / GE(_E) / EQ(_E) / LE(_E) / LT(_E)) | Output (TRUE: True value, FALSE: False value) | Output variable | BOOL |

#### Processing details

##### ■Operation processing

- These functions perform a comparison operation of input values of (s) and output operation results from (d) in the BOOL type.
  - GT(_E): These functions compare $[(s1)>(s2)]\&[(s2)>(s3)]\&\cdots\&[(s)_{(n-1)}>(s)_{(n)}]$.
    - When all the operation results are $(s)_{(n-1)}>(s)_{(n)}$, these functions output TRUE.
    - When any of the operation results is $(s)_{(n-1)}\leq(s)_{(n)}$, these functions output FALSE.
  - GE(_E): These functions compare $[(s1)\geq(s2)]\&[(s2)\geq(s3)]\&\cdots\&[(s)_{(n-1)}\geq(s)_{(n)}]$.
    - When all the operation results are $(s)_{(n-1)}\geq(s)_{(n)}$, these functions output TRUE.
    - When any of the operation result is $(s)_{(n-1)}<(s)_{(n)}$, these functions output FALSE.
  - EQ(_E): These functions compare $[(s1)=(s2)]\&[(s2)=(s3)]\&\cdots\&[(s)_{(n-1)}=(s)_{(n)}]$.
    - When all the operation results are $(s)_{(n-1)}=(s)_{(n)}$, these functions output TRUE.
    - When any of the operation results is $(s)_{(n-1)}\neq(s)_{(n)}$, these functions output FALSE.
  - LE(_E): These functions compare $[(s1)\leq(s2)]\&[(s2)\leq(s3)]\&\cdots\&[(s)_{(n-1)}\leq(s)_{(n)}]$.
    - When all the operation results are $(s)_{(n-1)}\leq(s)_{(n)}$, these functions output TRUE.
    - When any of the operation result is $(s)_{(n-1)}>(s)_{(n)}$, these functions output FALSE.
  - LT(_E): These functions compare $[(s1)<(s2)]\&[(s2)<(s3)]\&\cdots\&[(s)_{(n-1)}<(s)_{(n)}]$.
    - When all the operation results are $(s)_{(n-1)}<(s)_{(n)}$, these functions output TRUE.
    - When any of the operation results is $(s)_{(n-1)}\geq(s)_{(n)}$, these functions output FALSE.
- A data value of the INT, DINT, REAL, BOOL, WORD, DWORD, TIME, or STRING type can be input to (s).

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

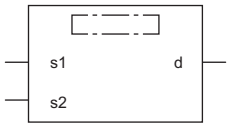
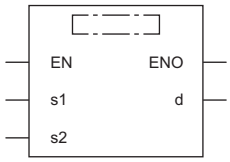
• (s1) to (s28) are the STRING type

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) to (s28) and later, "00H" does not exist. |
| 3405H | The character string specified by (s1) to (s28) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 22.2 Compare

## NE(_E)

These functions output the data comparison result of input values.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=NE(s1,s2);<br>[With EN/ENO]<br>d:=NE_E(EN,ENO,s1,s2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | ANY_SIMPLE |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(NE(_E)) | Output (TRUE: True value, FALSE: False value) | Output variable | BOOL |

### Processing details

#### ■Operation processing

- These functions perform a comparison operation of input values of (s) and output operation results from (d) in the BOOL type.
  - NE(_E): These functions compare [(s1)≠(s2)].
    - When (s1)≠(s2), these functions output TRUE.
    - These functions output FALSE when (s1)=(s2).
- A data value of the INT, DINT, REAL, BOOL, WORD, DWORD, TIME, or STRING type can be input to (s).

#### ■Operation result

***1.*** Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

- (s1) and (s2) are the STRING type

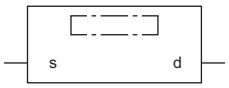
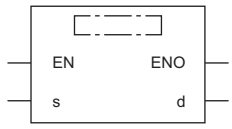
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| 3406H | The whole specified character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 23 CHARACTER STRING FUNCTIONS

## 23.1 Character String Length Detection

### LEN(_E)

These functions detect the length of an input character string and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=LEN(s);<br>[With EN/ENO]<br>d:=LEN _E(EN,ENO,s); |
|  |  | |

#### Setting data

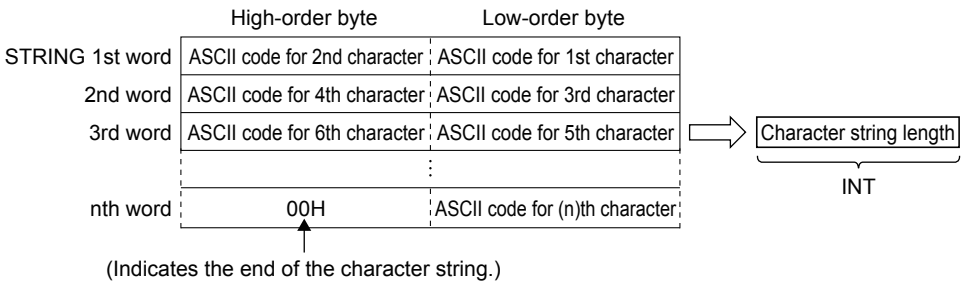
##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(255) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(LEN(_E)) | Output | Output variable | INT |

#### Processing details

##### ■Operation processing

- These functions detect the length of a character string input to (s) and output the result from (d).




- A value input to (s) is the STRING type data value and within the range from 0 to 255 byte(s).

##### ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

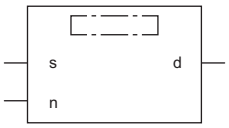
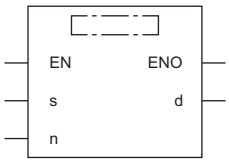
#### Operation error

There is no operation error.

# 23.2 Extracting Character String Data from the Left/Right

## LEFT(_E), RIGHT(_E)

- LEFT(_E): These functions output specified number of characters from the left of input character string data.
- RIGHT(_E): These functions output specified number of characters from the right of input character string data.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=LEFT(s,n);<br>d:=RIGHT(s,n);<br>[With EN/ENO]<br>d:=LEFT_E(EN,ENO,s,n);<br>d:=RIGHT_E(EN,ENO,s,n); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(255) |
| n(L) | Specification of number of characters to be extracted | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(LEFT(_E) / RIGHT(_E)) | Output | Output variable | STRING(255) |

### Processing details

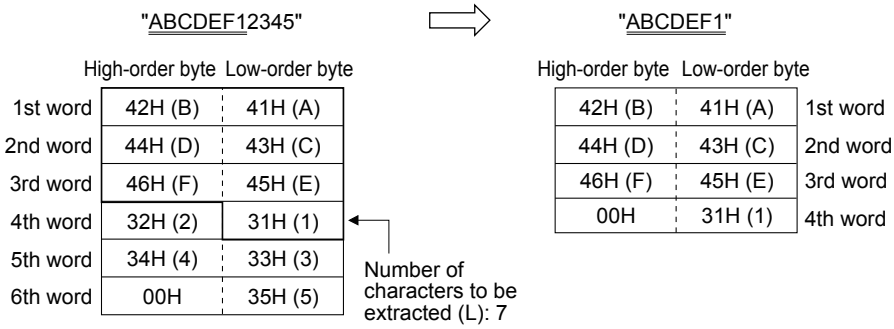
#### ■Operation processing

- LEFT(_E)

  These functions output the data for the specified number of characters from the left of a character string input to (s) from (d).
  The value input to (n) specifies the number of characters to be extracted.

**Ex.**

When the value input to (n) is 7

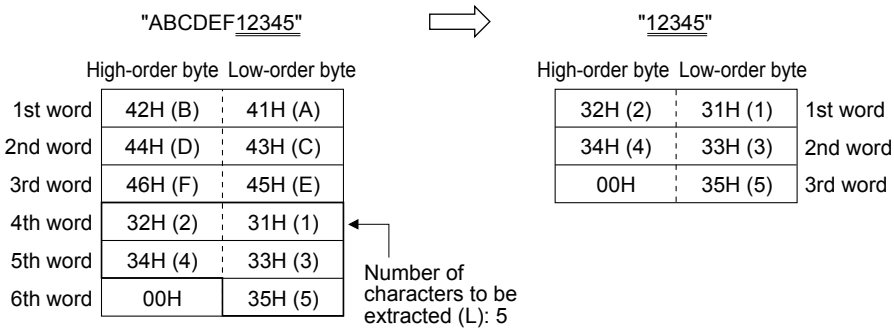


• RIGHT(_E)

These functions output the data for the specified number of characters from the right of a character string input to (s) from (d).
The value input to (n) specifies the number of characters to be extracted.

Ex.

When the value input to (n) is 5




- A value input to (s) is the STRING type data value and within the range from 0 to 255 byte(s).
- A value input to (n) is the INT type data value and within the range from 0 to 255. (However, the value must be within the number of characters of the character string to be input to (s).)

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

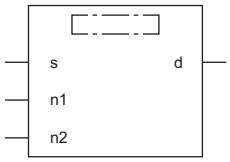
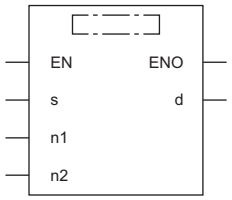
## Operation error

There is no operation error.

# 23.3 Extract Mid String

## MID(_E)

These functions output the specified number of characters from an arbitrary position of an input character string.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=MID(s,n1,n2);<br>[With EN/ENO]<br>d:=MID_E(EN,ENO,s,n1,n2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(255) |
| n1(L) | Specification of number of characters to be extracted | Input variable | INT |
| n2(P) | Specification of head character position of a character string to be extracted | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MID(_E)) | Output | Output variable | STRING(255) |

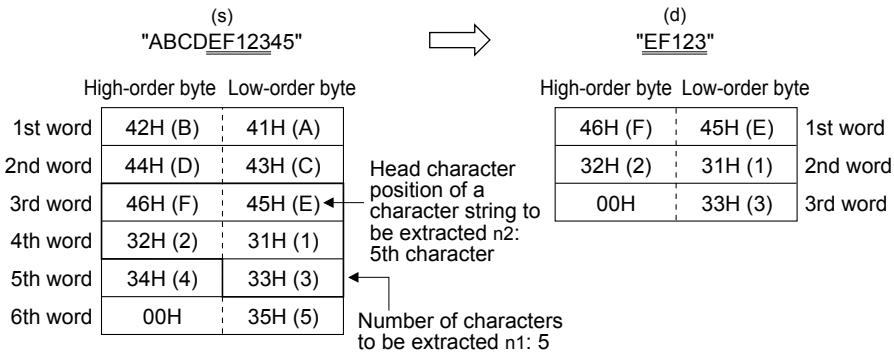
### Processing details

#### ■Operation processing

- These functions output the data for the specified number of characters from an arbitrary position of a character string input to (s).
- The value input to (n1) specifies the number of characters to be extracted.
- The value input to (n2) specifies the number of the head character position of a character string to be extracted.

**Ex.**

When the value input to (n1) and (n2) is 5




- A value input to (s) is the STRING type data value and within the range from 0 to 255 byte(s).
- A value input to (n1) is the INT type data value and within the range from 0 to 255. (However, the value must be within the number of characters of the character string to be input to (s).)
- A value input to (n2) is the INT type data value and within the range from 1 to 255. (However, the value must be within the number of characters of the character string to be input to (s).)

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

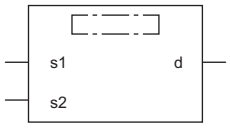
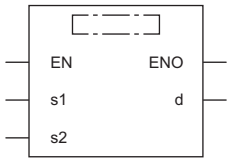
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s) and later, "00H" does not exist. |
| 3405H | The character string specified by (s) has more than 16383 characters. |
| | Data outside the allowable range was set to (n1) and (n2).<br>• The value stored in a device specified in (n1) and (n2) is 0 or less.<br>• The value stored in a device specified in (n2) is any value other than an effective value (-1, 0, 1, or more).<br>• The value stored in a device specified in (n1) exceeds the number of characters of (s).<br>• The total of the values stored in devices specified in (n1) and (n2) exceeds the number of characters of (s). |

# 23.4 Link Character Strings

## CONCAT(_E)

These functions concatenate character strings and output the result.

| Ladder diagram, FBD/LD*1 | | Structured text*1 |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=CONCAT(s1,s2);<br>[With EN/ENO]<br>d:=CONCAT_E(EN,ENO,s1,s2); |
|  |  | |

*1 The input variable "s" can be changed in the range of 2 to 28.

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) to s28(IN28) | Input | Input variable | STRING(255) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(CONCAT(_E)) | Output | Output variable | STRING(255) |

### Processing details

#### ■Operation processing

- These functions concatenate the character string input to the input variables (s2) to (s28) after the one input to (s1) and output the result from (d).
- When character strings are concatenated, 00H indicating an end of the character string specified by (s1) is ignored and the character string specified by (s2) to (s28) is concatenated.
- When the concatenated character string exceeds 255 bytes, these functions output a character string within 255 bytes.

(s1) "ABCDE" + (s2) "123456" ⇨ (d) "ABCDE123456"

| | High-order byte | Low-order byte |
|---|---|---|
| 1st word | 42H (B) | 41H (A) |
| 2nd word | 44H (D) | 43H (C) |
| 3rd word | 00H | 45H (E) |

| | High-order byte | Low-order byte |
|---|---|---|
| 1st word | 32H (2) | 31H (1) |
| 2nd word | 34H (4) | 33H (3) |
| 3rd word | 36H (6) | 35H (5) |
| 4th word | 00H | |

| | High-order byte | Low-order byte |
|---|---|---|
| 1st word | 42H (B) | 41H (A) |
| 2nd word | 44H (D) | 43H (C) |
| 3rd word | 31H (1) | 45H (E) |
| 4th word | 33H (3) | 32H (2) |
| 5th word | 35H (5) | 34H (4) |
| 6th word | 00H | 36H (6) |

- A value input to the input variables (s1) and (s2) to (s28) is the STRING type data value and within the range from 0 to 255 byte(s).

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

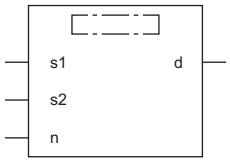
| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) to (s28) and later, "00H" does not exist. |
| 3406H | The whole concatenated character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

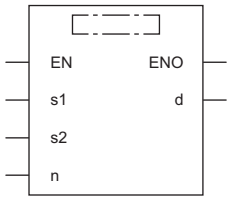
# 23.5 Inserting Character String

## INSERT(_E)

These functions insert a character string into another character string and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=INSERT(s1,s2,n);<br>[With EN/ENO]<br>d:=INSERT_E(EN,ENO,s1,s2,n); |




### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | STRING(255) |
| n(P) | Specification of head character position of a character string to be inserted | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(INSERT(_E)) | Output | Output variable | STRING(255) |

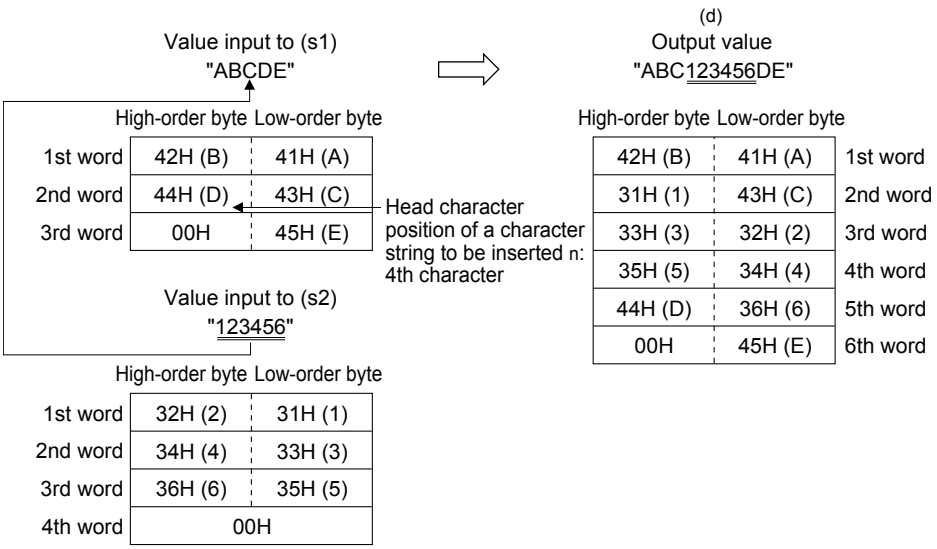
### Processing details

#### ■Operation processing

- These functions insert the character string input to (s2) at the (n)th character from the start of the character string input to (s1) (head position of the insertion) and output from (d).
- After the character string specified by (s2) is inserted to the one specified by (s1), 00H indicating an end of the character string specified by (s2) is ignored.
- When the inserted character string exceeds 255 bytes, these functions output a character string within 255 bytes.

**Ex.**

When the value input to (n) is 4




- A value input to (s1) and (s2) is the STRING type data value and within the range from 0 to 255 byte(s).
- A value input to (n) is the INT type data value and within the range from 1 to 255. (However, the value must be within the number of characters of the character string to be input to (s1).)

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

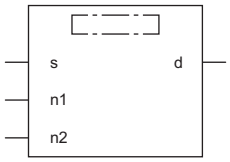
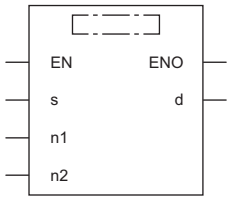
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | In the corresponding device range of the device specified by (s1) to (s28) and later, "00H" does not exist. |
| 3406H | The whole concatenated character string cannot be stored in the devices from the device specified by (d) to the last device in the corresponding device range. |

# 23.6 Deleting Character String

## DELETE(_E)

These functions delete an arbitrary range of a character string and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=DELETE(s,n1,n2);<br>[With EN/ENO]<br>d:=DELETE_E(EN,ENO,s,n1,n2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Input | Input variable | STRING(255) |
| n1(L) | Specification of number of characters to be deleted | Input variable | INT |
| n2(P) | Specification of head character position of a character string to be deleted | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DELETE(_E)) | Output | Output variable | STRING(255) |

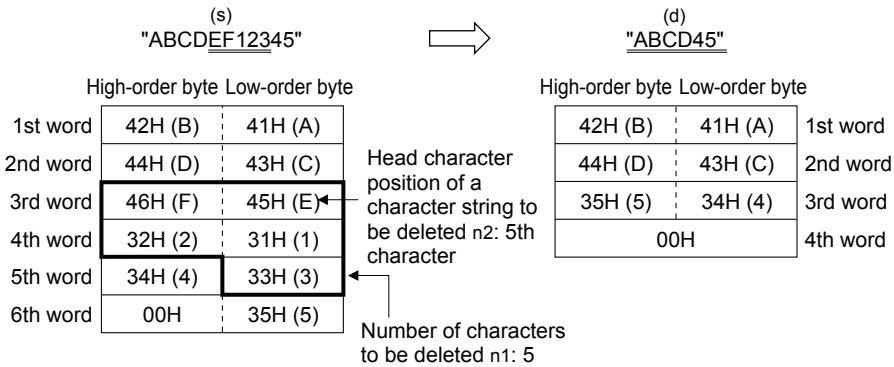
### Processing details

#### ■Operation processing

- These functions delete the data for the specified number of characters from an arbitrary position of a character string input to (s) and output the remaining character strings from (d).
- The value input to (n1) specifies the number of characters to be deleted.
- The value input to (n2) specifies the number of the head character position of a character string to be deleted.

**Ex.**

When the value input to (n1) and (n2) is 5




- A value input to (s) is the STRING type data value and within the range from 0 to 255 byte(s).
- A value input to (n1) is the INT type data value and within the range from 0 to 255. (However, the value must be within the number of characters of the character string to be input to (s).)
- A value input to (n2) is the INT type data value and within the range from 1 to 255. (However, the value must be within the number of characters of the character string to be input to (s).)

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

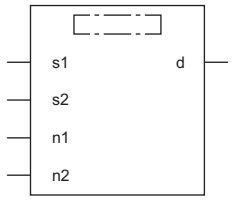
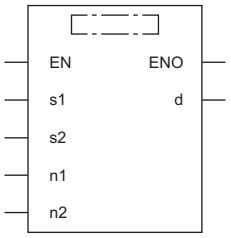
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | "00H" is not set to devices from the device number specified by (s) to the end device number of corresponding device. |
| 3405H | The character strings specified by (s) have more than 255 characters. |
| | The device value specified by (n1) is out of the valid range (0 to 255). |
| | The device value specified by (n2) is out of the valid range (1 to 255). |
| | (n1) exceeds the number of characters of a character string specified by (s). |
| | (n2) exceeds the number of characters of a character string specified by (s). |
| 3406H | The whole deleted character string cannot be stored in the devices from the device specified by (d) to the last device of the target device. |

# 23.7 Replacing Character String

## REPLACE(_E)

These functions replace an arbitrary range of a character string and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>d:=REPLACE(s1,s2,n1,n2);<br>[With EN/ENO]<br>d:=REPLACE_E(EN,ENO,s1,s2,n1,n2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | STRING(255) |
| n1(L) | Specification of number of characters to be replaced | Input variable | INT |
| n2(P) | Specification of head character position of a character string to be replaced | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(REPLACE(_E)) | Output | Output variable | STRING(255) |

### Processing details

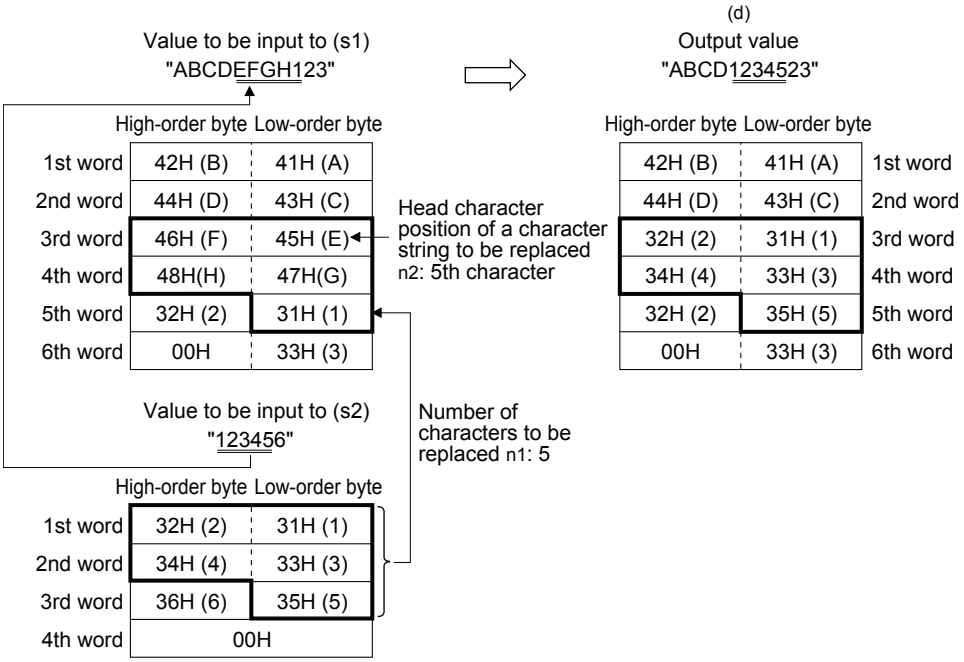
#### ■Operation processing

- These functions replace the data for the specified number of characters from an arbitrary position of a character string input to (s1) with a character string input to (s2) and output from (d).
- The value input to (n1) specifies the number of characters to be replaced.

• The value input to (n2) specifies the number of the head character position of a character string to be replaced.

Ex.

When the value input to (n1) and (n2) is 5




• A value input to (s1) and (s2) is the STRING type data value and within the range from 0 to 255 byte(s).

• A value input to (n1) is the INT type data value and within the range from 0 to 255. (However, the value must be within the number of characters of the character string to be input to (s1).)

• A value input to (n2) is the INT type data value and within the range from 1 to 255. (However, the value must be within the number of characters of the character string to be input to (s1).)

## ■Operation result

### *1.* Function without EN/ENO

The following table lists the operation results.

| Operation result | (d) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

### *2.* Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| EN | ENO | (d) |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred)*1 | Indefinite value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

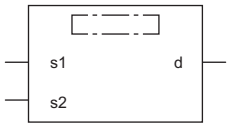
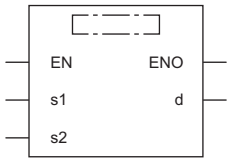
## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 2820H | "00H" is not set to devices from the device number specified by (s1) to the end device number of corresponding device. |
| | "00H" is not set to devices from the device number specified by (s2) to the end device number of corresponding device. |
| 3405H | The character strings specified by (s1) have more than 255 characters. |
| | The character strings specified by (s2) have more than 255 characters. |
| | The device value specified by (n1) is out of the valid range (0 to 255). |
| | The device value specified by (n2) is out of the valid range (1 to 255). |
| | (n1) exceeds the number of characters of a character string specified by (s2). |
| | (n2) exceeds the number of characters of a character string specified by (s1). |
| 3406H | The whole deleted character string cannot be stored in the devices from the device specified by (d) to the last device of the target device. |

# 23.8 Searching Character String

## FIND(_E)

These functions search for a character string and output the result.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=FIND(s1,s2);<br>[With EN/ENO]<br>d:=FIND_E(EN,ENO,s1,s2); |

### Setting data

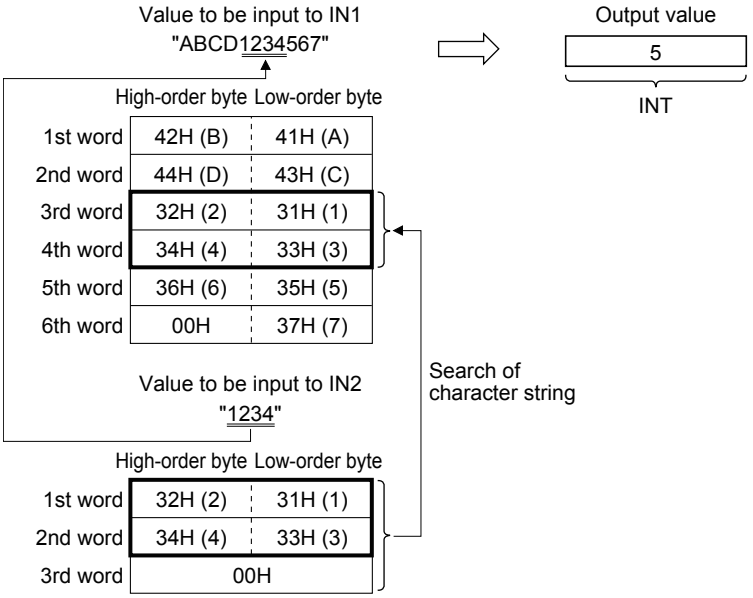
#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | STRING(255) |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(FIND(_E)) | Output | Output variable | INT |

### Processing details

#### ■Operation processing

- These functions search for a character string input to (s2) from the start of the character string input to (s1) and output the result from (d).
- This function outputs the head character position of the searched character string detected first as the search result.
- If a character string specified by (s2) cannot be searched from the one specified by (s1), these functions output "0".




- A value input to (s1) and (s2) is the STRING type data value and within the range from 0 to 255 byte(s).

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

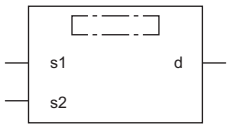
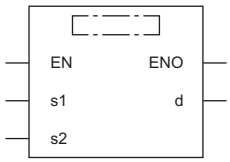
There is no operation error.

# 24 TIME DATA FUNCTIONS

## 24.1 Addition

### ADD_TIME(_E)

These functions output the sum of input values (TIME data) ((s1) + (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=ADD_TIME(s1,s2);<br>[With EN/ENO]<br>d:=ADD_TIME_E(EN,ENO,s1,s2); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(ADD_TIME(_E)) | Output | Output variable | TIME |

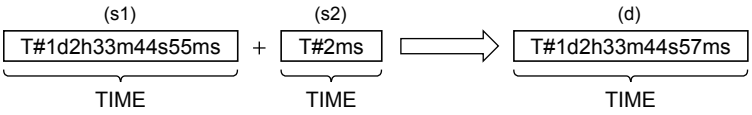
#### Processing details

##### ■Operation processing

- These functions add the TIME type data input to (s1) and (s2) ((s1) + (s2)), and output the operation result from (d) as TIME type data.

Ex.

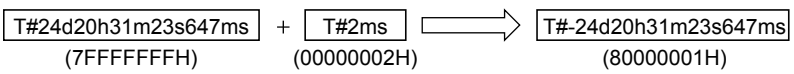
When a value input to (s1) and (s2) is T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and T#2ms (2 milliseconds)




- A value input to (s1) and (s2) is the TIME type data value.
- Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. The data is output from (d) as follows: "ADD_TIME_E" outputs "TRUE" from the output variable ENO.

Ex.

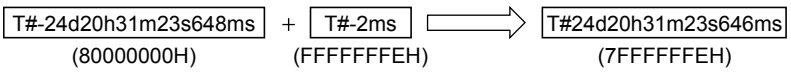
Overflow




The most significant bit becomes 1, and a negative time is output.

Ex.

Underflow




The most significant bit becomes 0, and a positive time is output.

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

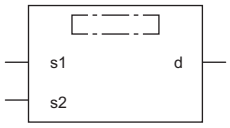
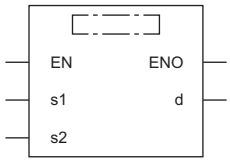
## Operation error

There is no operation error.

# 24.2 Subtraction

## SUB_TIME(_E)

These functions output the difference of input values (TIME data) ((s1) - (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=SUB_TIME(s1,s2);<br>[With EN/ENO]<br>d:=SUB_TIME_E(EN,ENO,s1,s2); |
|  |  | |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1), s2(IN2) | Input | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(SUB_TIME(_E)) | Output | Output variable | TIME |

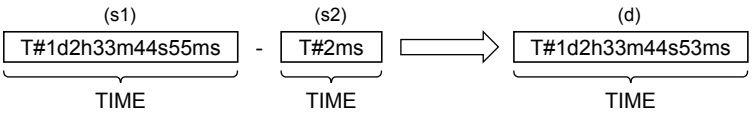
### Processing details

#### ■Operation processing

- These functions subtract the TIME type data input to (s1) and (s2) ((s1) - (s2)), and output the operation result from (d) as TIME type data.

Ex.

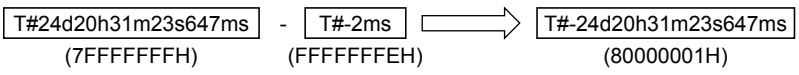
When a value input to (s1) and (s2) is T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and T#2ms (2 milliseconds)




- A value input to (s1) and (s2) is the TIME type data value.
- Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. The data is output from (d) as follows: "SUB_TIME_E" outputs "TRUE" from the output variable ENO.

Ex.

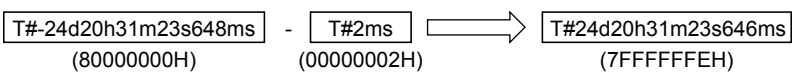
Overflow




The most significant bit becomes 1, and a negative time is output.

Ex.

Underflow




The most significant bit becomes 0, and a positive time is output.

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

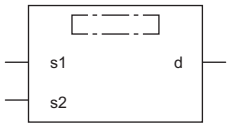
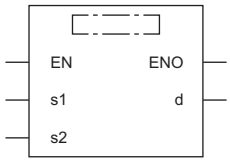
## Operation error

There is no operation error.

# 24.3 Multiplication

## MUL_TIME(_E)

These functions output the multiplication of input values (TIME) ((s1) × (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>d:=MUL_TIME(s1,s2);<br>[With EN/ENO]<br>d:=MUL_TIME_E(EN,ENO,s1,s2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) | Input | Input variable | TIME |
| s2(IN2) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(MUL_TIME(_E)) | Output | Output variable | TIME |

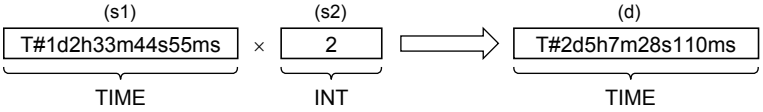
### Processing details

#### ■Operation processing

- These functions multiply the TIME type data input to (s1) and (s2) ((s1) × (s2)), and output the operation result from (d) as TIME type data.

Ex.

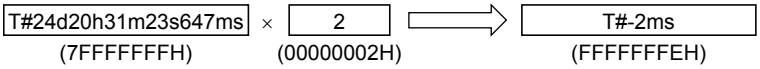
When a value input to (s1) and (s2) is T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2




- A value input to (s1) is the TIME type data value.
- A value input to (s2) is the INT, DINT, or REAL type.
- Even if underflow or overflow occurs in the operation result, it is not regarded as an operation error. The data is output from (d) as follows: "MUL_TIME_E" outputs "TRUE" from the output variable ENO. (The operation result is the 64-bit data, however, the output data is the time type data with high-order 32 bits deleted.)

Ex.

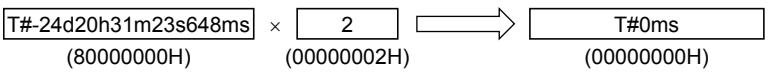
Overflow




The most significant bit becomes 1, and a negative time is output.

Ex.

Underflow




The most significant bit becomes 0, and a positive time is output.

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

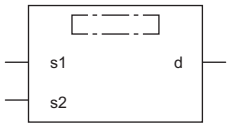
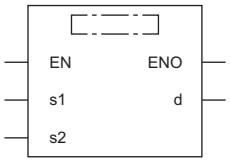
## Operation error

There is no operation error.

# 24.4 Division

## DIV_TIME(_E)

These functions output the quotient of input values (TIME data) ((s1) ÷ (s2)).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>d:=DIV_TIME(s1,s2);<br>[With EN/ENO]<br>d:=DIV_TIME_E(EN,ENO,s1,s2); |
|  |  | |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(IN1) | Input | Input variable | TIME |
| s2(IN2) | Input | Input variable | ANY_NUM |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(DIV_TIME(_E)) | Output | Output variable | TIME |

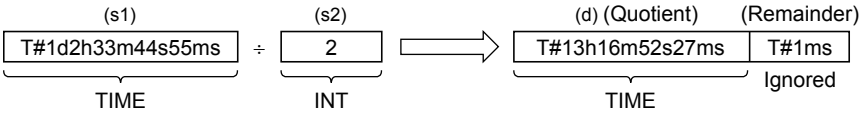
### Processing details

#### ■Operation processing

- These functions divide the TIME type data input to (s1) and (s2) ((s1) ÷ (s2)), and output the operation result from (d) as TIME type data. The remainder is ignored.

**Ex.**

When a value input to (s1) and (s2) is T#1d2h33m44s55ms (1 day 2 hours 33 minutes 44 seconds 55 milliseconds) and 2




- A value input to (s1) is the TIME type data value.
- A value input to (s2) is the INT, DINT, or REAL type. (However, input other than 0 to (s2).)

## ■Operation result

***1.*** Function without EN/ENO

The operation processing is executed. The operation output value is output from (d).

***2.*** Function with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE*1 | Indefinite value |

*1 When FALSE is output from ENO, data output from (d) is undefined. In that case, modify a program so that the data output from (d) is not used.

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3400H | A value input to (s2) is 0. (Zero division) |

# PART 6 FUNCTION BLOCKS

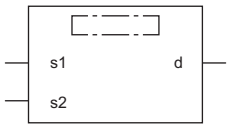
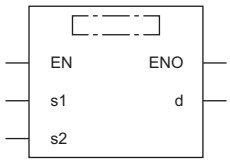
This part consists of the following chapters.

# 25 BISTABLE FUNCTION BLOCKS

## 25.1 Bistable Function Blocks (Set Priority)

### SR(_E)

These function blocks judge two input values and output 1 (TRUE) or 0 (FALSE).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>SR_1(S1:=s1,R:=s2,Q1:=d);<br>[With EN/ENO]<br>SR_E_1(EN:=EN,ENO:=ENO S1:=s1,R:=s2,Q1:=d); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(S1) | Set instruction | Input variable | BOOL |
| s2(R) | Reset instruction | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(Q1) | Output | Output variable | BOOL |

#### Processing details

##### ■Operation processing

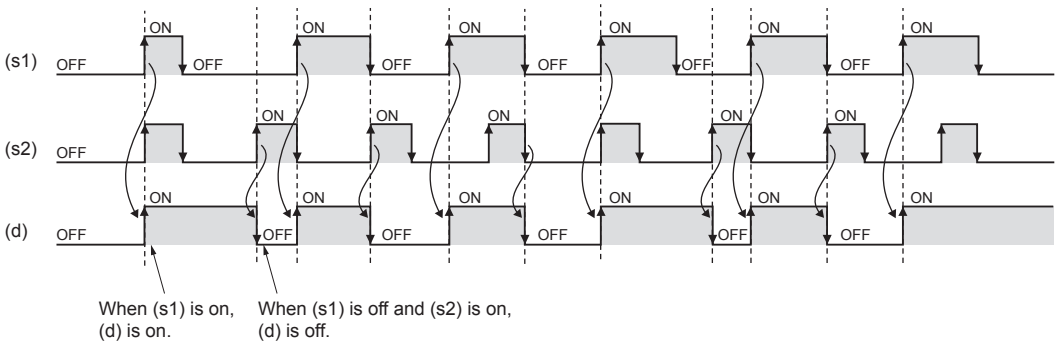
- When (s1) turns ON, (d) is set. If (s2) is turned ON when (s1) is OFF, (d) is reset.
- If (s2) is turned ON when (s1) is ON, (d) is not reset.

##### ■Operation result

***1.*** Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d).

- Timing chart

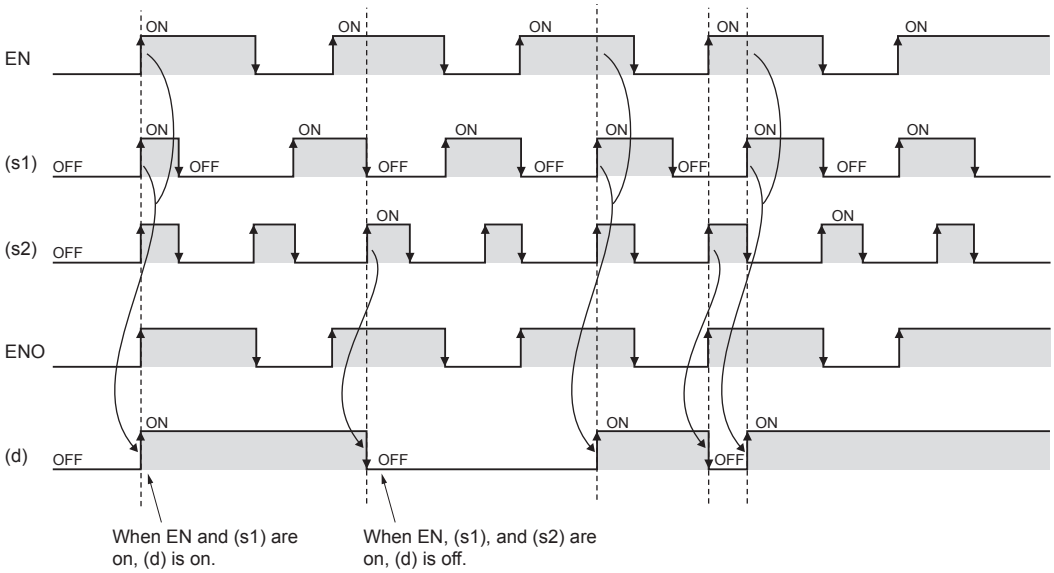


### 2. Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart



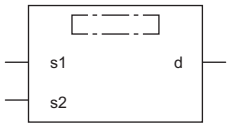
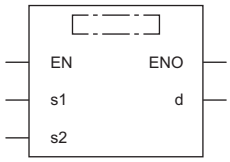

## Operation error

There is no operation error.

# 25.2 Bistable Function Blocks (Reset Priority)

## RS(_E)

These function blocks judge two input values and output 1 (TRUE) or 0 (FALSE).

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>RS_1(S:=s1,R1:=s2,Q1:=d);<br>[With EN/ENO]<br>RS_E_1(EN:=EN, ENO:=ENO S:=s1,R1:=s2,Q1:=d); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(S) | Set instruction | Input variable | BOOL |
| s2(R1) | Reset instruction | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(Q1) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

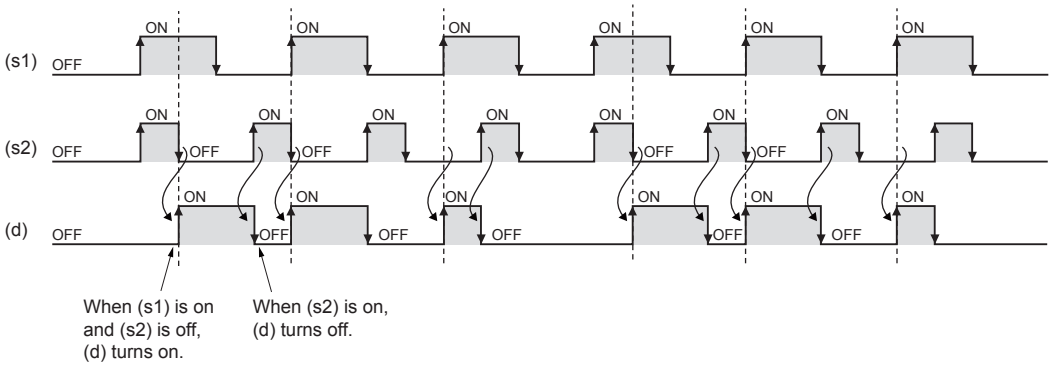
- When (s1) turns ON, (d) is set. When (s2) is turned ON, (d) is reset.
- If (s1) is turned ON when (s2) is ON, (d) is not set.

#### ■Operation result

***1.*** Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d).

- Timing chart

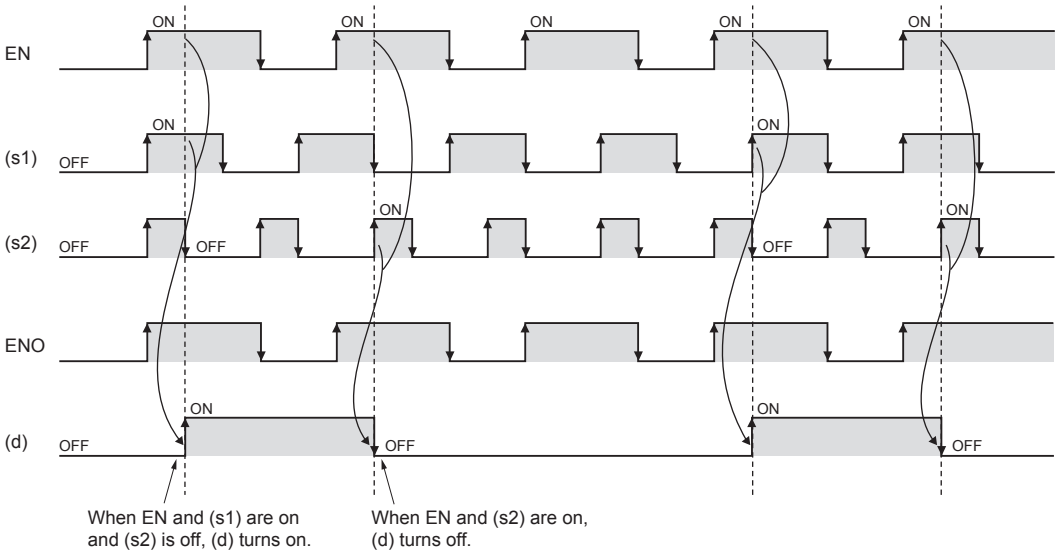


**2.** Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart




## Operation error

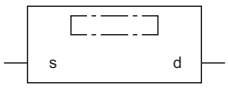
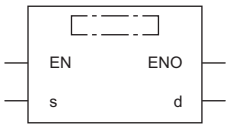
There is no operation error.

# 26 EDGE DETECTION FUNCTION BLOCKS

## 26.1 Rising Edge Detector

### R_TRIG(_E)

These functions detect the rising edge of a signal, and output a pulse signal.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>R_TRIG_1(CLK:=s,Q:=d);<br>[With EN/ENO]<br>R_TRIG_E_1(EN:=EN, ENO:=ENO CLK:=s,Q:=d); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(CLK) | Rising edge detector input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(Q) | Output | Output variable | BOOL |

#### Processing details

##### ■Operation processing

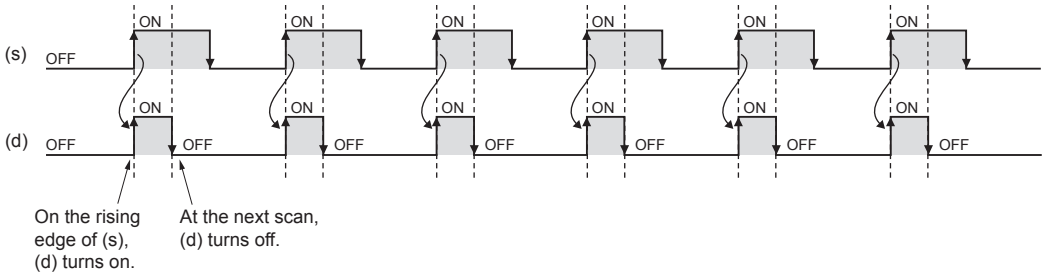
When (s) turns ON, (d) is turned ON only for one scan.

##### ■Operation result

***1.*** Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d).

- Timing chart



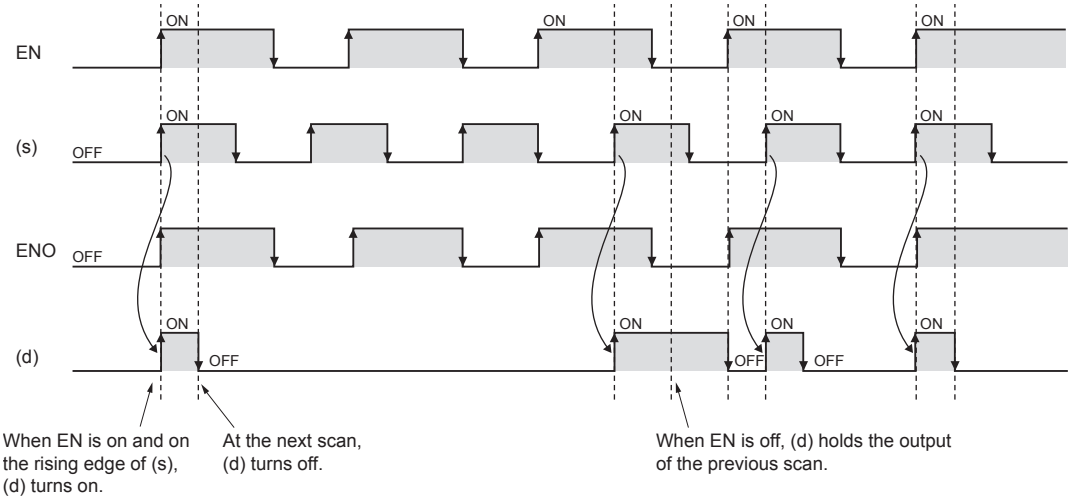
**2.** Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |



- Timing chart



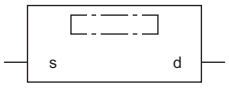
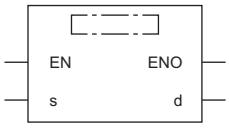

## Operation error

There is no operation error.

# 26.2 Falling Edge Detector

## F_TRIG(_E)

These function blocks detect the falling edge of a signal, and output a pulse signal.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  | [With EN/ENO]<br>  | [Without EN/ENO]<br>F_TRIG_1(CLK:=s,Q:=d);<br>[With EN/ENO]<br>F_TRIG_E_1(EN:=EN, ENO:=ENO CLK:=s,Q:=d); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(CLK) | Falling edge detector input | Input variable | BOOL |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d(Q) | Output | Output variable | BOOL |

### Processing details

#### ■Operation processing

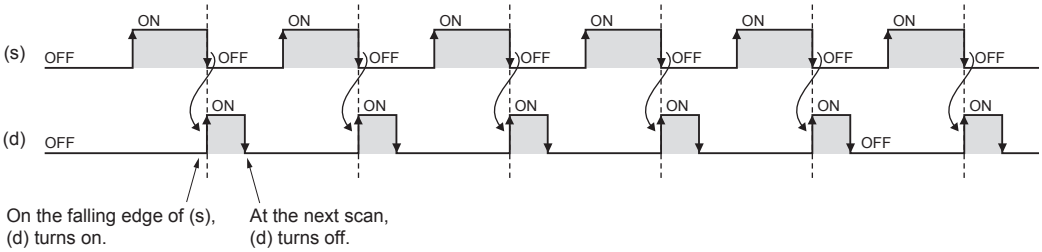
When (s) turns OFF, (d) is turned ON only for one scan.

#### ■Operation result

***1.*** Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d).

- Timing chart



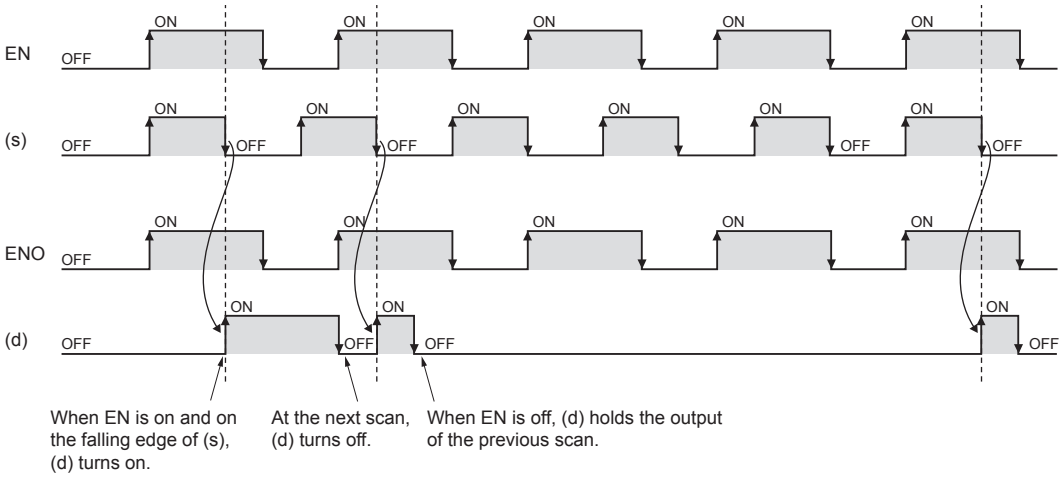
**2.** Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |



- Timing chart




## Operation error

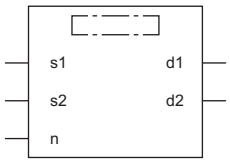
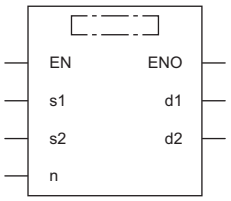
There is no operation error.

# 27 COUNTER FUNCTION BLOCKS

## 27.1 Up Counter

### CTU(_E)

These function blocks count up the number of times of rising of a signal.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>CTU_1(CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2);<br>[With EN/ENO]<br>CTU_E_1(EN:=EN, ENO:=ENO CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(CU) | Count signal input | Input variable | BOOL |
| s2(R) | Count value reset | Input variable | BOOL |
| n(PV) | Count maximum value | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(Q) | Count end | Output variable | BOOL |
| d2(CV) | Count value | Output variable | INT |

#### Processing details

##### ■Operation processing

***1.*** Count up

- These function blocks count up (add "1" to) the value of (d2) when (s1) turns ON from OFF.
- When the value of (d2) reaches the value of (n) of the counter, (d1) turns ON and the function blocks stop counting up.
- Set the maximum value of the counter for (n). When (s2) is turned ON, (d1) turns OFF and (d2) is set to 0.

***2.*** Count maximum value

The effective setting range of (n) is from 0 to 32767.

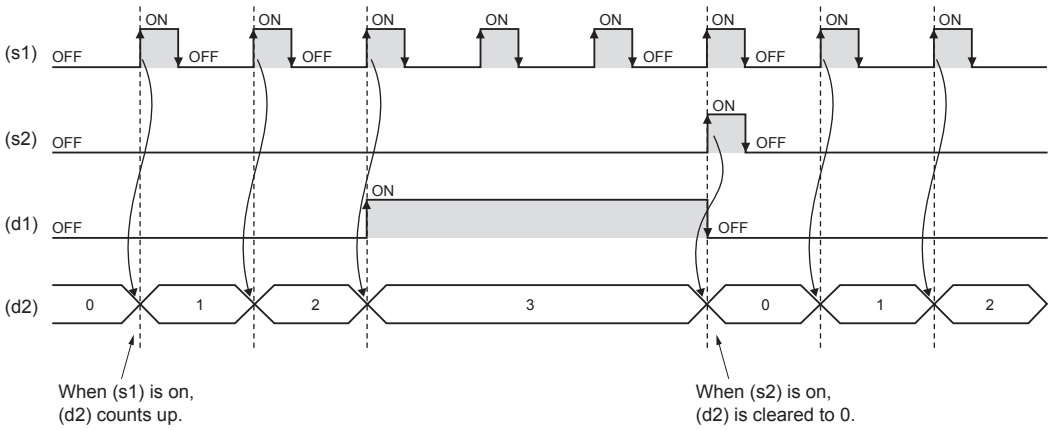
## ■Operation result

### *1.* Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d1) and (d2).

- Timing chart

When 3 is specified in n



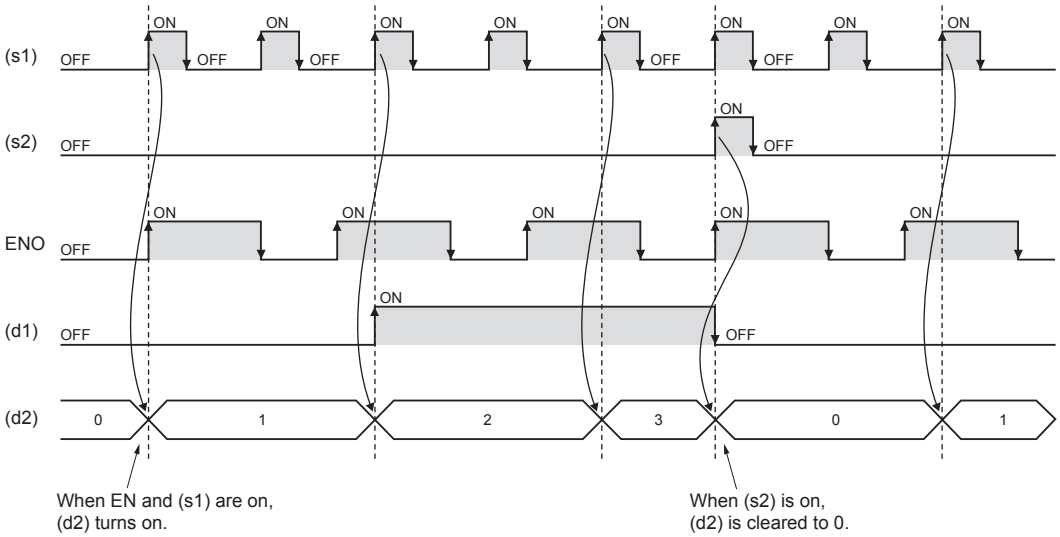

### *2.* Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d1), (d2)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When 3 is specified in n



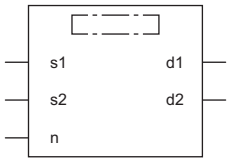
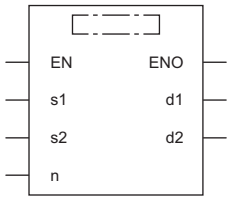

## Operation error

There is no operation error.

# 27.2 Down Counter

## CTD(_E)

These function blocks count down the number of times of rising of a signal.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>CTD_1(CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2);<br>[With EN/ENO]<br>CTD_E_1(EN:=EN, ENO:=ENO CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(CD) | Count signal input | Input variable | BOOL |
| s2(LD) | Count value set | Input variable | BOOL |
| n(PV) | Count start value | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(Q) | Count end | Output variable | BOOL |
| d2(CV) | Count value | Output variable | INT |

### Processing details

#### ■Operation processing

***1.*** Count down

- These function blocks count down (subtract "-1" from) the value of (d2) when (s1) turns ON from OFF.
- When the value of (d2) is 0, (d1) turns ON and the function blocks stop counting down.
- Set the count start value for (n). When (s2) is turned ON, (d1) turns OFF and (n) is set for (d2).

***2.*** Count start value

The effective setting range of (n) is from 0 to 32767.

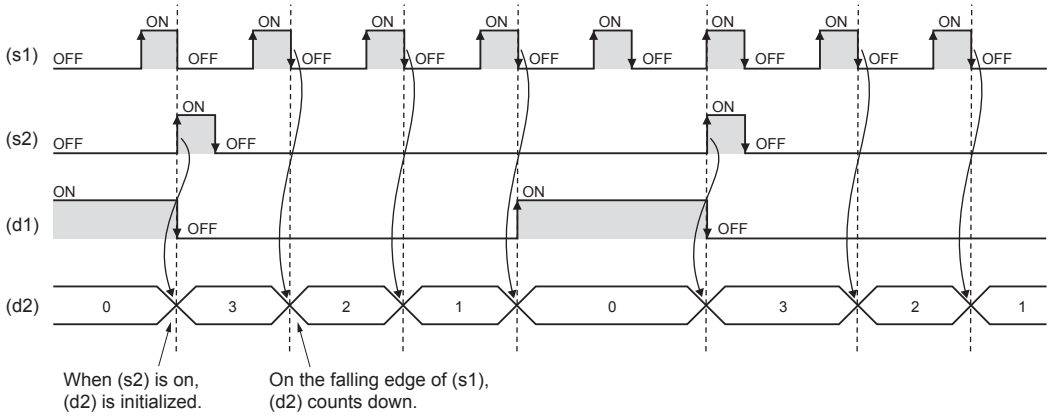
## ■Operation result

### *1.* Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d1) and (d2).

- Timing chart

When 3 is specified in n



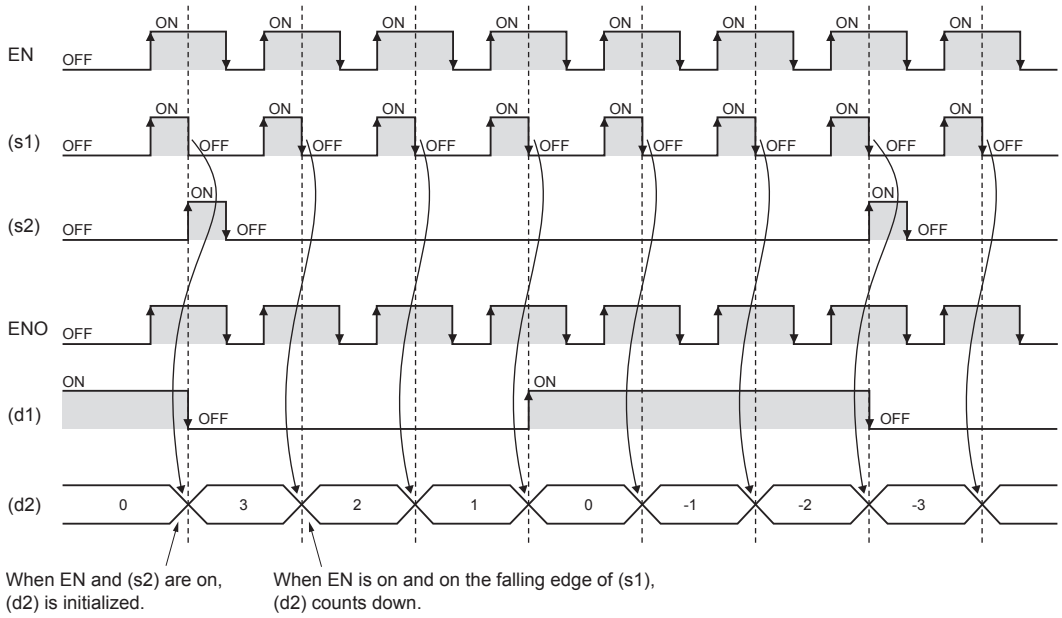

### *2.* Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d1), (d2)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When 3 is specified in n



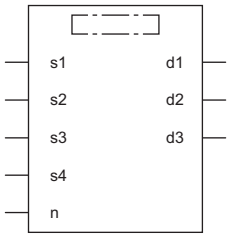
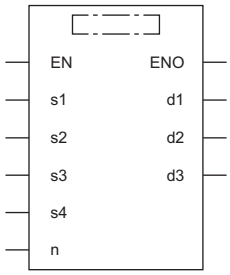

## Operation error

There is no operation error.

# 27.3 Up-down Counter

## CTUD(_E)

These functions count up/down the number of times of rising of a signal.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO] | [With EN/ENO] | [Without EN/ENO]<br>CTUD_1(CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3);<br>[With EN/ENO]<br>CTUD_E_1(EN:=EN,ENO:=ENO,CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3); |
|  |  | |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s1(CU) | Count up signal input | Input variable | BOOL |
| s2(CD) | Count down signal input | Input variable | BOOL |
| s3(R) | Count value reset | Input variable | BOOL |
| s4(LD) | Count value set | Input variable | BOOL |
| n(PV) | Count maximum value/start value | Input variable | INT |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(QU) | Count up end | Output variable | BOOL |
| d2(QD) | Count down end | Output variable | BOOL |
| d3(CV) | Current count value | Output variable | INT |

### Processing details

#### ■Operation processing

***1.*** Count up

- These function blocks count up (add "1" to) the value of (d3) when (s1) turns ON from OFF.
- When the value of (d3) reaches the value of (n), (d1) turns ON and the function blocks stop counting up.
- Set the maximum value of the counter for (n). When (s3) is turned ON, (d1) turns OFF and (d3) is set to 0.

***2.*** Count down

- These function blocks count down (subtract "-1" from) the value of (d3) when (s2) turns ON from OFF.
- When the value of (d3) is 0, (d2) turns ON and the function blocks stop counting down.
- Set the counter start value for (n). When (s4) is turned ON, (d2) turns OFF and (n) is set for (d3).

***3.*** Count maximum value/start value

The effective setting range of (n) is from 0 to 32767.

***4.*** Others

- When (s1) and (s2) are simultaneously turned ON from OFF, (s1) is prioritized and the function blocks count up (add "1" to) the value of (d3).
- When (s3) and (s4) are simultaneously turned ON, (s3) is prioritized and the value of (d3) is set to 0.

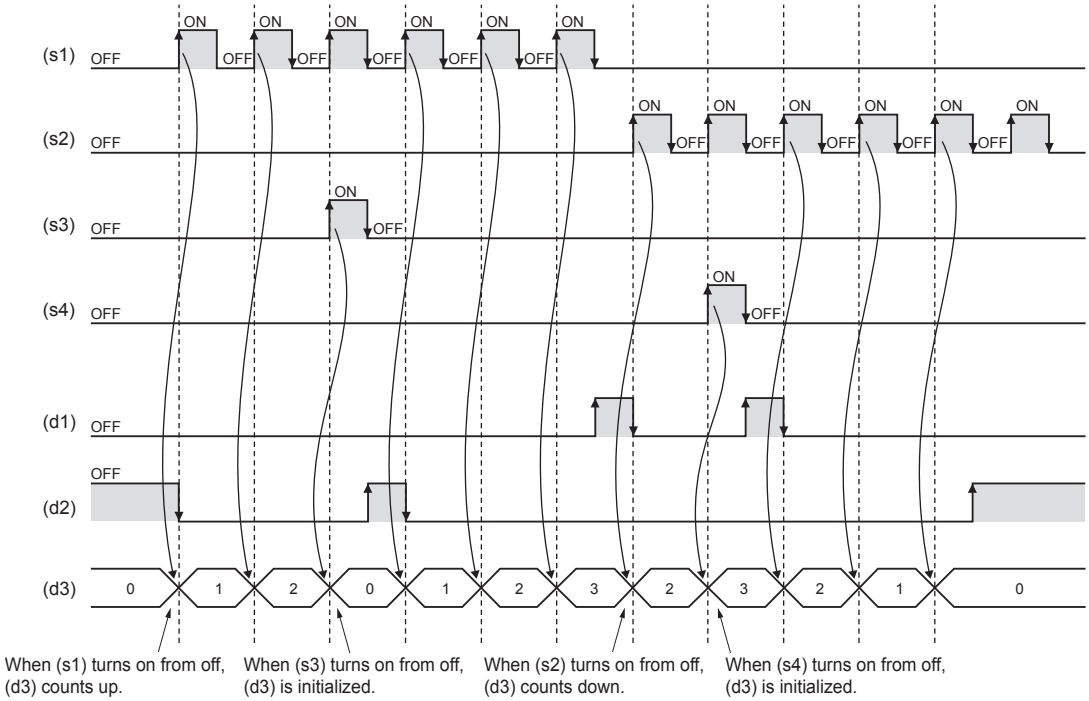
## ■Operation result

### *1.* Function block without EN/ENO

The operation processing is executed. The operation output value is output from (d1), (d2), and (d3).

- Timing chart

When 3 is specified in n



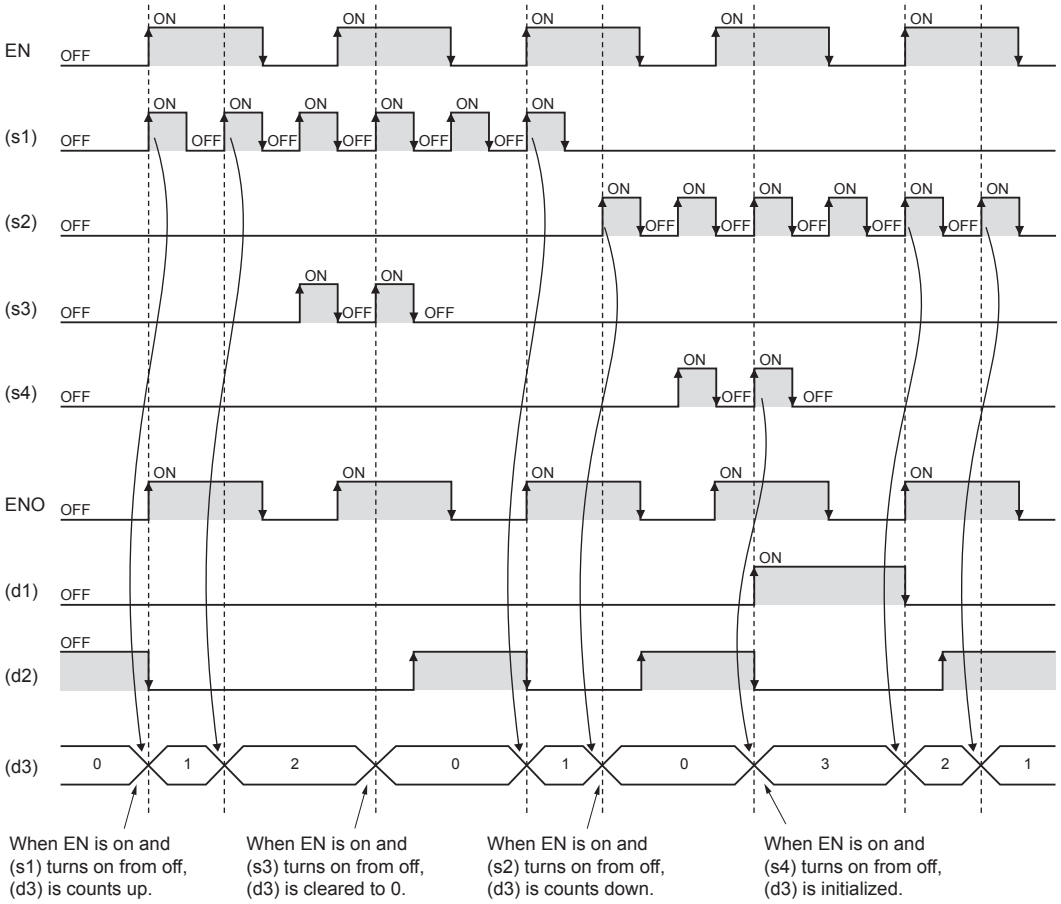
### 2. Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d1), (d2), (d3)** |
| TRUE (Executes operation) | TRUE | Operation output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When 3 is specified in n



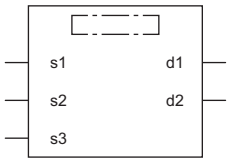

## Operation error

There is no operation error.

# 27.4 Counter Function Block

## COUNTER_FB_M

When the execution condition is established, this function block starts counting up.

| Ladder diagram, FBD/LD | Structured text |
|---|---|
|  | COUNTER_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2); |



### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| s1(Coil) | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s2(Preset) | Counter set value | Input variable | INT |
| s3(ValueIn) | Counter initial value | Input variable | INT |
| d1(ValueOut) | Counter current value | Output variable | ANY16 |
| d2(Status) | Output | Output variable | BOOL |

### Processing details

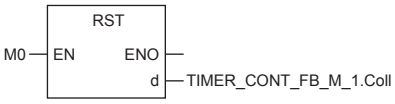
#### ■Operation processing

- The counter starts counting when detecting the rising edge (from OFF to ON) of (s1). It does not start counting if (s1) remains ON. The counting is started from the value of (s3). When the count value reaches the value of (s2), (d2) turns ON. The current count value is stored in (d1).
- A value in the range of 0 to 32767 can be specified for (s2).
- A value in the range of -32768 to 32767 can be specified for (s3). However, when a negative value is specified, the initial value is set to 0.
- To reset the current value of the counter (d1), reset (s1) of FB directly.

Ex.

When the label name is TIMER_CONT_FB_M_1

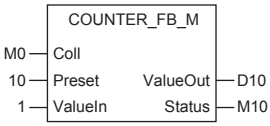
[Ladder]



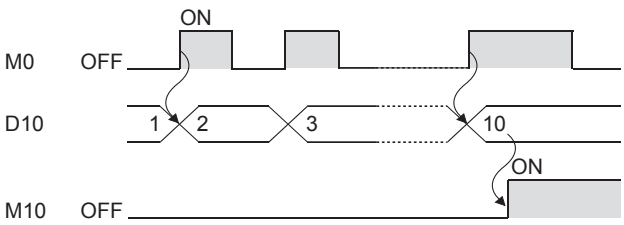

[ST]

RST(M0,TIMER_CONT_FB_M_1.Coil)

[Ladder example]




[Timing chart]



## Operation error

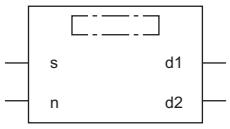
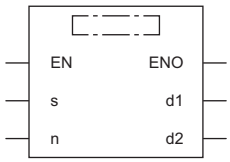
There is no error.

# 28 TIMER FUNCTION BLOCKS

## 28.1 Pulse Timer

### TP(_E), TP_10(_E)

These function blocks keep on a signal for specified duration.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>TP_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TP_10_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>[With EN/ENO]<br>TP_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TP_10_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2); |

#### Setting data

##### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Output start | Input variable | BOOL |
| n(PT) | Output time setting value | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(Q) | Output | Output variable | BOOL |
| d2(ET) | Elapsed time | Output variable | TIME |

#### Processing details

##### ■Operation processing

***1.*** Output

- When (s) turns on, (d1) turns on for the time specified by (n). The elapsed time from when (d1) is turned on is set for (d2).
- The timer device is used for counting the elapsed time.

***2.*** End of output

- (d1) turns off when the elapsed time reaches the set value.
- If (s) is off after (d1) is turned off, the elapsed time is reset.
- Even if (s) is turned off when (d1) is on, (d1) is not turned off.

***3.*** Setting of output time

For TP(_E), the output time value of (n) is set in units of 100 ms or more. For TP_10(_E), it is set in units of 10 ms or more. The value of when (d1) is turned on from off (rising edge) is used for the setting value of (n). If the value of (n) is changed while (d1) is on, the changed value becomes valid at the next output start.

## ■Operation result

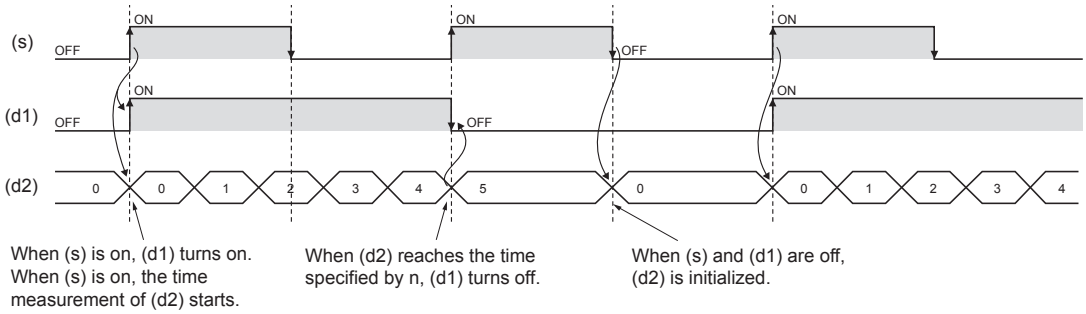
### *1.* Function block without EN/ENO

The following table lists the operation results.

| Operation result | (d1), (d2) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

- Timing chart

When T#5s (5 seconds) is specified in n



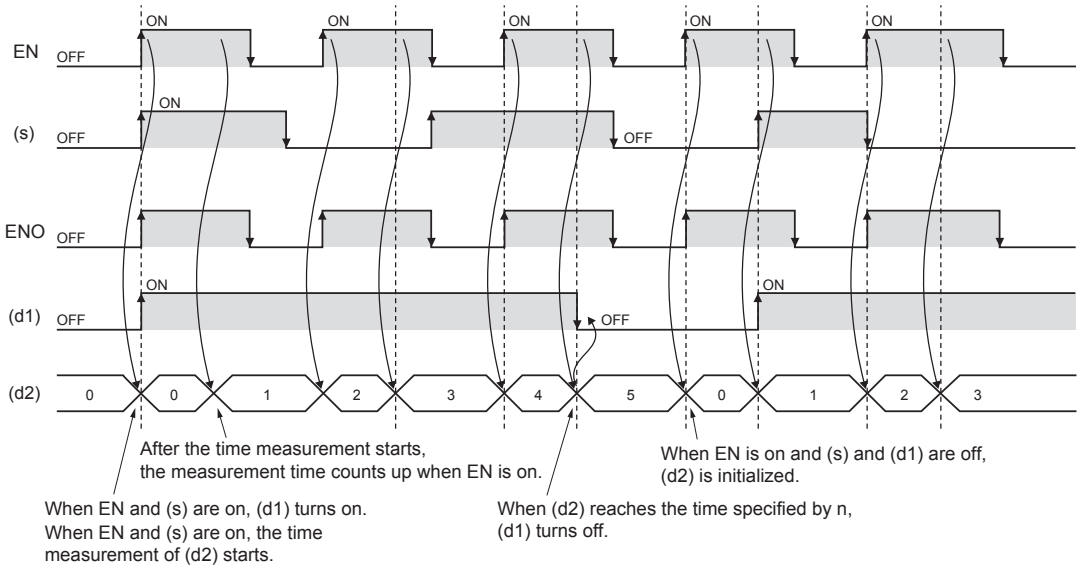

### *2.* Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d1), (d2)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred) | Indefinite value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When T#5s (5 seconds) is specified in n



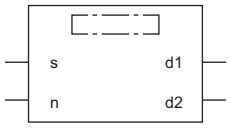
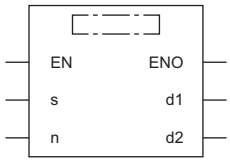

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The set value of the output time exceeds the effective range. |

# 28.2 On-delay Timer

## TON(_E), TON_10(_E)

These function blocks turn on a signal after a specified time.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>TON_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TON_10_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>[With EN/ENO]<br>TON_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TON_10_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Time measurement | Input variable | BOOL |
| n(PT) | Delay time setting value | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(Q) | Output | Output variable | BOOL |
| d2(ET) | Elapsed time | Output variable | TIME |

### Processing details

#### ■Operation processing

***1.*** Output

- When (s) turns on, (d1) is turned on after the time specified by (n). The delay elapsed time from when (d1) is turned on is set for (d2).
- When (s) is turned off, (d1) is turned off and the delay elapsed time is reset.
- The timer device is used for counting the elapsed time.

***2.*** Setting of delay time

For TON(_E), the output time value of (n) is set in units of 100 ms or more. For TON_10(_E), it is set in units of 10 ms or more. The value of when (s) is turned on from off (rising edge) is used for the setting value of (n). If the value of (n) is changed while (s) is on, the changed value becomes valid on the next rising edge of (s).

## ■Operation result

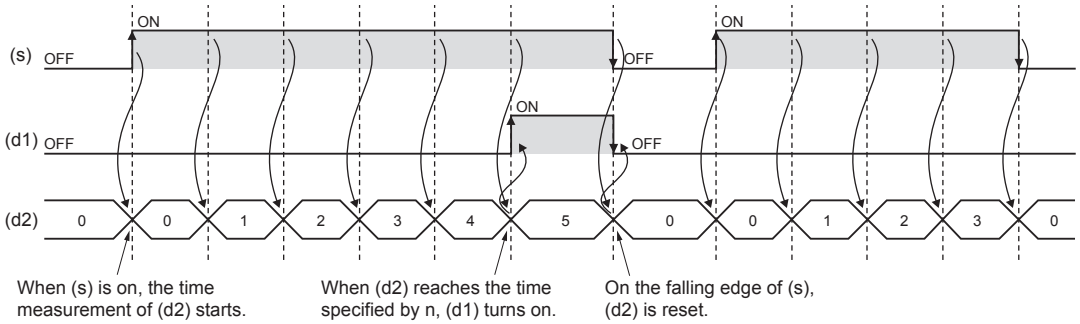
### *1.* Function block without EN/ENO

The following table lists the operation results.

| Operation result | (d1), (d2) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

- Timing chart

When T#5s (5 seconds) is specified in n



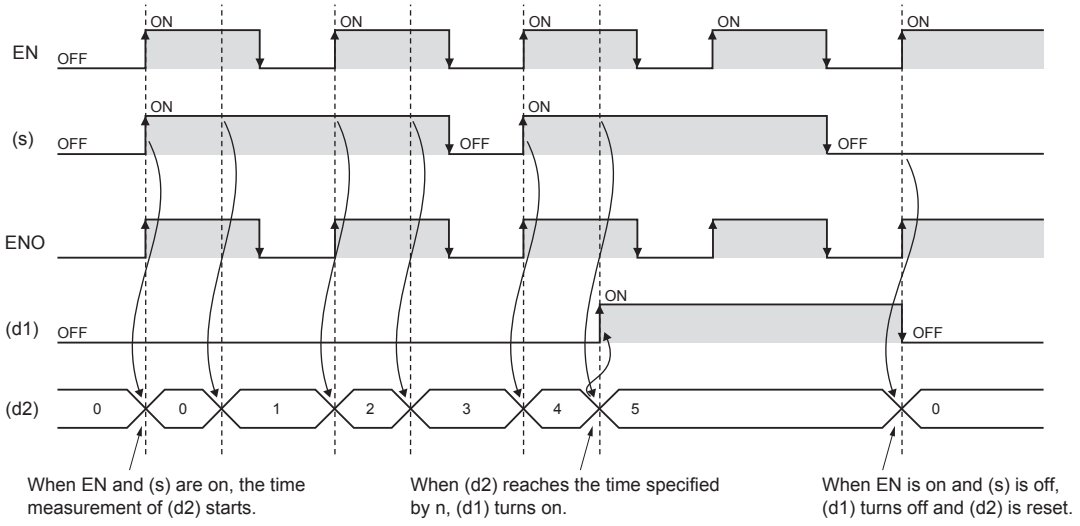

### *2.* Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d1), (d2)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred) | Previous output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When T#5s (5 seconds) is specified in n



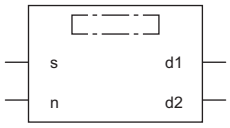
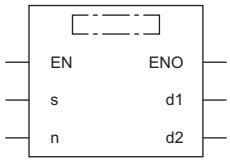

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The set value of the output time exceeds the effective range. |

# 28.3 Off-delay Timer

## TOF(_E), TOF_10(_E)

These function blocks turn off a signal after a specified time.

| Ladder diagram, FBD/LD | | Structured text |
|---|---|---|
| [Without EN/ENO]<br>  <br> | [With EN/ENO]<br>  <br> | [Without EN/ENO]<br>TOF_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TOF_10_1(IN:=s,PT:=n,Q:=d1,ET:=d2);<br>[With EN/ENO]<br>TOF_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2);<br>TOF_10_E_1(EN:=EN,ENO:=ENO,IN:=s,PT:=n,Q:=d1,ET:=d2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| EN | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s(IN) | Time measurement | Input variable | BOOL |
| n(PT) | Delay time setting value | Input variable | TIME |
| ENO | Output status (TRUE: Normal, FALSE: Abnormal) | Output variable | BOOL |
| d1(Q) | Output | Output variable | BOOL |
| d2(ET) | Elapsed time | Output variable | TIME |

### Processing details

#### ■Operation processing

***1.*** Output

- When (s) turns on, (d1) is turned on.
- When (s) turns off from on, (d1) is turned off after the time specified by (n). The elapsed time until (d1) is turned off is set for (d2).
- The timer device is used for counting the elapsed time.

***2.*** Setting of delay time

For TOF(_E), the output time value of (n) is set in units of 100 ms or more. For TOF_10(_E), it is set in units of 10 ms or more. The value of when (s) is turned on from off (falling edge) is used for the setting value of (n). If the value of (n) is changed while (s) is off, the changed value becomes valid at the next falling edge of (s).

## ■Operation result

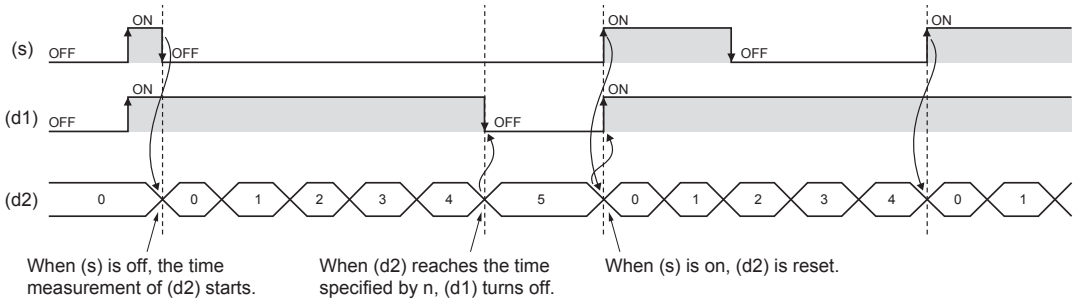
### *1.* Function block without EN/ENO

The following table lists the operation results.

| Operation result | (d1), (d2) |
|---|---|
| No operation error occurred | Operation output value |
| An operation error occurred | Indefinite value |

- Timing chart

When T#5s (5 seconds) is specified in n



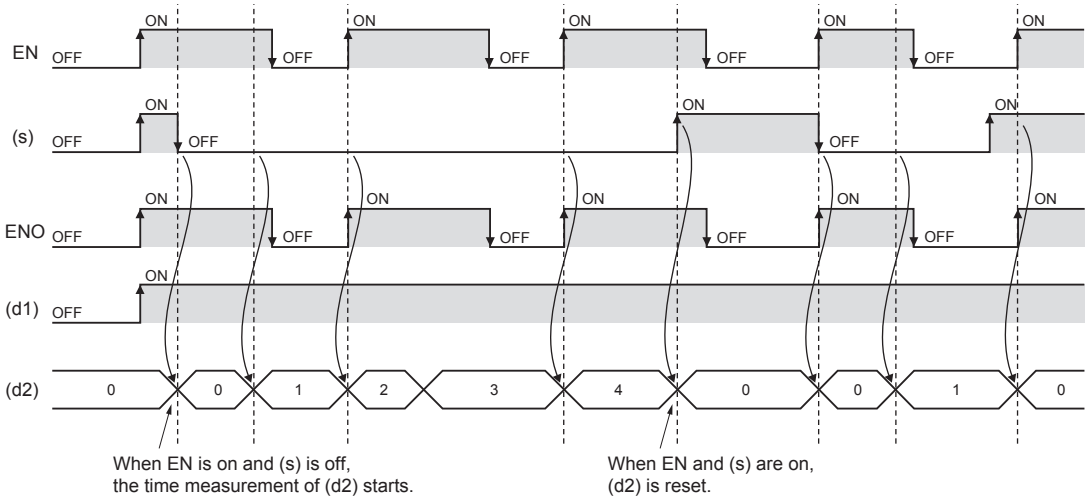

### *2.* Function block with EN/ENO

The following table lists the execution conditions and operation results.

| Execution condition | Operation result | |
|---|---|---|
| **EN** | **ENO** | **(d)** |
| TRUE (Executes operation) | TRUE (Operation error did not occur) | Operation output value |
| | FALSE (Operation error occurred) | Previous output value |
| FALSE (Stops operation) | FALSE | Previous output value |

- Timing chart

When T#5s (5 seconds) is specified in n



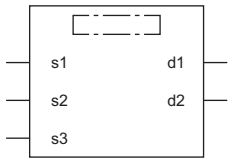

## Operation error

| Error code (SD0/SD8067) | Description |
|---|---|
| 3401H | The set value of the output time exceeds the effective range. |

# 28.4 Timer Function Blocks

## TIMER_□_M

When the execution condition is established, these function blocks start the timer count to the set time.

| Ladder diagram, FBD/LD | Structured text |
|---|---|
|  (□ indicates TIMER_1_FB_M, TIMER_10_FB_M, TIMER_100_FB_M, TIMER_CONT_FB_M, TIMER_CONTHS_FB_M.) | TIMER_1_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);<br>TIMER_10_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);<br>TIMER_100_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);<br>TIMER_CONT_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);<br>TIMER_CONTHS_FB_M_1(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2); |

### Setting data

#### ■Descriptions, types, and data types

| Argument | Description | Type | Data type |
|---|---|---|---|
| s1(Coil) | Execution condition (TRUE: Execution, FALSE: Stop) | Input variable | BOOL |
| s2(Preset) | Timer set value | Input variable | INT |
| s3(ValueIn) | Timer initial value | Input variable | INT |
| d1(ValueOut) | Timer current value | Output variable | ANY16 |
| d2(Status) | Output | Output variable | BOOL |

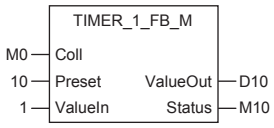
### Processing details

#### ■TIMER_1_FB_M

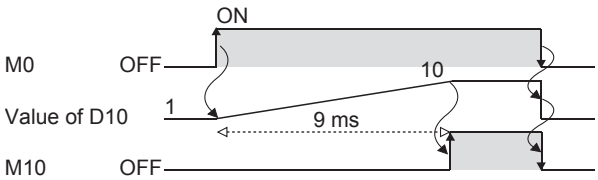
- When the execution condition of (s1) turns on, counting the current value starts. The timer starts counting from "(s3) × 1 ms". When it counts up to "(s2) × 1 ms", (d2) turns on. The current measurement value is output into (d1).
- When the execution condition of (s1) turns off, the current value is reset to (s3) and (d2) turns off.
- A value in the range of 0 to 32767 can be specified for (s2).
- A value in the range of -32768 to 32767 can be specified for (s3). However, when a negative value is specified, the initial value is set to 0.

Ex.

[Ladder example]




[Timing chart]

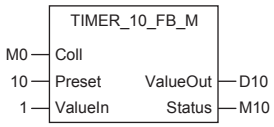


## ■TIMER_10_FB_M

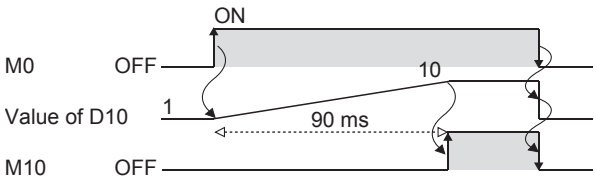
- When the execution condition of (s1) turns on, counting the current value starts. The timer starts counting from "(s3) × 10 ms". When it counts up to "(s2) × 10 ms", (d2) turns on. The current measurement value is output into (d1).
- When the execution condition of (s1) turns off, the current value is reset to (s3) and (d2) turns off.
- A value in the range of 0 to 32767 can be specified for (s2).
- A value in the range of -32768 to 32767 can be specified for (s3). However, when a negative value is specified, the initial value is set to 0.

**Ex.**

[Ladder example]




[Timing chart]

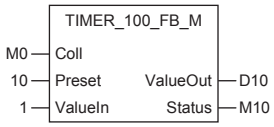



## ■TIMER_100_FB_M

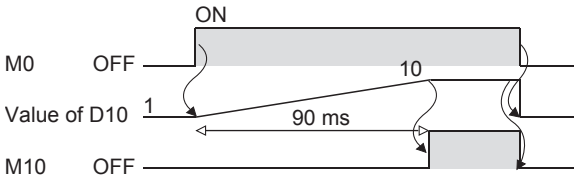
- When the execution condition of (s1) turns on, counting the current value starts. The timer starts counting from "(s3) × 100 ms". When it counts up to "(s2) × 100 ms", (d2) turns on. The current measurement value is output into (d1).
- When the execution condition of (s1) turns off, the current value is reset to (s3) and (d2) turns off.
- A value in the range of 0 to 32767 can be specified for (s2).
- A value in the range of -32768 to 32767 can be specified for (s3). However, when a negative value is specified, the initial value is set to 0.

**Ex.**

[Ladder example]




[Timing chart]



## ■TIMER_CONT_FB_M

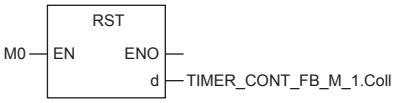
- This is a retentive timer that counts the time when the variable is on. When the execution condition of (s1) turns on, counting the current value starts. There are two retentive timers: low-speed (TIMER_CONT_FB_M) and highspeed (TIMER_CONTHS_FB_M) retentive timers.
- The timer starts counting from "(s3) × 100 ms"( or 1ms if the high-speed retentive timer is used). When it counts up to "(s2) × 100 ms"( or 1ms if the high-speed retentive timer is used), (d2) turns on. The current measurement value is output into (d1).
- The on/off status of (d1) and (d2) is maintained even if the execution condition of (s1) turns off. When the execution condition of (s1) turns on, the timer resume counting from the measurement it holds.
- A value in the range of 0 to 32767 can be specified for (s2).
- A value in the range of -32768 to 32767 can be specified for (s3). However, when a negative value is specified, the initial value is set to 0.
- To reset (d1) of the retentive timer, reset (s1) of FB directly.



**Ex.**

For label name TIMER_CONT_FB_M_1

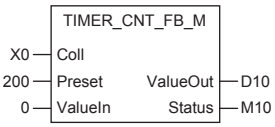
[Ladder program]



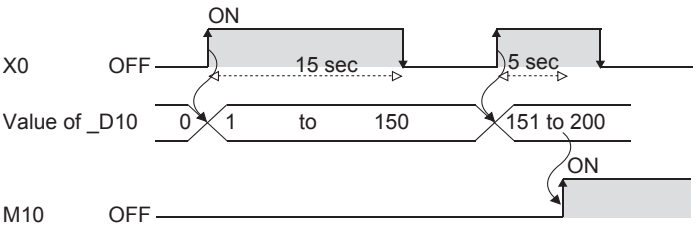

[ST]

RST(M0,TIMER_CONT_FB_M_1.Coil)

[Ladder example]




[Timing chart]




### Operation error

There is no error.

# APPENDICES

## Appendix 1 Reference data: Main Instruction Execution Time

The main instruction execution time is as shown below:

| Instruction | Execution time in ON status (μs) | Execution time in OFF status (μs) | Remarks |
|---|---|---|---|
| LD | 0.034 | | |
| AND | 0.042 | | |
| OUT | 0.024 | | |
| OUT T | 0.509 | 0.317 | |
| OUT C | 0.341 | 0.275 | |
| LD= | 0.042 | | |
| LDD= | 0.042 | | |
| + | 0.059 | 0.017 | In case of 2 operands |
| - | 0.059 | 0.017 | In case of 2 operands |
| D+ | 0.059 | 0.017 | In case of 2 operands |
| D- | 0.059 | 0.017 | |
| * | 0.059 | 0.017 | |
| / | 3.827 | 0.017 | |
| D* | 4.333 | 0.017 | |
| D/ | 4.413 | 0.017 | |
| WAND | 0.059 | 0.017 | In case of 2 operands |
| DAND | 0.059 | 0.017 | In case of 2 operands |
| BCD | 2.556 | 0.017 | |
| DBCD | 2.931 | 0.017 | |
| BIN | 2.381 | 0.017 | |
| DBIN | 2.66 | 0.017 | |
| MOV | 0.034 | 0.017 | |
| DMOV | 0.034 | 0.017 | |

# Appendix 2 Number of Instruction Steps



The number of instruction steps are shown below.

The number of steps may increase depending on the contents of the source and destination (such as specification of BFM or character string), bit specification of word device, nibble specification of bit devices, indexing etc.

| Instruction name | Number of minimum steps |
|---|---|
| LD | 2 |
| LDI | 2 |
| AND | 2 |
| ANI | 2 |
| OR | 2 |
| ORI | 2 |
| LDP | 4 |
| LDF | 4 |
| ANDP | 4 |
| ANDF | 4 |
| ORP | 4 |
| ORF | 4 |
| LDPI | 4 |
| LDFI | 4 |
| ANDPI | 4 |
| ANDFI | 4 |
| ORPI | 4 |
| ORFI | 4 |
| ANB | 1 |
| ORB | 1 |
| MPS | 1 |
| MRD | 1 |
| MPP | 1 |
| INV | 1 |
| MEP | 3 |
| MEF | 3 |
| OUT | 2 |
| OUT T/ST | 5 |
| OUTH T/ ST | 5 |
| OUTHS T/ST | 5 |
| OUT C | 5 |
| OUT LC | 5 |
| OUT F | 3 |
| SET | 2 |
| RST | 2 |
| SET F | 3 |
| RST F | 3 |
| ANS | 7 |
| ANR | 1 |
| ANRP | 1 |
| PLS | 3 |
| PLF | 3 |
| FF | 3 |
| ALT | 3 |
| ALTP | 3 |
| SFT | 3 |
| SFTP | 3 |

| Instruction name | Number of minimum steps |
|---|---|
| SFR | 4 |
| SFRP | 4 |
| SFL | 4 |
| SFLP | 4 |
| BSFR | 4 |
| BSFRP | 4 |
| BSFL | 4 |
| BSFLP | 4 |
| DSFR | 4 |
| DSFRP | 4 |
| DSFL | 4 |
| DSFLP | 4 |
| SFTR | 6 |
| SFTRP | 6 |
| SFTL | 6 |
| SFTLP | 6 |
| WSFR | 6 |
| WSFRP | 6 |
| WSFL | 6 |
| WSFLP | 6 |
| MC | 5 |
| MCR | 3 |
| FEND | 1 |
| END | 1 |
| STOP | 1 |
| LD= | 4 |
| LD<> | 4 |
| LD> | 4 |
| LD<= | 4 |
| LD< | 4 |
| LD>= | 4 |
| AND= | 4 |
| AND<> | 4 |
| AND> | 4 |
| AND<= | 4 |
| AND< | 4 |
| AND>= | 4 |
| OR= | 4 |
| OR<> | 4 |
| OR> | 4 |
| OR<= | 4 |
| OR< | 4 |
| OR>= | 4 |
| LD=_U | 4 |
| LD<>_U | 4 |
| LD>_U | 4 |
| LD<=_U | 4 |
| LD<_U | 4 |
| LD>=_U | 4 |
| AND=_U | 4 |
| AND<>_U | 4 |
| AND>_U | 4 |
| AND<=_U | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| AND<_U | 4 |
| AND>=_U | 4 |
| OR=_U | 4 |
| OR<>_U | 4 |
| OR>_U | 4 |
| OR<=_U | 4 |
| OR<_U | 4 |
| OR>=_U | 4 |
| LDD= | 4 |
| LDD<> | 4 |
| LDD> | 4 |
| LDD<= | 4 |
| LDD< | 4 |
| LDD>= | 4 |
| ANDD= | 4 |
| ANDD<> | 4 |
| ANDD> | 4 |
| ANDD<= | 4 |
| ANDD< | 4 |
| ANDD>= | 4 |
| ORD= | 4 |
| ORD<> | 4 |
| ORD> | 4 |
| ORD<= | 4 |
| ORD< | 4 |
| ORD>= | 4 |
| LDD=_U | 4 |
| LDD<>_U | 4 |
| LDD>_U | 4 |
| LDD<=_U | 4 |
| LDD<_U | 4 |
| LDD>=_U | 4 |
| ANDD=_U | 4 |
| ANDD<>_U | 4 |
| ANDD>_U | 4 |
| ANDD<=_U | 4 |
| ANDD<_U | 4 |
| ANDD>=_U | 4 |
| ORD=_U | 4 |
| ORD<>_U | 4 |
| ORD>_U | 4 |
| ORD<=_U | 4 |
| ORD<_U | 4 |
| ORD>=_U | 4 |
| CMP | 5 |
| CMPP | 5 |
| CMP_U | 5 |
| CMPP_U | 5 |
| DCMP | 5 |
| DCMPP | 5 |
| DCMP_U | 5 |
| DCMPP_U | 5 |
| ZCP | 6 |

| Instruction name | Number of minimum steps |
|---|---|
| ZCPP | 6 |
| ZCP_U | 6 |
| ZCPP_U | 6 |
| DZCP | 6 |
| DZCPP | 6 |
| DZCP_U | 6 |
| DZCPP_U | 6 |
| BKCMP= | 6 |
| BKCMP<> | 6 |
| BKCMP> | 6 |
| BKCMP<= | 6 |
| BKCMP< | 6 |
| BKCMP>= | 6 |
| BKCMP=P | 6 |
| BKCMP<>P | 6 |
| BKCMP>P | 6 |
| BKCMP<=P | 6 |
| BKCMP<P | 6 |
| BKCMP>=P | 6 |
| BKCMP=_U | 6 |
| BKCMP<>_U | 6 |
| BKCMP>_U | 6 |
| BKCMP<=_U | 6 |
| BKCMP<_U | 6 |
| BKCMP>=_U | 6 |
| BKCMP=P_U | 6 |
| BKCMP<>P_U | 6 |
| BKCMP>P_U | 6 |
| BKCMP<=P_U | 6 |
| BKCMP<P_U | 6 |
| BKCMP>=P_U | 6 |
| DBKCMP= | 6 |
| DBKCMP<> | 6 |
| DBKCMP> | 6 |
| DBKCMP<= | 6 |
| DBKCMP< | 6 |
| DBKCMP>= | 6 |
| DBKCMP=P | 6 |
| DBKCMP<>P | 6 |
| DBKCMP>P | 6 |
| DBKCMP<=P | 6 |
| DBKCMP<P | 6 |
| DBKCMP>=P | 6 |
| DBKCMP=_U | 6 |
| DBKCMP<>_U | 6 |
| DBKCMP>_U | 6 |
| DBKCMP<=_U | 6 |
| DBKCMP<_U | 6 |
| DBKCMP>=_U | 6 |
| DBKCMP=P_U | 6 |
| DBKCMP<>P_U | 6 |
| DBKCMP>P_U | 6 |
| DBKCMP<=P_U | 6 |

| Instruction name | Number of minimum steps |
|---|---|
| DBKCMP<P_U | 6 |
| DBKCMP>=P_U | 6 |
| + (s) (d) | 5 |
| +P (s) (d) | 5 |
| + (s1) (s2) (d) | 5 |
| +P (s1) (s2) (d) | 5 |
| +_U (s) (d) | 5 |
| +P_U (s) (d) | 5 |
| +_U (s1) (s2) (d) | 5 |
| +P_U (s1) (s2) (d) | 5 |
| ADD | 5 |
| ADDP | 5 |
| ADD_U | 5 |
| ADDP_U | 5 |
| - (s) (d) | 5 |
| -P (s) (d) | 5 |
| - (s1) (s2) (d) | 5 |
| -P (s1) (s2) (d) | 5 |
| -_U (s) (d) | 5 |
| -P_U (s) (d) | 5 |
| -_U (s1) (s2) (d) | 5 |
| -P_U (s1) (s2) (d) | 5 |
| SUB | 5 |
| SUBP | 5 |
| SUB_U | 5 |
| SUBP_U | 5 |
| D+ (s) (d) | 5 |
| D+P (s) (d) | 5 |
| D+ (s1) (s2) (d) | 5 |
| D+P (s1) (s2) (d) | 5 |
| D+_U (s) (d) | 5 |
| D+P_U (s) (d) | 5 |
| D+_U (s1) (s2) (d) | 5 |
| D+P_U (s1) (s2) (d) | 5 |
| DADD | 5 |
| DADDP | 5 |
| DADD_U | 5 |
| DADDP_U | 5 |
| D- (s) (d) | 5 |
| D-P (s) (d) | 5 |
| D- (s1) (s2) (d) | 5 |
| D-P (s1) (s2) (d) | 5 |
| D-_U (s) (d) | 5 |
| D-P_U (s) (d) | 5 |
| D-_U (s1) (s2) (d) | 5 |
| D-P_U (s1) (s2) (d) | 5 |
| DSUB | 5 |
| DSUBP | 5 |
| DSUB_U | 5 |
| DSUBP_U | 5 |
| * | 7 |
| *P | 7 |
| *_U | 7 |

| Instruction name | Number of minimum steps |
|---|---|
| *P_U | 7 |
| MUL | 7 |
| MULP | 7 |
| MUL_U | 7 |
| MULP_U | 7 |
| / | 7 |
| /P | 7 |
| /_U | 7 |
| /P_U | 7 |
| DIV | 7 |
| DIVP | 7 |
| DIV_U | 7 |
| DIVP_U | 7 |
| D* | 7 |
| D*P | 7 |
| D*_U | 7 |
| D*P_U | 7 |
| DMUL | 7 |
| DMULP | 7 |
| DMUL_U | 7 |
| DMULP_U | 7 |
| D/ | 7 |
| D/P | 7 |
| D/_U | 7 |
| D/P_U | 7 |
| DDIV | 7 |
| DDIVP | 7 |
| DDIV_U | 7 |
| DDIVP_U | 7 |
| B+ (s) (d) | 4 |
| B+P (s) (d) | 4 |
| B+ (s1) (s2) (d) | 5 |
| B+P (s1) (s2) (d) | 5 |
| B- (s) (d) | 4 |
| B-P (s) (d) | 4 |
| B- (s1) (s2) (d) | 5 |
| B-P (s1) (s2) (d) | 5 |
| DB+ (s) (d) | 4 |
| DB+P (s) (d) | 4 |
| DB+ (s1) (s2) (d) | 5 |
| DB+P (s1) (s2) (d) | 5 |
| DB- (s) (d) | 4 |
| DB-P (s) (d) | 4 |
| DB- (s1) (s2) (d) | 5 |
| DB-P (s1) (s2) (d) | 5 |
| B* | 7 |
| B*P | 7 |
| B/ | 7 |
| B/P | 7 |
| DB* | 7 |
| DB*P | 7 |
| DB/ | 7 |
| DB/P | 7 |

| Instruction name | Number of minimum steps |
|---|---|
| BK+ | 6 |
| BK+P | 6 |
| BK+_U | 6 |
| BK+P_U | 6 |
| BK- | 6 |
| BK-P | 6 |
| BK-_U | 6 |
| BK-P_U | 6 |
| DBK+ | 6 |
| DBK+P | 6 |
| DBK+_U | 6 |
| DBK+P_U | 6 |
| DBK- | 6 |
| DBK-P | 6 |
| DBK-_U | 6 |
| DBK-P_U | 6 |
| INC | 3 |
| INCP | 3 |
| INC_U | 3 |
| INCP_U | 3 |
| DEC | 3 |
| DECP | 3 |
| DEC_U | 3 |
| DECP_U | 3 |
| DINC | 3 |
| DINCP | 3 |
| DINC_U | 3 |
| DINCP_U | 3 |
| DDEC | 3 |
| DDECP | 3 |
| DDEC_U | 3 |
| DDECP_U | 3 |
| WAND (s) (d) | 5 |
| WANDP (s) (d) | 5 |
| WAND (s1) (s2) (d) | 5 |
| WANDP (s1) (s2) (d) | 5 |
| DAND (s) (d) | 5 |
| DANDP (s) (d) | 5 |
| DAND (s1) (s2) (d) | 5 |
| DANDP (s1) (s2) (d) | 5 |
| BKAND | 6 |
| BKANDP | 6 |
| WOR (s) (d) | 5 |
| WORP (s) (d) | 5 |
| WOR (s1) (s2) (d) | 5 |
| WORP (s1) (s2) (d) | 5 |
| DOR (s) (d) | 5 |
| DORP (s) (d) | 5 |
| DOR (s1) (s2) (d) | 5 |
| DORP (s1) (s2) (d) | 5 |
| BKOR | 6 |
| BKORP | 6 |
| WXOR (s) (d) | 5 |

| Instruction name | Number of minimum steps |
|---|---|
| WXORP (s) (d) | 5 |
| WXOR (s1) (s2) (d) | 5 |
| WXORP (s1) (s2) (d) | 5 |
| DXOR (s) (d) | 5 |
| DXORP (s) (d) | 5 |
| DXOR (s1) (s2) (d) | 5 |
| DXORP (s1) (s2) (d) | 5 |
| BKXOR | 6 |
| BKXORP | 6 |
| WXNR (s) (d) | 5 |
| WXNRP (s) (d) | 5 |
| WXNR (s1) (s2) (d) | 5 |
| WXNRP (s1) (s2) (d) | 5 |
| DXNR (s) (d) | 5 |
| DXNRP (s) (d) | 5 |
| DXNR (s1) (s2) (d) | 5 |
| DXNRP (s1) (s2) (d) | 5 |
| BKXNR | 6 |
| BKXNRP | 6 |
| BSET | 4 |
| BSETP | 4 |
| BRST | 4 |
| BRSTP | 4 |
| TEST | 5 |
| TESTP | 5 |
| DTEST | 5 |
| DTESTP | 5 |
| BKRST | 4 |
| BKRSTP | 4 |
| ZRST | 4 |
| ZRSTP | 4 |
| BCD | 4 |
| BCDP | 4 |
| DBCD | 4 |
| DBCDP | 4 |
| BIN | 4 |
| BINP | 4 |
| DBIN | 4 |
| DBINP | 4 |
| FLT2INT | 4 |
| FLT2INTP | 4 |
| FLT2UINT | 4 |
| FLT2UINTP | 4 |
| FLT2DINT | 4 |
| FLT2DINTP | 4 |
| FLT2UDINT | 4 |
| FLT2UDINTP | 4 |
| INT2UINT | 4 |
| INT2UINTP | 4 |
| INT2DINT | 4 |
| INT2DINTP | 4 |
| INT2UDINT | 4 |
| INT2UDINTP | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| UINT2INT | 4 |
| UINT2INTP | 4 |
| UINT2DINT | 4 |
| UINT2DINTP | 4 |
| UINT2UDINT | 4 |
| UINT2UDINTP | 4 |
| DINT2INT | 4 |
| DINT2INTP | 4 |
| DINT2UINT | 4 |
| DINT2UINTP | 4 |
| DINT2UDINT | 4 |
| DINT2UDINTP | 4 |
| UDINT2INT | 4 |
| UDINT2INTP | 4 |
| UDINT2UINT | 4 |
| UDINT2UINTP | 4 |
| UDINT2DINT | 4 |
| UDINT2DINTP | 4 |
| GRY | 4 |
| GRYP | 4 |
| GRY_U | 4 |
| GRYP_U | 4 |
| DGRY | 4 |
| DGRYP | 4 |
| DGRY_U | 4 |
| DGRYP_U | 4 |
| GBIN | 4 |
| GBINP | 4 |
| GBIN_U | 4 |
| GBINP_U | 4 |
| DGBIN | 4 |
| DGBINP | 4 |
| DGBIN_U | 4 |
| DGBINP_U | 4 |
| DABIN | 4 |
| DABINP | 4 |
| DABIN_U | 4 |
| DABINP_U | 4 |
| DDABIN | 4 |
| DDABINP | 4 |
| DDABIN_U | 4 |
| DDABINP_U | 4 |
| HEXA | 5 |
| HEXAP | 5 |
| VAL | 5 |
| VALP | 5 |
| VAL_U | 5 |
| VALP_U | 5 |
| DVAL | 5 |
| DVALP | 5 |
| DVAL_U | 5 |
| DVALP_U | 5 |
| NEG | 3 |

| Instruction name | Number of minimum steps |
|---|---|
| NEGP | 3 |
| DNEG | 3 |
| DNEGP | 3 |
| DECO | 5 |
| DECOP | 5 |
| ENCO | 5 |
| ENCOP | 5 |
| SEGD | 4 |
| SEGDP | 4 |
| SEGL | 5 |
| DIS | 5 |
| DISP | 5 |
| UNI | 5 |
| UNIP | 5 |
| NDIS | 5 |
| NDISP | 5 |
| NUNI | 5 |
| NUNIP | 5 |
| WTOB | 5 |
| WTOBP | 5 |
| BTOW | 5 |
| BTOWP | 5 |
| DSW | 6 |
| MOV | 4 |
| MOVP | 4 |
| DMOV | 4 |
| DMOVP | 4 |
| CML | 4 |
| CMLP | 4 |
| DCML | 4 |
| DCMLP | 4 |
| SMOV | 7 |
| SMOVP | 7 |
| CMLB | 4 |
| CMLBP | 4 |
| BMOV | 5 |
| BMOVP | 5 |
| FMOV | 5 |
| FMOVP | 5 |
| DFMOV | 5 |
| DFMOVP | 5 |
| XCH | 4 |
| XCHP | 4 |
| DXCH | 4 |
| DXCHP | 4 |
| SWAP | 3 |
| SWAPP | 3 |
| DSWAP | 3 |
| DSWAPP | 3 |
| MOVB | 4 |
| MOVBP | 4 |
| PRUN | 5 |
| PRUNP | 5 |

| Instruction name | Number of minimum steps |
|---|---|
| DPRUN | 5 |
| DPRUNP | 5 |
| BLKMOVB | 5 |
| BLKMOVBP | 5 |
| ROR | 4 |
| RORP | 4 |
| RCR | 4 |
| RCRP | 4 |
| DROR | 4 |
| DRORP | 4 |
| DRCR | 4 |
| DRCRP | 4 |
| ROL | 4 |
| ROLP | 4 |
| RCL | 4 |
| RCLP | 4 |
| DROL | 4 |
| DROLP | 4 |
| DRCL | 4 |
| DRCLP | 4 |
| CJ | 3 |
| CJP | 3 |
| GOEND | 1 |
| DI | 1 |
| DI (s) | 3 |
| EI | 1 |
| IMASK | 3 |
| SIMASK | 5 |
| IRET | 1 |
| WDT | 1 |
| WDTP | 1 |
| FOR | 3 |
| NEXT | 1 |
| BREAK | 5 |
| BREAKP | 5 |
| CALL | 3 |
| CALLP | 3 |
| RET | 1 |
| SRET | 1 |
| XCALL | 3 |
| SFRD | 5 |
| SFRDP | 5 |
| POP | 5 |
| POPP | 5 |
| SFWR | 5 |
| SFWRP | 5 |
| FINS | 5 |
| FINSP | 5 |
| FDEL | 5 |
| FDELP | 5 |
| LD$= | 4 |
| LD$<> | 4 |
| LD$> | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| LD$<= | 4 |
| LD$< | 4 |
| LD$>= | 4 |
| AND$= | 4 |
| AND$<> | 4 |
| AND$> | 4 |
| AND$<= | 4 |
| AND$< | 4 |
| AND$>= | 4 |
| OR$= | 4 |
| OR$<> | 4 |
| OR$> | 4 |
| OR$<= | 4 |
| OR$< | 4 |
| OR$>= | 4 |
| $+ (s) (d) | 4 |
| $+P (s) (d) | 4 |
| $+ (s1) (s2) (d) | 5 |
| $+P (s1) (s2) (d) | 5 |
| $MOV | 4 |
| $MOVP | 4 |
| BINDA | 4 |
| BINDAP | 4 |
| BINDA_U | 4 |
| BINDAP_U | 4 |
| DBINDA | 4 |
| DBINDAP | 4 |
| DBINDA_U | 4 |
| DBINDAP_U | 4 |
| ASCI | 5 |
| ASCIP | 5 |
| STR | 5 |
| STRP | 5 |
| STR_U | 5 |
| STRP_U | 5 |
| DSTR | 5 |
| DSTRP | 5 |
| DSTR_U | 5 |
| DSTRP_U | 5 |
| ESTR | 5 |
| ESTRP | 5 |
| DESTR | 5 |
| DESTRP | 5 |
| LEN | 4 |
| LENP | 4 |
| RIGHT | 5 |
| RIGHTP | 5 |
| LEFT | 5 |
| LEFTP | 5 |
| MIDR | 5 |
| MIDRP | 5 |
| MIDW | 5 |
| MIDWP | 5 |

| Instruction name | Number of minimum steps |
|---|---|
| INSTR | 6 |
| INSTRP | 6 |
| STRINS | 5 |
| STRINSP | 5 |
| STRDEL | 5 |
| STRDELP | 5 |
| LDE= | 4 |
| LDE<> | 4 |
| LDE> | 4 |
| LDE<= | 4 |
| LDE< | 4 |
| LDE>= | 4 |
| ANDE= | 4 |
| ANDE<> | 4 |
| ANDE> | 4 |
| ANDE<= | 4 |
| ANDE< | 4 |
| ANDE>= | 4 |
| ORE= | 4 |
| ORE<> | 4 |
| ORE> | 4 |
| ORE<= | 4 |
| ORE< | 4 |
| ORE>= | 4 |
| DECMP | 5 |
| DECMPP | 5 |
| DEZCP | 6 |
| DEZCPP | 6 |
| E+ (s) (d) | 4 |
| E+P (s) (d) | 4 |
| E+ (s1) (s2) (d) | 5 |
| E+P (s1) (s2) (d) | 5 |
| DEADD | 5 |
| DEADDP | 5 |
| E- (s) (d) | 4 |
| E-P (s) (d) | 4 |
| E- (s1) (s2) (d) | 5 |
| E-P (s1) (s2) (d) | 5 |
| DESUB | 5 |
| DESUBP | 5 |
| E* | 5 |
| E*P | 5 |
| DEMUL | 5 |
| DEMULP | 5 |
| E/ | 5 |
| E/P | 5 |
| DEDIV | 5 |
| DEDIVP | 5 |
| INT2FLT | 4 |
| INT2FLTP | 4 |
| UINT2FLT | 4 |
| UINT2FLTP | 4 |
| DINT2FLT | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| DINT2FLTP | 4 |
| UDINT2FLT | 4 |
| UDINT2FLTP | 4 |
| EVAL | 4 |
| EVALP | 4 |
| DEVAL | 4 |
| DEVALP | 4 |
| DEBCD | 4 |
| DEBCDP | 4 |
| DEBIN | 4 |
| DEBINP | 4 |
| ENEG | 3 |
| ENEGP | 3 |
| DENEG | 3 |
| DENEGP | 3 |
| EMOV | 4 |
| EMOVP | 4 |
| DEMOV | 4 |
| DEMOVP | 4 |
| SIN | 4 |
| SINP | 4 |
| DSIN | 4 |
| DSINP | 4 |
| COS | 4 |
| COSP | 4 |
| DCOS | 4 |
| DCOSP | 4 |
| TAN | 4 |
| TANP | 4 |
| DTAN | 4 |
| DTANP | 4 |
| ASIN | 4 |
| ASINP | 4 |
| DASIN | 4 |
| DASINP | 4 |
| ACOS | 4 |
| ACOSP | 4 |
| DACOS | 4 |
| DACOSP | 4 |
| ATAN | 4 |
| ATANP | 4 |
| DATAN | 4 |
| DATANP | 4 |
| RAD | 4 |
| RADP | 4 |
| DRAD | 4 |
| DRADP | 4 |
| DEG | 4 |
| DEGP | 4 |
| DDEG | 4 |
| DDEGP | 4 |
| DESQR | 4 |
| DESQRP | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| ESQRT | 4 |
| ESQRTP | 4 |
| EXP | 4 |
| EXPP | 4 |
| DEXP | 4 |
| DEXPP | 4 |
| LOG | 4 |
| LOGP | 4 |
| DLOGE | 4 |
| DLOGEP | 4 |
| POW | 5 |
| POWP | 5 |
| LOG10 | 4 |
| LOG10P | 4 |
| DLOG10 | 4 |
| DLOG10P | 4 |
| EMAX | 5 |
| EMAXP | 5 |
| EMIN | 5 |
| EMINP | 5 |
| RND | 3 |
| RNDP | 3 |
| ZPUSH (d) | 3 |
| ZPUSHP (d) | 3 |
| ZPUSH (s) (d) | 4 |
| ZPUSHP (s) (d) | 4 |
| ZPOP (d) | 3 |
| ZPOPP (d) | 3 |
| ZPOP (s) (d) | 4 |
| ZPOPP (s) (d) | 4 |
| LIMIT | 6 |
| LIMITP | 6 |
| LIMIT_U | 6 |
| LIMITP_U | 6 |
| DLIMIT | 6 |
| DLIMITP | 6 |
| DLIMIT_U | 6 |
| DLIMITP_U | 6 |
| BAND | 6 |
| BANDP | 6 |
| BAND_U | 6 |
| BANDP_U | 6 |
| DBAND | 6 |
| DBANDP | 6 |
| DBAND_U | 6 |
| DBANDP_U | 6 |
| ZONE | 6 |
| ZONEP | 6 |
| ZONE_U | 6 |
| ZONEP_U | 6 |
| DZONE | 6 |
| DZONEP | 6 |
| DZONE_U | 6 |

| Instruction name | Number of minimum steps |
|---|---|
| DZONEP_U | 6 |
| SCL | 5 |
| SCLP | 5 |
| SCL_U | 5 |
| SCLP_U | 5 |
| DSCL | 5 |
| DSCLP | 5 |
| DSCL_U | 5 |
| DSCLP_U | 5 |
| SCL2 | 5 |
| SCL2P | 5 |
| SCL2_U | 5 |
| SCL2P_U | 5 |
| DSCL2 | 5 |
| DSCL2P | 5 |
| DSCL2_U | 5 |
| DSCL2P_U | 5 |
| TTMR | 4 |
| STMR | 7 |
| UDCNTF | 5 |
| ROTC | 6 |
| RAMPF | 6 |
| SPD | 5 |
| DSPD | 5 |
| PLSY | 5 |
| DPLSY | 5 |
| PWM | 5 |
| DPWM | 5 |
| MTR | 6 |
| IST | 7 |
| ABSD | 9 |
| DABSD | 9 |
| INCD | 9 |
| CCD | 5 |
| CCDP | 5 |
| SERMM | 6 |
| SERMMP | 6 |
| DSERMM | 6 |
| DSERMMP | 6 |
| SUM | 4 |
| SUMP | 4 |
| DSUM | 4 |
| DSUMP | 4 |
| BON | 5 |
| BONP | 5 |
| DBON | 5 |
| DBONP | 5 |
| MAX | 5 |
| MAXP | 5 |
| MAX_U | 5 |
| MAXP_U | 5 |
| DMAX | 5 |
| DMAXP | 5 |

| Instruction name | Number of minimum steps |
|---|---|
| DMAX_U | 5 |
| DMAXP_U | 5 |
| MIN | 5 |
| MINP | 5 |
| MIN_U | 5 |
| MINP_U | 5 |
| DMIN | 5 |
| DMINP | 5 |
| DMIN_U | 5 |
| DMINP_U | 5 |
| SORTTBL | 7 |
| SORTTBL_U | 7 |
| SORTTBL2 | 7 |
| SORTTBL2_U | 7 |
| DSORTTBL2 | 7 |
| DSORTTBL2_U | 7 |
| WSUM | 7 |
| WSUMP | 7 |
| WSUM_U | 7 |
| WSUMP_U | 7 |
| DWSUM | 7 |
| DWSUMP | 7 |
| DWSUM_U | 7 |
| DWSUMP_U | 7 |
| MEAN | 5 |
| MEANP | 5 |
| MEAN_U | 5 |
| MEANP_U | 5 |
| DMEAN | 5 |
| DMEANP | 5 |
| DMEAN_U | 5 |
| DMEANP_U | 5 |
| SQRT | 4 |
| SQRTP | 4 |
| DSQRT | 4 |
| DSQRTP | 4 |
| CRC | 5 |
| CRCP | 5 |
| ADRSET | 4 |
| ADRSETP | 4 |
| TRD | 3 |
| TRDP | 3 |
| TWR | 3 |
| TWRP | 3 |
| TADD | 5 |
| TADDP | 5 |
| TSUB | 5 |
| TSUBP | 5 |
| HTOS | 4 |
| HTOSP | 4 |
| DHTOS | 4 |
| DHTOSP | 4 |
| STOH | 4 |

| Instruction name | Number of minimum steps |
|---|---|
| STOHP | 4 |
| DSTOH | 4 |
| DSTOHP | 4 |
| LDDT= | 5 |
| LDDT<> | 5 |
| LDDT> | 5 |
| LDDT<= | 5 |
| LDDT< | 5 |
| LDDT>= | 5 |
| ANDDT= | 5 |
| ANDDT<> | 5 |
| ANDDT> | 5 |
| ANDDT<= | 5 |
| ANDDT< | 5 |
| ANDDT>= | 5 |
| ORDT= | 5 |
| ORDT<> | 5 |
| ORDT> | 5 |
| ORDT<= | 5 |
| ORDT< | 5 |
| ORDT>= | 5 |
| LDTM= | 5 |
| LDTM<> | 5 |
| LDTM> | 5 |
| LDTM<= | 5 |
| LDTM< | 5 |
| LDTM>= | 5 |
| ANDTM= | 5 |
| ANDTM<> | 5 |
| ANDTM> | 5 |
| ANDTM<= | 5 |
| ANDTM< | 5 |
| ANDTM>= | 5 |
| ORTM= | 5 |
| ORTM<> | 5 |
| ORTM> | 5 |
| ORTM<= | 5 |
| ORTM< | 5 |
| ORTM>= | 5 |
| TCMP | 7 |
| TCMPP | 7 |
| TZCP | 6 |
| TZCPP | 6 |
| DUTY | 5 |
| HOURM | 5 |
| DHOURM | 5 |
| REF | 4 |
| REFP | 4 |
| RFS | 4 |
| RFSP | 4 |
| FROM | 6 |
| FROMP | 6 |
| DFROM | 6 |

| Instruction name | Number of minimum steps |
|---|---|
| DFROMP | 6 |
| TO | 6 |
| TOP | 6 |
| DTO | 6 |
| DTOP | 6 |
| FROMD | 6 |
| FROMDP | 6 |
| DFROMD | 6 |
| DFROMDP | 6 |
| TOD | 6 |
| TODP | 6 |
| DTOD | 6 |
| DTODP | 6 |
| STL | 3 |
| RETSTL | 1 |
| SP.SOCOPEN | 10 |
| SP.SOCCLOSE | 10 |
| SP.SOCRCV | 12 |
| SP.SOCSND | 12 |
| SP.SOCCINF | 10 |
| S.SOCRDATA | 12 |
| SP.SOCRDATA | 12 |
| SP.ECPRTCL | 7 |
| PID | 6 |
| DHSCS | 5 |
| DHSCR | 5 |
| DHSZ | 6 |
| HIOEN | 5 |
| HIOENP | 5 |
| DHIOEN | 5 |
| DHIOENP | 5 |
| HCMOV | 5 |
| HCMOVP | 5 |
| DHCMOV | 5 |
| DHCMOVP | 5 |
| RS2 | 7 |
| IVCK | 7 |
| IVDR | 7 |
| IVRD | 7 |
| IVWR | 7 |
| IVBWR | 7 |
| IVMC | 13 |
| ADPRW | 13 |
| S.CPRTCL | 7 |
| SP.CPRTCL | 7 |
| DSZR | 6 |
| DDSZR | 6 |
| DVIT | 6 |
| DDVIT | 6 |
| TBL | 4 |
| DRVTBL | 7 |
| DRVMUL | 13 |
| DABS | 5 |

| Instruction name | Number of minimum steps |
|---|---|
| PLSV | 5 |
| DPLSV | 5 |
| DRVI | 6 |
| DDRVI | 6 |
| DRVA | 6 |
| DDRVA | 6 |
| RBFM | 7 |
| WBFM | 7 |

# Appendix 3 Adding and Changing Functions



The functions added or changed with the CPU module and engineering tool, and the supported CPU modules' firmware version and engineering tool software version are given below.

- The firmware version can be confirmed with module diagnosis (CPU diagnosis). Refer to the User's Manual (Hardware) for the CPU module in use for details on diagnosing the module (CPU diagnosis).
- Refer to the GX Works3 Operating Manual for details on the software version.

| Add/Change Function | Supported CPU module firmware version | Supported engineering tool software version | Reference |
|---|---|---|---|
| The number of settable high-speed comparison tables was changed from maximum 4 to 32. | "1.015" and above*1 | "1.015R" and above | Page 722 HIOEN(P)<br>Page 725 DHIOEN(P) |
| The number of high-speed comparisons was changed from 4 to 32. | "1.015" and above*1 | "1.015R" and above | Page 722 HIOEN(P)<br>Page 725 DHIOEN(P) |

*1 Supported with CPU module serial No. 158**** and above.

# INSTRUCTION INDEX

## Symbols



## A



## B

## C



## D



I

## E



## F

## G



## H



## I



## L

## M



## N



## O

## P



## R



## S



## T

## U



## V



## W



## X



## Z

# MEMO

# REVISIONS

| Revision date | Revision | Description |
|---|---|---|
| October 2014 | A | First Edition |
| January 2015 | B | ■Added or modified parts<br>Section 6.5, 6.6, 7.12, 7.16, 7.17, Chapter 8, 9, 10, 11, 12, 13, Appendix 1<br>"FBD/LD language" (for each instruction)<br>Error correction |
| February 2015 | C | Error correction |
| April 2015 | D | A part of the cover design is changed. |
| August 2015 | E | ■Added or modified parts<br>RELEVANT MANUALS, Section 8.24, 10.1, Chapter 12, Appendix 3<br>Section 1.4 was changed to Chapter 2. |

# WARRANTY

Please confirm the following product warranty details before using this product.

## 1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company. However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

**[Gratis Warranty Term]**

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place. Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

**[Gratis Warranty Range]**

1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.
2) Even within the gratis warranty term, repairs shall be charged for in the following cases.
   a) Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
   b) Failure caused by unapproved modifications, etc., to the product by the user.
   c) When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
   d) Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
   e) Relay failure or output contact failure caused by usage beyond the specified life of contact (cycles).
   f) Failure caused by external irresistible forces such as fires or abnormal voltages, and failure caused by force majeure such as earthquakes, lightning, wind and water damage.
   g) Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
   h) Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

## 2. Onerous repair term after discontinuation of production

1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued.
   Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.
2) Product supply (including repair parts) is not available after production is discontinued.

## 3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

## 4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation of damages caused by any cause found not to be the responsibility of Mitsubishi, loss in opportunity, lost profits incurred to the user or third person by failure of Mitsubishi products, special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products, replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

## 5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

## 6. Product application

1) In using the Mitsubishi MELSEC programmable controller, the usage conditions shall be that the application will not lead to a major accident even if any problem or fault should occur in the programmable controller device, and that backup and fail-safe functions are systematically provided outside of the device for any problem or fault.
2) The Mitsubishi programmable controller has been designed and manufactured for applications in general industries, etc. Thus, applications in which the public could be affected such as in nuclear power plants and other power plants operated by respective power companies, and applications in which a special quality assurance system is required, such as for railway companies or public service purposes shall be excluded from the programmable controller applications.
   In addition, applications in which human life or property that could be greatly affected, such as in aircraft, medical applications, incineration and fuel devices, manned transportation, equipment for recreation and amusement, and safety devices, shall also be excluded from the programmable controller range of applications.
   However, in certain cases, some applications may be possible, providing the user consults their local Mitsubishi representative outlining the special requirements of the project, and providing that all parties concerned agree to the special circumstances, solely at the user's discretion.

# TRADEMARKS

Microsoft® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Ethernet is a trademark of Xerox Corporation.
MODBUS® is a registered trademark of Schneider Electric SA.
The company name and the product name to be described in this manual are the registered trademarks or trademarks of each company.

Manual number: JY997D55801E
Model: FX5-P-MF-E
Model code: 09R539



MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN