MITSUBISHI ELECTRIC

Programmable Controller

MELSEC iQ-R series

MELSEC iQ-R Programming Manual (Motion Module Instructions, Standard Functions/ Function Blocks)

-RD78G4
-RD78G8
-RD78G16
-RD78G32
-RD78G64
-RD78GHV
-RD78GHW

SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using MELSEC iQ-R series programmable controllers, please read the manuals for the product and the relevant manuals introduced in those manuals carefully, and pay full attention to safety to handle the product correctly.

Make sure that the end users read this manual and then keep the manual in a safe place for future reference.

INTRODUCTION

Thank you for purchasing the Mitsubishi Electric MELSEC iQ-R series programmable controllers.

This manual describes the instructions and standard functions/function blocks required for programming.

Before using this product, please read this manual and the relevant manuals carefully and develop familiarity with the functions and performance of the MELSEC iQ-R series programmable controller to handle the product correctly.

When applying the program examples provided in this manual to an actual system, ensure the applicability and confirm that it will not cause system control problems.

Please make sure that the end users read this manual.

Relevant products

RD78G4, RD78G8, RD78G16, RD78G32, RD78G64, RD78GHV, RD78GHW

CONTENTS

PART 1 OVERVIEW

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

PART 3 SEQUENCE INSTRUCTIONS

PART 4 BASIC INSTRUCTIONS

PART 5 APPLICATION INSTRUCTIONS

CHAPTER 8 DATA PROCESSING 97

CHAPTER 9 STRING PROCESSING 102

CHAPTER 10 REAL VALUE PEOCESSING 105

PART 6 STANDARD FUNCTIONS

CHAPTER 11 TYPE CONVERSION FUNCTIONS 118

CHAPTER 15 SELECTION FUNCTIONS 180

PART 7 STANDARD FUNCTION BLOCKS

CHAPTER 16 BISTABLE FUNCTION BLOCKS 184

CHAPTER 17 EDGE DETECTION FUNCTION BLOCKS 187

CHAPTER 18 TIMER FUNCTION BLOCKS 189

PART 8 MOTION DEDICATED INSTRUCTIONS

CHAPTER 19 MOTION DEDICATED INSTRUCTIONS 202

INDEX 211

INSTRUCTION INDEX 212

RELEVANT MANUALS

Manual name [manual number]	Description	Available form
MELSEC iQ-R Programming Manual (Motion Module Instructions, Standard Functions/Function Blocks) [IB-0300431ENG] (This manual)	Instructions for the Motion module and standard functions/ function blocks	Print book
		e-Manual PDF
MELSEC iQ-R Motion Module User's Manual (Startup) [IB-0300406ENG]	Specifications, procedures before operation, system configuration, and wiring of the Motion module	Print book
		e-Manual PDF
MELSEC iQ-R Motion Module User's Manual (Application) [IB-0300411ENG]	Functions, I/O signals, variables, labels, programming, and troubleshooting of the Motion module	Print book
		e-Manual PDF
MELSEC iQ-R Motion Module User's Manual (Network) [IB-0300426ENG]	Functions, parameter settings, troubleshooting, and buffer memory of CC-Link IE TSN	Print book
		e-Manual PDF
MELSEC iQ-R Programming Manual (Motion Control Function Blocks) [IB-0300533ENG]	Motion control function blocks, variables, and programming	Print book
		e-Manual PDF
Motion Module Quick Start Guide [L03191ENG]	Describes system startup, parameter settings, and programming methods for first-time users of the Motion module	e-Manual PDF
Motion Module Quick Start Guide (PLC CPU Ladder Program) [L03194ENG]	Describes system startup, parameter settings, and programming methods for first-time users of the Motion module	e-Manual PDF
MELSEC iQ-R Programming Manual (Program Design) [SH-081265ENG]	Program specifications (ladder, ST, FBD/LD, and SFC programs)	e-Manual PDF
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3	e-Manual PDF

For programs, refer to the following.

📖MELSEC iQ-R Programming Manual (Program Design)

Point

e-Manual refers to the Mitsubishi Electric FA electronic book manuals that can be browsed using a dedicated tool.

e-Manual has the following features:

- Required information can be cross-searched in multiple manuals.
- Other manuals can be accessed from the links in the manual.
- The hardware specifications of each part can be found from the product figures.
- Pages that users often browse can be bookmarked.
- Sample programs can be copied to an engineering tool.

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
Intelligent function module	A module that has functions other than input and output, such as the A/D and the D/A converter module.
Control CPU	A CPU module that controls connected I/O modules and intelligent function modules. In a multiple CPU system, there are multiple CPU modules and each connected module can be controlled by a different CPU module.
Output variable	An output argument of FB.
Dedicated Instruction	An instruction for using functions of the module.
Device	Various memory data in a module. There are devices handled in each bit and in each word.
Input variable	An input argument of FB.
Buffer memory	Memory in an intelligent function module for storing data such as setting values and monitored values.
Module label	A label that represents one of memory areas (I/O signals and buffer memory areas) specific to each module in a given character string. GX Works3 automatically generates this label, which can be used as a global label in the PLC CPU module.
Label	A label that represents a device in a given character string.
Link device	A device in a module on CC-Link IE.

GENERIC TERMS AND ABBREVIATIONS

Unless otherwise specified, this manual uses the following generic terms and abbreviations.

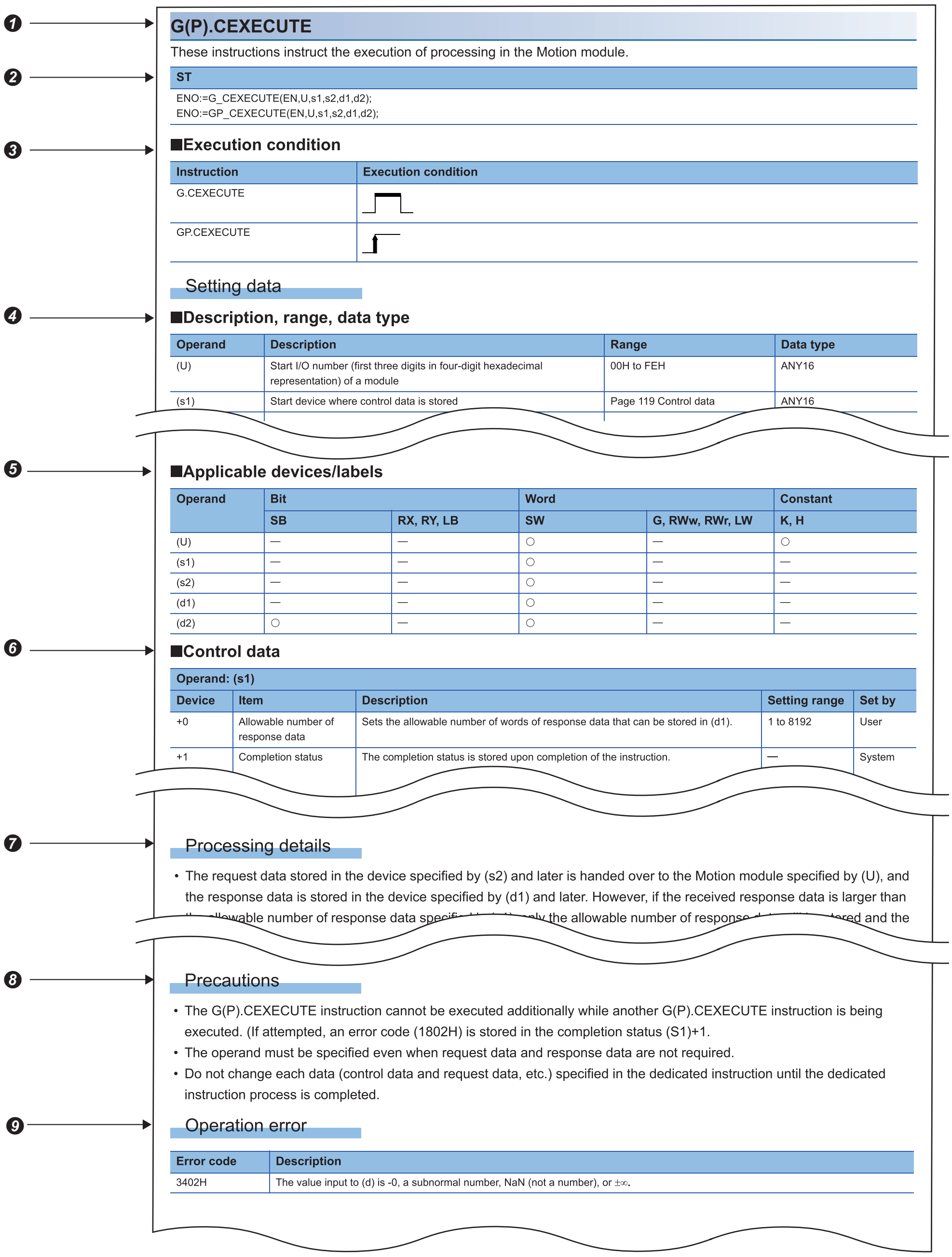
Generic term/abbreviation	Description
A/D converter module	It indicates MELSEC iQ-R series analog-digital converter module, channel isolated analog-digital converter module, and high speed analog-digital converter module.
CPU module	It indicates MELSEC iQ-R series CPU module.
D/A converter module	It indicates MELSEC iQ-R series digital-analog converter module, channel isolated digital-analog converter module, and high speed digital-analog converter module.
MCFB	It indicates Motion Control FB.
ST language	It indicates structured text language.
Engineering tool	It indicates GX Works3 and MR Configurator2.
Operand	It indicates the devices, such as source data (s), destination data (d), number of devices (n), and others, used as parts to configure instructions and functions.
Programmable controller CPU	It indicates the R00CPU, R01CPU, R02CPU, R04CPU, R04ENCPU, R08CPU, R08ENCPU, R16CPU, R16ENCPU, R32CPU, R32ENCPU, R120CPU, and R120ENCPU.
I/O module	It indicates the input module, output module, I/O combined module, and interrupt module.
Motion system	It indicates software that performs the motion control and the network control.
Motion module	It indicates RD78G4, RD78G8, RD78G16, RD78G32, RD78G64, RD78GHV and RD78GHW.

MANUAL PAGE ORGANIZATION

In this manual, pages are organized and the symbols are used as shown below.

How to read Part 3 to Part 6

The following illustration is for explanation purpose only, and should not be referred to as an actual documentation.



G(P).CEXECUTE

These instructions instruct the execution of processing in the Motion module.

ST

ENO:=G_CEXECUTE(EN,U,s1,s2,d1,d2);
ENO:=GP_CEXECUTE(EN,U,s1,s2,d1,d2);

■Execution condition

Instruction	Execution condition
G.CEXECUTE	
GP.CEXECUTE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(U)	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	ANY16
(s1)	Start device where control data is stored	Page 119 Control data	ANY16

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(U)	—	—	○	—	○
(s1)	—	—	○	—	—
(s2)	—	—	○	—	—
(d1)	—	—	○	—	—
(d2)	○	—	○	—	—

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Allowable number of response data	Sets the allowable number of words of response data that can be stored in (d1).	1 to 8192	User
+1	Completion status	The completion status is stored upon completion of the instruction.	—	System

Processing details

- The request data stored in the device specified by (s2) and later is handed over to the Motion module specified by (U), and the response data is stored in the device specified by (d1) and later. However, if the received response data is larger than the allowable number of response data specified by (s1), only the allowable number of response data will be stored and the

Precautions

- The G(P).CEXECUTE instruction cannot be executed additionally while another G(P).CEXECUTE instruction is being executed. (If attempted, an error code (1802H) is stored in the completion status (S1)+1.
- The operand must be specified even when request data and response data are not required.
- Do not change each data (control data and request data, etc.) specified in the dedicated instruction until the dedicated instruction process is completed.

Operation error

Error code	Description
3402H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

❶ Instruction symbol

- An instruction symbol followed by parentheses indicates multiple instructions. For example, "+(_U)" indicates two instructions: + and +_U.

Instruction symbol	Meaning
Instruction symbol followed by "(P)"	This instruction is executed only on the rising edge (off to on).
Instruction symbol followed by "(_U)"	This instruction handles 16-bit or 32-bit unsigned binary data.

❷ Description formats of structured text language

Execution condition is input to EN of each structured text. And, execution result should be described for ENO.

❸ Execution condition (☞ Page 27 Execution Condition)

❹ Description of operands, setting ranges, and data types

- For the data type, refer to the following.

☞ Page 15 Data Specification Method

❺ Devices that can be used as operands

Operand	Bit		Word		Constant
Applicable device[*1]	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H

*1 For details on each device, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)

❻ Control data. Some instructions require control data that determine the operations of the instructions. When control data need to be set by a user, set values according the setting range.

❼ Processing details of the instruction.

❽ Precautions

❾ Error code and error details if the instruction has any possible operation error

- For details, refer to the following.

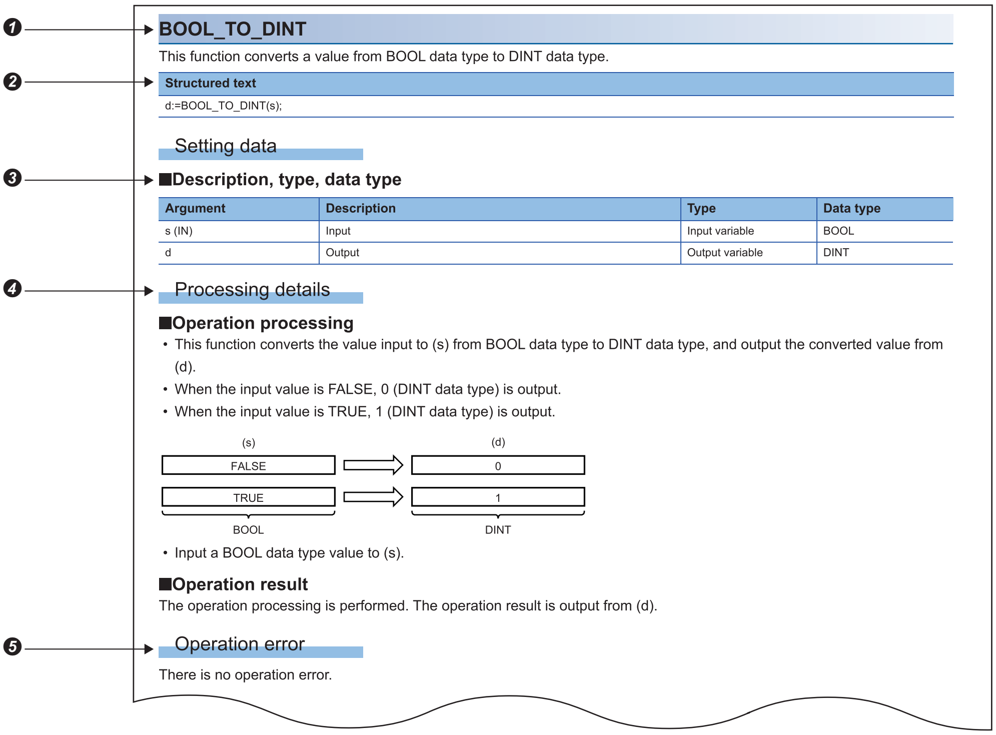
📖MELSEC iQ-R CPU Module User's Manual (Application)

- For the errors not provided here, refer to "List of Error Codes" in the following manual.

📖 MELSEC iQ-R Motion Module User's Manual (Application)

How to read Part 7 and Part 8

The following illustration is for explanation purpose only, and should not be referred to as an actual documentation.



BOOL_TO_DINT

This function converts a value from BOOL data type to DINT data type.

Structured text
d:=BOOL_TO_DINT(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to DINT data type, and output the converted value from (d).
- When the input value is FALSE, 0 (DINT data type) is output.
- When the input value is TRUE, 1 (DINT data type) is output.

- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

❶ Function symbol

❷ Description formats of structured text language

For instances, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

❸ Description of operands, types, and data types

- For the data type, refer to the following.

☞ Page 15 Data Specification Method

❹ Processing details of the standard function or standard function block

❺ Error code and error details if the standard function or standard function block has any possible operation error

- For details, refer to the following.

📖MELSEC iQ-R CPU Module User's Manual (Application)

- For the errors not provided here, refer to "List of Error Codes" in the following manual.

📖 MELSEC iQ-R Motion Module User's Manual (Application)

PART 1 OVERVIEW

This part consists of the following chapter.

1 OVERVIEW

1.1 Instruction Configuration

Many instructions available for motion systems 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. It is set during program creation and cannot be changed during program execution.
Bit device Word device	The user specifies the device where the data to be used in the operation is stored. Necessary data must be thus stored in the specified device before operation execution. 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.

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

Numerical value (n)

For the numerical values of the numbers of devices, transfers, data, and character strings, specify those used by an instruction which uses multiple devices or an instruction which specifies the numbers of repetitions, data to be processed, and character strings.

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

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.

Point

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

1.2 Data Specification Method

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

Data	Classification
Bit data	Bit data
16-bit data (word data)	16-bit signed binary data
	16-bit unsigned binary data
32-bit data (double word data)	32-bit signed binary data
	32-bit unsigned binary data
Real number data (floating-point data)	Single-precision real number data
	Double-precision real number data
Character string data	Character string
	Character string [Unicode]

Device data

The following table lists devices and constants that can be used to specify the setting data of instructions.

Data type	Description	Specifiable device/constant[*1]
Bit	Bit data can be handled.	• Bit device • Bit specification of word device
Word	Word data can be handled.	• Word device • Digit-specified bit device (K1 to K4)[*2] • Decimal constant • Hexadecimal constant
16-bit signed binary	16-bit data can be handled. The value range varies depending on whether the value is signed or unsigned.	
16-bit unsigned binary		
Double word	Double-word data can be handled.	• Word device • Double-word device • Digit-specified bit device (K1 to K8)[*2] • Decimal constant • Hexadecimal constant
32-bit signed binary	Two consecutive sets of 32-bit data or 16-bit data can be handled. The value range varies depending on whether the value is signed or unsigned.	
32-bit unsigned binary		
Single-precision real number	Single-precision real number data (single-precision floating-point data) can be handled.	• Word device • Double-word device • Real constant
Double-precision real number	Double-precision real number data (double-precision floating-point data) can be handled.	• Word device • Double-word device • Real constant
Character string	ASCII code and Shift JIS code character string data can be handled.	• Word device • Character string constant
Character string [Unicode]	Unicode character string data can be handled.	• Word device • Character string constant
Device name	A device can be specified directly.	• Device name corresponding to applicable device

[*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

The following table lists labels that can be used to specify the setting data of instructions.

■Primitive data type

Data type (label)	Specifiable label
Bit (BOOL)	• Bit type label • Bit-specified word [unsigned]/bit string [16 bits] type label • Bit-specified word [signed] type label • Timer/retentive timer/long timer/long retentive timer type label contact/coil • Counter/long counter type label contact/coil
Word [unsigned]/bit string [16 bits] (WORD)	• Word [unsigned]/bit string [16 bits] type label • Current value of timer/retentive timer type label • Current value of counter type label
Double word [unsigned]/bit string [32 bits] (DWORD)	• Double word [unsigned]/bit string [32 bits] type label • Current value of long timer/long retentive timer type label • Current value of long counter type label
Word [signed] (INT)	• Word [signed] type label • Current value of timer/retentive timer type label • Current value of counter type label
Double word [signed] (DINT)	• Double word [signed] type label • Current value of long timer/long retentive timer type label • Current value of long counter type label
Single-precision real number (REAL)	• Single-precision real data type label
Double-precision real number (LREAL)	• Double-precision real data type label
Time (TIME)	• Time type label
Character string (STRING)	• Character string type label
Character string [Unicode] (WSTRING)	• Character string [Unicode] type label

■Generic data type

The generic data type is the data type of the labels which summarize several primitive data types.

Generic data types are used when multiple data types are allowed for arguments and return values of functions or function blocks.

Labels defined in generic data types can be used in any sub-level data type.

Data type (label)							Specifiable data type
ANY*1	ANY_ELEMENTARY	ANY_BIT				ANY_BOOL	Bit
						ANY_BITADDR*1	Bit
						ANY16_U	Word [unsigned]/bit string [16 bits]
						ANY32_U	Double word [unsigned]/bit string [32 bits]
		ANY_WORDADDR	ANY_NUM	ANY_INT	ANY16	ANY16_S	Word [signed]
						ANY16_U	Word [unsigned]/bit string [16 bits]
					ANY32	ANY32_S	Double word [signed], time
						ANY32_U	Double word [unsigned]/bit string [32 bits]
				ANY_REAL		ANYREAL_32	Single-precision real number
						ANYREAL_64	Double-precision real number
			ANY_STRING			ANYSTRING_SINGLE	String
						ANYSTRING_DOUBLE	Character string [Unicode]
			ANY16_OR_STRING_SINGLE			ANY16_S	Word [signed]
						ANY16_U	Word [unsigned]/bit string [16 bits]
						ANYSTRING_SINGLE	String
			ANY_DT				Word [signed], word [unsigned]/bit string [16 bits]
			ANY_TM				Word [signed], word [unsigned]/bit string [16 bits]
	ANY_STRUCT*1						Structures
	STRUCT						Structures

*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 data type
ANYBIT_ARRAY			Bit array
ANYWORD_ARRAY	ANY16_ARRAY	ANY16_S_ARRAY	Word [signed] array
		ANY16_U_ARRAY	Word [unsigned]/bit string [16 bits] array
	ANY32_ARRAY	ANY32_S_ARRAY	Double word [signed] array, time array
		ANY32_U_ARRAY	Double word [unsigned]/bit string [32 bits] array
	ANY_REAL_ARRAY	ANY_REAL_32_ARRAY	Single-precision real number array
		ANY_REAL_64_ARRAY	Double-precision real number array
	ANY_STRING_ARRAY	ANY_STRING_SINGLE_ARRAY	Character string array
		ANY_STRING_DOUBLE_ARRAY	Character string [Unicode] array
STRUCT_ARRAY			Structure array

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

One point of bit device/label can handle 1-bit data.

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.

A bit in a word device can be specified by "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 G0 is specified as G0.5, and bit 10 (b10) of G0 is specified as G0.A.

The following word devices support bit specification.

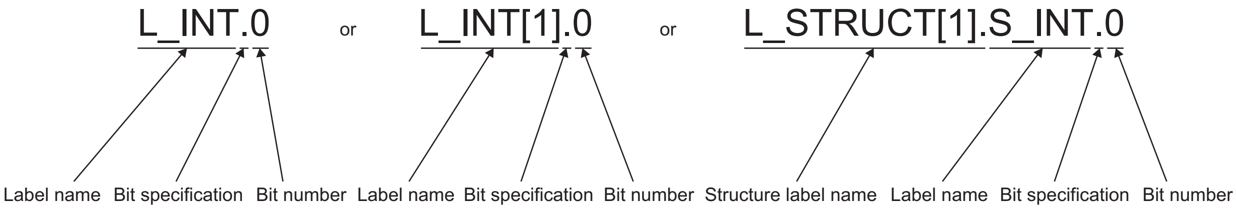
Item	Device
Word devices which support bit specification	• Buffer memory access device (G) • Remote register of link device (RWw, RWr) • Link register of link device (LW) • Link special register (SW)

Handling bit data with word type labels

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

A bit in a word type label can be specified by "Label name.Bit number".

Ex.



The following data types of labels support bit specification.

Item	Data type
Data types of labels which support bit specification.	• Word [signed] (INT type) • Word [unsigned]/bit string [16 bits] (WORD type) • Current value (N) of timer (TIMER type) • Current value (N) of retentive timer (RETENTIVETIMER type) • Current value (N) of counter (COUNTER type)

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 digit specification.

Item	Notation	Example
Bit device	K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 4.)	K4RX10

Precaution

Digit specification cannot be made for a bit type array label.

Digit specification range

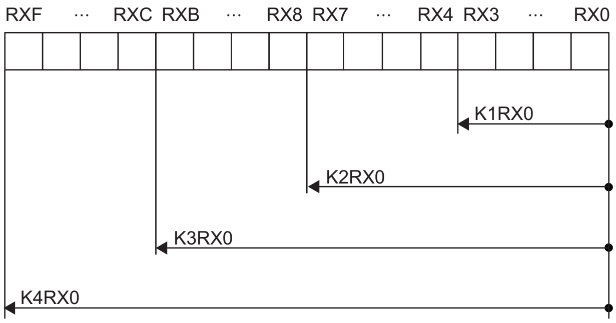
The following table lists the range of 16-bit data for each digit specification.

Digit 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 Unsigned 16-bit data: 0 to 65535	0000H to FFFFH

Ex.

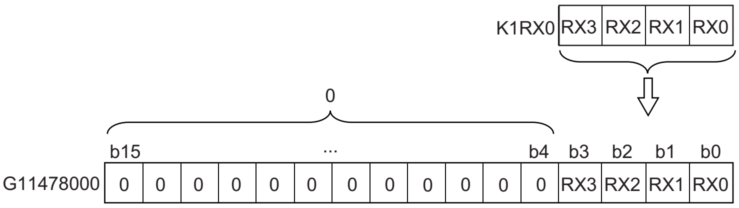
When digit specification is made for RX0, the applicable number of points is as follows.

- K1RX0→4 points from RX0 to RX3
- K2RX0→8 points from RX0 to RX7
- K3RX0→12 points from RX0 to RXB
- K4RX0→16 points from RX0 to RXF



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

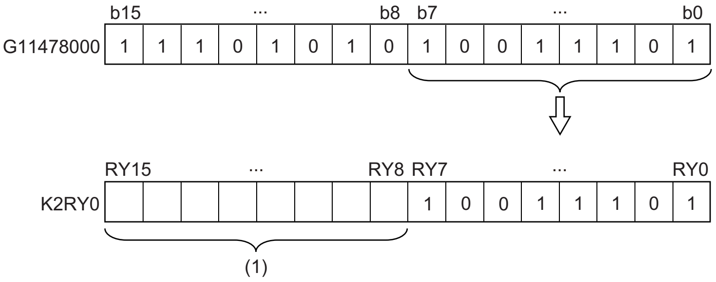
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

Ladder example	Processing
• 16-bit data instruction ENO:=MOV(EN, K1RX0, G11478000);	 K1RX0 RX3 RX2 RX1 RX0 0 b15 … b4 b3 b2 b1 b0 G11478000 0 0 0 0 0 0 0 0 0 0 0 0 RX3 RX2 RX1 RX0

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

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

The upper bit devices than the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device ENO:=MOV(EN, G11478000, K2RY0);	 b15 … b8 b7 … b0 G11478000 1 1 1 0 1 0 1 0 1 0 0 1 1 1 0 1 RY15 … RY8 RY7 … RY0 K2RY0 1 0 0 1 1 1 0 1 (1) (1) The data remain the same.

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 digit specification.

Item	Notation	Example
Bit device	K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 8.)	K8RX80

Precaution

Digit specification cannot be made for a bit type array label.

Digit specification range

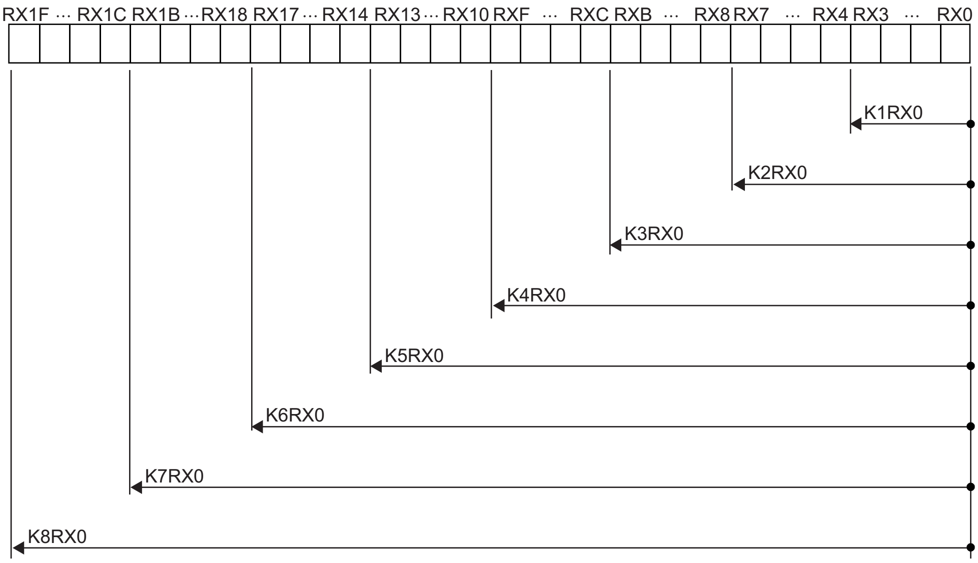
The following table lists the range of 32-bit data for each digit specification.

Digit 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 Unsigned 32-bit data: 0 to 4294967295	00000000H to FFFFFFFFH

Ex.

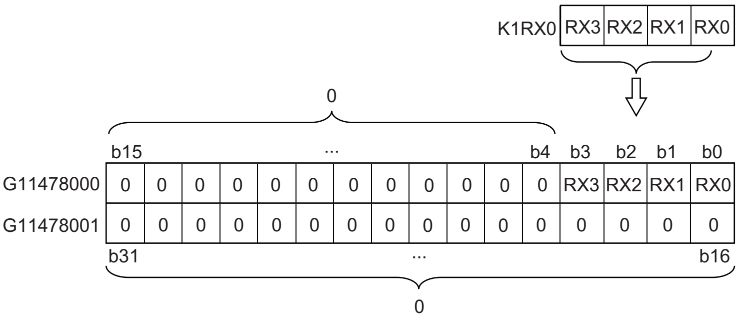
When digit specification is made for RX0, the applicable number of points is as follows.

- K1RX0→4 points from RX0 to RX3
- K2RX0→8 points from RX0 to RX7
- K3RX0→12 points from RX0 to RXB
- K4RX0→16 points from RX0 to RXF
- K5RX0→20 points from RX0 to RX13
- K6RX0→24 points from RX0 to RX17
- K7RX0→28 points from RX0 to RX1B
- K8RX0→32 points from RX0 to RX1F


RX1F ··· RX1C RX1B ··· RX18 RX17 ··· RX14 RX13 ··· RX10 RXF ··· RXC RXB ··· RX8 RX7 ··· RX4 RX3 ··· RX0
K1RX0
K2RX0
K3RX0
K4RX0
K5RX0
K6RX0
K7RX0
K8RX0

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

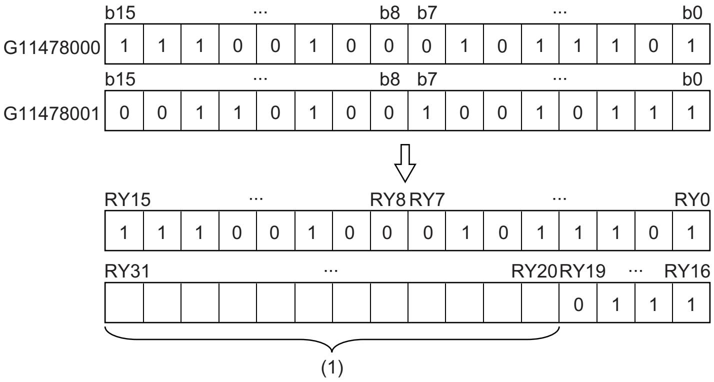
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

Ladder example	Processing
• 32-bit data instruction ENO:=DMOV(EN, K1RX0, G11478000);	 K1RX0 RX3 RX2 RX1 RX0 0 b15 ··· b4 b3 b2 b1 b0 G11478000 0 0 0 0 0 0 0 0 0 0 0 0 RX3 RX2 RX1 RX0 G11478001 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 b31 ··· b16 0

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

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

The upper bit devices than the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device ENO:=DMOV(EN, G11478000, K5RY0);	 b15 ··· b8 b7 ··· b0 G11478000 1 1 1 0 0 1 0 0 0 1 0 1 1 1 0 1 b15 ··· b8 b7 ··· b0 G11478001 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 RY15 ··· RY8 RY7 ··· RY0 1 1 1 0 0 1 0 0 0 1 0 1 1 1 0 1 RY31 ··· RY20 RY19 ··· RY16 0 1 1 1 (1) (1) The data remain the same.

Handling 32-bit data with word devices/labels

■Word device

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

■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 and double-precision 64-bit real number data.

Real number data can be stored only in devices other than bit devices or in single-precision or double-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}≤real number<2^{128}
	Zero		0
	Negative number		-2^{128}<real number≤-2^{-126}
Double-precision real number data (double-precision floating-point data)	Positive number	64 bits (4 word)	2^{-1022}≤real number<2^{1024}
	Zero		0
	Negative number		-2^{1024}<real number≤-2^{-1022}

Configuration of single-precision real number data

The configuration of single-precision real number data compliances with the IEEE754 format.

Configuration of double-precision real number data

The configuration of double-precision real number data compliances with the IEEE754 format.

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.

Set an input value within the following range. If the set value is out of the following range, a conversion error occurs.

Decimal point expression: 0.0000000001 ≤ Absolute value of real number data ≤ 999999900000.0

Exponential notation: 1.175494351E-38 ≤ Absolute value of real number data ≤ 3.402823466E+38

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

The number of significant digits is about 15 because the engineering tool processes double precision real number data in 64-bit double precision.

When the input value of double-precision real number data exceeds 15 digits, the 16th 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 "2147483646.12345678" is set as an input value, it is handled as "2147483646.12346" because 16th digit "6" is rounded off.

Ex.

When "E1.7976931348623157+307" is set as an input value, it is handled as "E1.79769313486232+307" because 16th digit "5" is rounded off.

Set an input value within the following range. If the set value is out of the following range, a conversion error occurs.

Decimal point expression: 0.00000000000000000001 ≤ Absolute value of real number data ≤ 9999999999999999000000.0

Exponential notation: 2.22507385850721E-308 ≤ Absolute value of real number data ≤ 1.79769313486231E+308

Point

The monitor function of the engineering tool can monitor real number data of motion systems.

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
- Double-precision real number data: b0 to b63

The setting range of real number data is as follows. For the operations to be performed when an overflow or underflow occurs or when a special value is input, refer to the following.

- Single-precision real number data: -2^{128}<[single-precision real number data]≤-2^{-126}, 0, 2^{-126}≤[single-precision real number data]<2^{128}
- Double-precision real number data: -2^{1024}<[double-precision real number data]≤-2^{-1022}, 0, 2^{-1022}≤[double-precision real number data]<2^{1024}

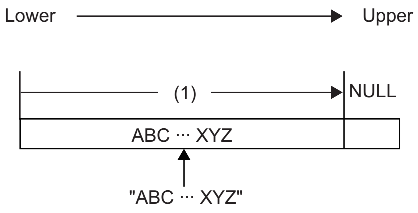
Character string data

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, Shift JIS code	NULL(00H)
Character string [Unicode]	Unicode (UTF-16 (little endian))	NULL(0000H)

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



(1) Character code string

Notation of character string

The following shows the notation of character strings in ST programs.

Data type		Notation	Example
String	STRING	Enclose a string (ASCII code, Shift JIS code) in single quotation marks (').	'ABC'
Character string [Unicode]	WSTRING	Enclose a character string [Unicode] in double quotation marks (").	"ABC"

Data range

The following table summarizes the ranges of character string data.

Type	Maximum number of characters that can be set in a label	Maximum number of characters that can be used for character string constant
Character string	255 single-byte characters (excluding the last NULL character)	255 single-byte characters (excluding the last NULL character)
Character string [Unicode][*1]	255 characters (excluding the last NULL character)	255 characters (excluding the last NULL character)

[*1] For the character string [Unicode], characters up to the basic multilingual plane can be used.

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	Number of words required for storing character strings [Unicode]
0 byte	1 [word]	1 [word]
Odd number of bytes	(Number of character string bytes+1) ÷ 2 [words]	— (because one character is an even number of bytes)
Even number of bytes	(Number of character string bytes÷2) +1 [words]	Number of characters+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 G0	Image of storing character string data from word type label array arrayA[0]
Null character string (("") or (''))	G0 NULL NULL	arrayA[0] NULL NULL
ABC	G0 B A G1 NULL C	arrayA[0] B A arrayA[1] NULL C
ABCD	G0 B A G1 D C G2 NULL NULL	arrayA[0] B A arrayA[1] D C arrayA[2] NULL NULL

■Character strings [Unicode]

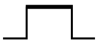
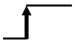
In each character string [Unicode] storage image, "NULL" indicates a NULL code (0000H).

Character string to be stored	Image of storing character string data from G0	Image of storing character string data from word type label array arrayA[0]
Null character string ("")	G0 NULL	arrayA[0] NULL
ABCD	G0 A G1 B G2 C G3 D G4 NULL	arrayA[0] A arrayA[1] B arrayA[2] C arrayA[3] D arrayA[4] NULL

1.3 Execution Condition

Types of execution conditions

The following table lists the execution conditions of instructions.

Execution condition		Description[*1]
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.
Every scan	—	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.

[*1] When the program is described in structured text language (ST), EN will be the precondition of the instruction.

Execution condition of each instruction

The execution condition varies depending on the instruction. For execution condition, refer to the details of each instruction in this manual.

When the program is described in structured text language (ST), EN will be the execution condition. The instruction is executed only when EN is TRUE. The status of ENO will be the same as that of EN.

Note that the execution condition of standard functions and function blocks differs depending on the existence of EN. If there is no EN, the standard function or function block is executed at every scan. For the execution condition of the standard function or function block with EN, refer to the details of each standard function or function block in this manual.

1.4 Precautions on Programming

Errors common to instructions

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

Error content	Error code
• The device or label specified by the instruction exceeds the available range. • The number of array elements is not enough.	3506H
• The device or label area used in the instruction exceeded the specified range. • An array is not selected.	3510H

For details, refer to "List of Error Codes" in the following manual.

📖MELSEC iQ-R Motion Module User's Manual (Application)

Timer, long timer, and long retentive timer type labels

The timer, long timer, and long retentive timer type labels are structures whose members are S (contact), C (coil), and N (current value). When the data to be handled exceeds the width (32 bits) of the current value, these operate by using not only the area of the current value but also the areas of the previous value, contact, and coil.

Data type	Member
Timer (T)	S (Contact): BOOL C (Coil): BOOL N (Current value): WORD
Retentive timer (ST)	
Counter (C)	
Long timer (LT)	S (Contact): BOOL C (Coil): BOOL N (Current value): DWORD
Long retentive timer (LST)	
Long counter (LC)	

Operations arising when the OUT and SET/RST instructions of the same device are used

This section describes the operation when two or more OUT and SET/RST instructions that use the same device are executed within one scan.

For OUT instructions of the same device

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.

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.

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

This part consists of the following chapters.

2 MOTION SYSTEM INSTRUCTIONS

The following table summarizes how to read the instruction lists.

Item	Description
Instruction symbol	An instruction name
Processing details	An overview of the instruction
Reference	Section where detailed information is described

2.1 Sequence Instructions

Output instructions

■Out (excluding the timer and counter)

Instruction symbol	Processing details	Reference
OUT	Outputs the operation result to the specified device.	Page 46 OUT

■Timer, long timer

Instruction symbol	Processing details	Reference
OUT_T	Starts time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state). • OUT_T: Low-speed timer instruction • OUTH_T: High-speed timer instruction • OUT_ST: Low-speed retentive timer instruction • OUTH_ST: High-speed retentive timer instruction • OUT_LT: Low-speed long timer instruction • OUT_LST: Low-speed long retentive timer instruction	Page 47 OUT_T, OUTH_T, OUT_ST, OUTH_ST
OUTH_T		
OUT_ST		
OUTH_ST		
OUT_LT		Page 50 OUT_LT, OUT_LST
OUT_LST		

■Counter, long counter

Instruction symbol	Processing details	Reference
OUT_C	Increments the current counter value (count value) by one when the operation result up to the OUT instruction turns on. When the count value reaches the set value, the normally open contact of the counter turns on (continuity state) and the normally closed contact turns off (non-continuity state). • OUT_C: Counter • OUT_LC: Long counter	Page 53 OUT_C
OUT_LC		Page 54 OUT_LC

■Setting devices

Instruction symbol	Processing details	Reference
SET	Turns on the specified bit.	Page 55 SET

■Resetting devices

Instruction symbol	Processing details	Reference
RST	Turns off the specified device.	Page 57 RST

2.2 Basic Instructions

Arithmetic operation instructions

■Adding/subtracting 16-bit binary data

Instruction symbol	Processing details	Reference
+	Adds the two sets of 16-bit binary data specified.	Page 60 +(_U)
+_U		
-	Performs subtraction between the two sets of 16-bit binary data specified.	Page 62 -(_U)
-_U		

■Adding/subtracting 32-bit binary data

Instruction symbol	Processing details	Reference
D+	Adds the two sets of 32-bit binary data specified.	Page 64 D+(_U)
D+_U		
D-	Performs subtraction between the two sets of 32-bit binary data specified.	Page 66 D-(_U)
D-_U		

■Multiplying/dividing 16-bit binary data

Instruction symbol	Processing details	Reference
*	Multiplies the two sets of 16-bit binary data specified.	Page 68 *(_U)
*_U		
/	Performs division between the two sets of 16-bit binary data specified.	Page 69 /(_U)
/_U		

■Multiplying/dividing 32-bit binary data

Instruction symbol	Processing details	Reference
D*	Multiplies the two sets of 32-bit binary data specified.	Page 70 D*(_U)
D*_U		
D/	Performs division between the two sets of 32-bit binary data specified.	Page 71 D/(_U)
D/_U		

■Incrementing/decrementing 16-bit binary data

Instruction symbol	Processing details	Reference
INC	Increments the specified 16-bit binary data by one.	Page 72 INC(_U)
INC_U		
DEC	Decrements the specified 16-bit binary data by one.	Page 73 DEC(_U)
DEC_U		

■Incrementing/decrementing 32-bit binary data

Instruction symbol	Processing details	Reference
DINC	Increments the specified 32-bit binary data by one.	Page 74 DINC(_U)
DINC_U		
DDEC	Decrements the specified 32-bit binary data by one.	Page 75 DDEC(_U)
DDEC_U		

Logical operation instructions

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

Instruction symbol	Processing details	Reference
WAND	Performs an AND operation on the two sets of 16-bit binary data specified.	Page 76 WAND
DAND	Performs an AND operation on the two sets of 32-bit binary data specified.	Page 77 DAND

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

Instruction symbol	Processing details	Reference
WOR	Performs an OR operation on the two sets of 16-bit binary data specified.	Page 78 WOR
DOR	Performs an OR operation on the two sets of 32-bit binary data specified.	Page 79 DOR

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

Instruction symbol	Processing details	Reference
WXOR	Performs an XOR operation on the two sets of 16-bit binary data specified.	Page 80 WXOR
DXOR	Performs an XOR operation on the two sets of 32-bit binary data specified.	Page 81 DXOR

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

Instruction symbol	Processing details	Reference
WXNR	Performs an XNOR operation on the two sets of 16-bit binary data specified.	Page 82 WXNR
DXNR	Performs an XNOR operation on the two sets of 32-bit binary data specified.	Page 83 DXNR

Data conversion instructions

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

Instruction symbol	Processing details	Reference
NEG	Inverts the sign of 16-bit binary device. $\underset{\uparrow\text{BIN}}{\overline{(d)}} \longrightarrow (d)$	Page 84 NEG
DNEG	Inverts the sign of 32-bit binary device. $\underset{\uparrow\text{BIN}}{\overline{(d)+1,\ (d)}} \longrightarrow (d)+1,\ (d)$	Page 85 DNEG

Data transfer instructions

■Transferring 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MOV	Transfers the 16-bit binary data in the device specified. $(s) \longrightarrow (d)$	Page 86 MOV
DMOV	Transfers the 32-bit binary data in the device specified. $(s)+1,\ (s) \longrightarrow (d)+1,\ (d)$	Page 87 DMOV

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

Instruction symbol	Processing details	Reference
CML	Inverts the specified 16-bit binary data bit by bit, and transfers the inverted data. $\overline{(s)} \longrightarrow (d)$	Page 88 CML
DCML	Inverts the specified 32-bit binary data bit by bit, and transfers the inverted data. $\overline{(s)+1,\ (s)} \longrightarrow (d)+1,\ (d)$	Page 89 DCML

■Inverting and transferring 1-bit data

Instruction symbol	Processing details	Reference
CMLB	Inverts the bit data in the device specified by (s), and stores the inverted data in the device specified by (d).	Page 90 CMLB

■Transferring 1-bit data

Instruction symbol	Processing details	Reference
MOVB	Stores the bit data in the device specified by (s) in the device specified by (d).	Page 91 MOVB

2.3 Application Instructions

Program control

Program execution control instructions

■Disabling/enabling interrupt programs

Instruction symbol	Processing details	Reference
DI	Disables the execution of fixed scan execution type programs.	Page 94 DI, EI
EI	Clears the fixed scan execution type programs execution disabled state.	

Program control instructions

■Changing the program execution type to standby type

Instruction symbol	Processing details	Reference
PSTOP	Changes the type of the specified program to standby type.	Page 95 PSTOP

■Changing the program execution type to scan execution type

Instruction symbol	Processing details	Reference
PSCAN	Changes the type of the specified program to normal execution type.	Page 96 PSCAN

Data processing

Data processing instructions

■Adding 16-bit binary data

Instruction symbol	Processing details	Reference
WSUM	Adds the (n) points of 16-bit binary data in the device starting from the one specified by (s), and stores the result in the device specified by (d).	Page 97 WSUM(_U)
WSUM_U		

■Adding 32-bit binary data

Instruction symbol	Processing details	Reference
DWSUM	Adds the (n) points of 32-bit binary data in the device starting from the one specified by (s), and stores the result in the device specified by (d).	Page 98 DWSUM(_U)
DWSUM_U		

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

Instruction symbol	Processing details	Reference
MEAN	Calculates the average value of the (n) points of 16-bit data in the device starting from the one specified by (s), and stores the average value in the device specified by (d).	Page 99 MEAN(_U)
MEAN_U		
DMEAN	Calculates the average value of the (n) points of 32-bit data in the device starting from the one specified by (s), and stores the average value in the device specified by (d).	Page 100 DMEAN(_U)
DMEAN_U		

■Calculating the square root of 32-bit binary data

Instruction symbol	Processing details	Reference
DSQRT	Performs a square root operation of the specified 32-bit binary data.[*1] $\sqrt{(s)+1}$, (s)→(d)	Page 101 DSQRT

*1 When calculating the square root except 32-bit binary data, use SQRT of standard functions.
☞ Page 159 SQRT

String processing

String processing instructions

■Transferring string data

Instruction symbol	Processing details	Reference
$MOV	Transfers the character strings in the device specified by (s) to the device specified by (d) and later.	Page 102 $MOV
$MOV_WS	Transfers the character strings [Unicode] in the device specified by (s) to the device specified by (d) and later.	Page 104 $MOV_WS

Real value processing

Floating-point instruction

■Adding/subtracting single-precision real numbers

Instruction symbol	Processing details	Reference
E+	Adds single-precision real numbers.	Page 105 E+
E-	Performs subtraction between single-precision real numbers.	Page 106 E-

■Adding/subtracting double-precision real numbers

Instruction symbol	Processing details	Reference
ED+	Adds double-precision real numbers.	Page 107 ED+
ED-	Performs subtraction between double-precision real numbers.	Page 108 ED-

■Multiplying/dividing single-precision real numbers

Instruction symbol	Processing details	Reference
E*	Multiplies single-precision real numbers.	Page 109 E*
E/	Performs division between single-precision real numbers.	Page 110 E/

■Multiplying/dividing double-precision real numbers

Instruction symbol	Processing details	Reference
ED*	Multiplies double-precision real numbers.	Page 111 ED*
ED/	Performs division between double-precision real numbers.	Page 112 ED/

■Inverting the sign of single-precision real number

Instruction symbol	Processing details	Reference
ENEG	Inverts the sign of single-precision real number data. (d)+1, (d) → (d)+1, (d) (1) (1) Real number	Page 113 ENEG

■Inverting the sign of double-precision real number

Instruction symbol	Processing details	Reference
EDNEG	Inverts the sign of double-precision real number data. (d)+3, (d)+2, (d)+1, (d) → (d)+3, (d)+2, (d)+1, (d) (1) (1) Real number	Page 114 EDNEG

■Transferring single-precision real number

Instruction symbol	Processing details	Reference
EMOV	Transfers single-precision real number data to the specified device. (s)+1, (s) → (d)+1, (d) (1) (1) Real number	Page 115 EMOV

■Transferring double-precision real number

Instruction symbol	Processing details	Reference
EDMOV	Transfers double-precision real number data to the specified device. $\underbrace{(s)+3, (s)+2, (s)+1, (s)}_{(1)} \longrightarrow (d)+3, (d)+2, (d)+1, (d)$ (1) Real number	Page 116 EDMOV

3 STANDARD FUNCTIONS/FUNCTION BLOCKS

How to read the list is shown below.

Item	Description
Function symbol and function block symbol	A function and function block name are shown.
Processing details	An overview of the functions and function blocks is explained.
Reference	Indicates the reference of detailed information.

3.1 Standard Functions

Type conversion functions

■Converting BOOL to WORD/DWORD

Function symbol	Processing details	Reference
BOOL_TO_WORD	Converts a value from BOOL data type to WORD data type.	Page 118 BOOL_TO_WORD
BOOL_TO_DWORD	Converts a value from BOOL data type to DWORD data type.	Page 119 BOOL_TO_DWORD

■Converting BOOL to INT/DINT

Function symbol	Processing details	Reference
BOOL_TO_INT	Converts a value from BOOL data type to INT data type.	Page 120 BOOL_TO_INT
BOOL_TO_DINT	Converts a value from BOOL data type to DINT data type.	Page 121 BOOL_TO_DINT

■Converting BOOL to TIME

Function symbol	Processing details	Reference
BOOL_TO_TIME	Converts a value from BOOL data type to TIME data type.	Page 122 BOOL_TO_TIME

■Converting WORD to BOOL

Function symbol	Processing details	Reference
WORD_TO_BOOL	Converts a value from WORD data type to BOOL data type.	Page 123 WORD_TO_BOOL

■Converting WORD to DWORD

Function symbol	Processing details	Reference
WORD_TO_DWORD	Converts a value from WORD data type to DWORD data type.	Page 124 WORD_TO_DWORD

■Converting WORD to INT/DINT

Function symbol	Processing details	Reference
WORD_TO_INT	Converts a value from WORD data type to INT data type.	Page 125 WORD_TO_INT
WORD_TO_DINT	Converts a value from WORD data type to DINT data type.	Page 126 WORD_TO_DINT

■Converting WORD to TIME

Function symbol	Processing details	Reference
WORD_TO_TIME	Converts a value from WORD data type to TIME data type.	Page 127 WORD_TO_TIME

■Converting DWORD to BOOL

Function symbol	Processing details	Reference
DWORD_TO_BOOL	Converts a value from DWORD data type to BOOL data type.	Page 128 DWORD_TO_BOOL

■Converting DWORD to WORD

Function symbol	Processing details	Reference
DWORD_TO_WORD	Converts a value from DWORD data type to WORD data type.	Page 129 DWORD_TO_WORD

■Converting DWORD to INT/DINT

Function symbol	Processing details	Reference
DWORD_TO_INT	Converts a value from DWORD data type to INT data type.	Page 130 DWORD_TO_INT
DWORD_TO_DINT	Converts a value from DWORD data type to DINT data type.	Page 131 DWORD_TO_DINT

■Converting DWORD to TIME

Function symbol	Processing details	Reference
DWORD_TO_TIME	Converts a value from DWORD data type to TIME data type.	Page 132 DWORD_TO_TIME

■Converting INT to BOOL

Function symbol	Processing details	Reference
INT_TO_BOOL	Converts a value from INT data type to BOOL data type.	Page 133 INT_TO_BOOL

■Converting INT to WORD/DWORD

Function symbol	Processing details	Reference
INT_TO_WORD	Converts a value from INT data type to WORD data type.	Page 134 INT_TO_WORD
INT_TO_DWORD	Converts a value from INT data type to DWORD data type.	Page 135 INT_TO_DWORD

■Converting INT to DINT

Function symbol	Processing details	Reference
INT_TO_DINT	Converts a value from INT data type to DINT data type.	Page 136 INT_TO_DINT

■Converting INT to REAL/LREAL

Function symbol	Processing details	Reference
INT_TO_REAL	Converts a value from INT data type to REAL data type.	Page 137 INT_TO_REAL
INT_TO_LREAL	Converts a value from INT data type to LREAL data type.	Page 138 INT_TO_LREAL

■Converting INT to TIME

Function symbol	Processing details	Reference
INT_TO_TIME	Converts a value from INT data type to TIME data type.	Page 139 INT_TO_TIME

■Converting DINT to BOOL

Function symbol	Processing details	Reference
DINT_TO_BOOL	Converts a value from DINT data type to BOOL data type.	Page 140 DINT_TO_BOOL

■Converting DINT to WORD/DWORD

Function symbol	Processing details	Reference
DINT_TO_WORD	Converts a value from DINT data type to WORD data type.	Page 141 DINT_TO_WORD
DINT_TO_DWORD	Converts a value from DINT data type to DWORD data type.	Page 142 DINT_TO_DWORD

■Converting DINT to INT

Function symbol	Processing details	Reference
DINT_TO_INT	Converts a value from DINT data type to INT data type.	Page 143 DINT_TO_INT

■Converting DINT to REAL/LREAL

Function symbol	Processing details	Reference
DINT_TO_REAL	Converts a value from DINT data type to REAL data type.	Page 144 DINT_TO_REAL
DINT_TO_LREAL	Converts a value from DINT data type to LREAL data type.	Page 145 DINT_TO_LREAL

■Converting DINT to TIME

Function symbol	Processing details	Reference
DINT_TO_TIME	Converts a value from DINT data type to TIME data type.	Page 146 DINT_TO_TIME

■Converting REAL to INT/DINT

Function symbol	Processing details	Reference
REAL_TO_INT	Converts a value from REAL data type to INT data type.	Page 147 REAL_TO_INT
REAL_TO_DINT	Converts a value from REAL data type to DINT data type.	Page 148 REAL_TO_DINT

■Converting REAL to LREAL

Function symbol	Processing details	Reference
REAL_TO_LREAL	Converts a value from REAL data type to LREAL data type.	Page 149 REAL_TO_LREAL

■Converting LREAL to INT/DINT

Function symbol	Processing details	Reference
LREAL_TO_INT	Converts a value from LREAL data type to INT data type.	Page 150 LREAL_TO_INT
LREAL_TO_DINT	Converts a value from LREAL data type to DINT data type.	Page 151 LREAL_TO_DINT

■Converting LREAL to REAL

Function symbol	Processing details	Reference
LREAL_TO_REAL	Converts a value from LREAL data type to REAL data type.	Page 152 LREAL_TO_REAL

■Converting TIME to BOOL

Function symbol	Processing details	Reference
TIME_TO_BOOL	Converts a value from TIME data type to BOOL data type.	Page 153 TIME_TO_BOOL

■Converting TIME to WORD/DWORD

Function symbol	Processing details	Reference
TIME_TO_WORD	Converts a value from TIME data type to WORD data type.	Page 154 TIME_TO_WORD
TIME_TO_DWORD	Converts a value from TIME data type to DWORD data type.	Page 155 TIME_TO_DWORD

■Converting TIME to INT/DINT

Function symbol	Processing details	Reference
TIME_TO_INT	Converts a value from TIME data type to INT data type.	Page 156 TIME_TO_INT
TIME_TO_DINT	Converts a value from TIME data type to DINT data type.	Page 157 TIME_TO_DINT

Single variable functions

■Calculating the absolute value

Function symbol	Processing details	Reference
ABS	Outputs the absolute value of an input value.	Page 158 ABS

■Calculating the square root

Function symbol	Processing details	Reference
SQRT	Calculates the square root of an input value.	Page 159 SQRT

■Calculating the natural logarithm

Function symbol	Processing details	Reference
LN	Outputs the natural logarithm (logarithm with base e) of an input value.	Page 160 LN

■Calculating the common logarithm

Function symbol	Processing details	Reference
LOG	Outputs the common logarithm (logarithm with base 10) of an input value.	Page 161 LOG

■Calculating the exponent

Function symbol	Processing details	Reference
EXP	Outputs the exponent of an input value.	Page 162 EXP

■Calculating the sine/cosine/tangent

Function symbol	Processing details	Reference
SIN	Outputs the sine of an input value.	Page 163 SIN
COS	Outputs the cosine of an input value.	Page 164 COS
TAN	Outputs the tangent of an input value.	Page 165 TAN

■Calculating the arc sine/arc cosine/arc tangent

Function symbol	Processing details	Reference
ASIN	Outputs the arc sine (SIN^{-1}) of an input value.	Page 166 ASIN
ACOS	Outputs the arc cosine (COS^{-1}) of an input value.	Page 167 ACOS
ATAN	Outputs the arc tangent (TAN^{-1}) of an input value.	Page 168 ATAN

Arithmetic operation functions

■Addition

Function symbol	Processing details	Reference
ADD	Outputs the sum of input values ((s1)+(s2)+···+(s28)).	Page 169 ADD

■Multiplication

Function symbol	Processing details	Reference
MUL	Outputs the product of input values ((s1)×(s2)×···×(s28)).	Page 171 MUL

■Subtraction

Function symbol	Processing details	Reference
SUB	Outputs the difference between input values ((s1)-(s2)).	Page 173 SUB

■Division

Function symbol	Processing details	Reference
DIV	Outputs the quotient of input values ((s1)÷(s2)).	Page 175 DIV

■Remainder

Function symbol	Processing details	Reference
MOD	Outputs the remainder of input values ((s1)÷(s2)).	Page 177 MOD

■Assignment (move operation)

Function symbol	Processing details	Reference
MOVE	Outputs the assignment value of an input value.	Page 178 MOVE

Boolean function

■NOT operation

Function symbol	Processing details	Reference
NOT	Outputs the logical NOT of input values.	Page 179 NOT

Selection functions

■Selecting the maximum/minimum value

Function symbol	Processing details	Reference
MAX	Outputs the maximum input value.	Page 180 MAX, MIN
MIN	Outputs the minimum input value.	

3.2 Standard Function Blocks

Bistable function blocks

■Bistable function block (set-dominant)

Function block symbol	Processing details	Reference
SR	Discriminates between two input values, and outputs1 (TRUE) or 0 (FALSE).	Page 184 SR

■Bistable function block (reset-dominant)

Function block symbol	Processing details	Reference
RS	Discriminates between two input values, and outputs1 (TRUE) or 0 (FALSE).	Page 185 RS

Edge detection function blocks

■Detecting a rising edge

Function block symbol	Processing details	Reference
R_TRIG	Detects a signal rising edge, and outputs the pulse signal.	Page 187 R_TRIG

■Detecting a falling edge

Function block symbol	Processing details	Reference
F_TRIG	Detects a signal falling edge, and outputs the pulse signal.	Page 188 F_TRIG

Timer function blocks

■Pulse timer

Function block symbol	Processing details	Reference
TP	Keeps the signal on for the specified period of time.	Page 189 TP

■On delay timer

Function block symbol	Processing details	Reference
TON	Turns on a signal after the specified period of time.	Page 191 TON

■Off delay timer

Function block symbol	Processing details	Reference
TOF	Turns off a signal after the specified period of time.	Page 193 TOF

■Timer function block

Function block symbol	Processing details	Reference
TIMER_10_FB_M	Starts counting a timer when the execution condition is satisfied, and continues counting until the timer reaches the set value.	Page 195 TIMER_□_M
TIMER_100_FB_M		
TIMER_HIGH_FB_M		
TIMER_LOW_FB_M		
TIMER_CONT_FB_M		
TIMER_CONTHFB_M		

MEMO

4 MOTION DEDICATED INSTRUCTIONS

User Function Execution Instruction

Instruction symbol	Processing details	Reference
G(P).CEXECUTE	Instructs the execution of processing in the Motion module.	Page 203 G(P).CEXECUTE

MEMO

PART 3 SEQUENCE INSTRUCTIONS

This part consists of the following chapters.

5 SEQUENCE INSTRUCTIONS

5.1 Output Instructions

Out (excluding the timer and counter)

OUT

This instruction outputs the operation result to the specified device.

ST
ENO:=OUT(EN,d);

■Execution condition

Instruction	Execution condition
OUT	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(d)	On/off target device number	—	ANY_BOOL
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

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

Condition	Operation result	Coil/Specified bit
When a bit device is used	Off	Off
	On	On
When a bit-specified word device is used	Off	0
	On	1

Operation error

There is no operation error.

Timer

OUT_T, OUTH_T, OUT_ST, OUTH_ST

- OUT_T: Low-speed timer instruction
- OUTH_T: High-speed timer instruction
- OUT_ST: Low-speed retentive timer instruction
- OUTH_ST: High-speed retentive timer instruction

These instructions start time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).

ST

ENO:=OUT_T(EN,Coil,Value);
ENO:=OUTH(EN,Coil,Value);

■Execution condition

Instruction	Execution condition
OUT_T OUTH_T OUT_ST OUTH_ST	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type
Coil	Timer type label	—	ANY_BOOL
Value	Value set for the timer	0 to 32767	ANY_INT
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
Coil	—	—	○	—	—
Value	—	—	○	○	○[*1]

[*1] Only K (decimal constant) can be used.

Processing details

- These instructions start time measurement, triggered by the coil specified by Coil, when the operation result up to the OUT instruction is on. When time is up (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the operation result up to the OUT instruction turns off, the contact responds as shown below.

Type	Timer coil	Current value	Before time is up		After time is up	
			Normally open contact	**Normally closed contact**	**Normally open contact**	**Normally closed contact**
Low-speed timer	Off	0	Non-continuity	Continuity	Non-continuity	Continuity
High-speed timer						
Low-speed retentive timer	Off	Current value retained	Non-continuity	Continuity	Continuity	Non-continuity
High-speed retentive timer						

- When the timer set value is 0, the time will be up at execution of the OUT instruction.
- The following operations are performed at execution of the OUT instruction.
  - The coil used as a trigger of the OUT_T, OUTH_T, OUT_ST, or OUTH_ST instruction turns on or off.
  - The contact used as a trigger of the OUT_T, OUTH_T, OUT_ST, or OUTH_ST instruction turns on or off.
  - The current value of the OUT_T, OUTH_T, OUT_ST, or OUTH_ST instruction is changed.
- If the same OUT_T, OUTH_T, OUT_ST, or OUTH_ST instruction is executed two times or more in a single scan, the current value is updated by the number of times the instruction is executed.

Point

- The timer limit value is set in parameter using the engineering tool.

Low-speed timer/low-speed retentive timer: 1 to 10000 ms (in increments of 1 ms) (Default: 100 ms)

High-speed timer/high-speed retentive timer: 1 to 10000 μs (in increments of 1 μs) (Default: 500 μs)

- For the counting method, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

- Even if the same time is specified, the counting result may not match for the timer and the long timer as the counting method differs.

Precautions

To create a program in which the operation of a timer contact triggers the operation of another timer, program the timers in order from the one that operates last.

In the following cases, if a program is created in order of timer measurements, all timers turn on in the same scan.

- The set value is smaller than the scan time.
- The set value is 1.

Ex.

When timers Time_0 to Time_2 are programmed in order from the one that measures last

[Label definitions]

Label name	Data type	Class
Time_0	Timer	VAR
Time_1	Timer	VAR
Time_2	Timer	VAR
Flag_Label	Bit	VAR
ENO_Label	Bit	VAR

[Program]

```
ENO_Label := OUT_T(Time_1.S,Time_2,1);
ENO_Label := OUT_T(Time_0.S,Time_1,1);
ENO_Label := OUT_T(Flag_Label,Time_0,1);
```

(1) Timer Time_2 starts measurement from the next scan after the contact of timer Time_1 turns on.

(2) Timer Time_1 starts measurement from the next scan after the contact of timer Time_0 turns on.

(3) Timer Time_0 starts measurement when Flag_Label turns on.

Ex.

When timers Time_0 to Time_2 are programmed in order of measurement

[Label definitions]

Label name	Data type	Class
Time_0	Timer	VAR
Time_1	Timer	VAR
Time_2	Timer	VAR
Flag_Label	Bit	VAR
ENO_Label	Bit	VAR

[Program]

```
ENO_Label := OUT_T(Flag_Label,Time_0,1);
ENO_Label := OUT_T(Time_0.S,Time_1,1);
ENO_Label := OUT_T(Time_1.S,Time_2,1);
```

(1) Timer Time_0 starts measurement when Flag_Label turns on.

(2) When the contact of timer Time_0 turns on, the contacts of timers Time_1 and Time_2 also turn on.

Operation error

There is no operation error.

Precautions

This section describes the precautions when using the timer.

■Precautions about timer usage

- Do not describe more than one coil (the timer instruction) on the same timer during a single scanning. Doing so results in improper measurement because the timer current value is updated when the coil for each timer is executed.
- When timer is not used for data collection for each scan, proper measurement is impossible.
- The timer cannot be used in the initial execution type program and the fixed scan execution type program.
- Even when the setting value is increased after the timer time is up, the timer status does not change (time continues to be up) and the timer does not operate.
- Do not set the timer setting value to 32768 or above. If used when set to 32768 or above, the timer contact may not turn on.

Long timer

OUT_LT, OUT_LST

- OUT_LT: Low-speed long timer instruction
- OUT_LST: Low-speed long retentive timer instruction

These instructions start time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).

ST
ENO:=OUT_T(EN,Coil,Value);

■Execution condition

Instruction	Execution condition
OUT_LT OUT_LST	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type
Coil	Long timer type label	—	ANY_BOOL
Value	Value set for the long timer	0 to 4294967295	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
Coil	—	—	—	—	—
Value	—	—	○	○	○[*1]

*1 Only K (decimal constant) can be used.

Processing details

- These instructions start time measurement, triggered by the coil specified by Coil, when the operation result up to the OUT instruction is on. When time is up (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the operation result up to the OUT instruction turns off, the contact responds as shown below.

Type	Timer coil	Current value	Before time is up		After time is up	
			Normally open contact	Normally closed contact	Normally open contact	Normally closed contact
Long timer	Off	0	Non-continuity	Continuity	Non-continuity	Continuity
Long retentive timer	Off	Current value retained	Non-continuity	Continuity	Continuity	Non-continuity

- When the timer set value is 0, the time will be up at execution of the OUT instruction.
- The following operations are performed at execution of the OUT instruction.
  - The coil used as a trigger of the OUT_LT or OUT_LST instruction turns on or off.
  - The contact used as a trigger of the OUT_LT or OUT_LST instruction turns on or off.
  - The current value of the OUT_LT or OUT_LST instruction is changed.
- If the same OUT_LT or OUT_LST instruction is executed two times or more in a single scan, the current value is updated by the number of times the instruction is executed.

Point

- The timer limit value is set in parameter using the engineering tool.

Long timer/long retentive timer: 1 to 1000000 μs (in increments of 1 μs) (Default: 500 μs)

- For the counting method, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

- Even if the same time is specified, the counting result may not match for the timer and the long timer as the counting method differs.

Precautions

To create a program in which the operation of a long timer contact triggers the operation of another long timer, program the long timers in order from the one that operates last.

In the following cases, if a program is created in order of timer measurements, all timers turn on in the same scan.

- The set value is smaller than the scan time.
- The set value is 1.

Ex.

When timers LTime_0 to LTime_2 are programmed in order from the one that measures last

[Label definitions]

Label name	Data type	Class
LTime_0	Long timer	VAR
LTime_1	Long timer	VAR
LTime_2	Long timer	VAR
Flag_Label	Bit	VAR
ENO_Label	Bit	VAR

[Program]

```
ENO_Label := OUT_LT(LTime_1.S,LTime_2,1);
ENO_Label := OUT_LT(LTime_0.S,LTime_1,1);
ENO_Label := OUT_LT(Flag_Label,LTime_0,1);
```

(1) Long timer LTime_2 starts measurement from the next scan after the contact of long timer LTime_1 turns on.

(2) Long timer LTime_1 starts measurement from the next scan after the contact of long timer LTime_0 turns on.

(3) Long timer LTime_0 starts measurement when Flag_Label turns on.

Ex.

When long timers LTime_0 to LTime_2 are programmed in order of measurement

[Label definitions]

Label name	Data type	Class
LTime_0	Long timer	VAR
LTime_1	Long timer	VAR
LTime_2	Long timer	VAR
Flag_Label	Bit	VAR
ENO_Label	Bit	VAR

[Program]

```
ENO_Label := OUT_LT(Flag_Label,LTime_0,1);
ENO_Label := OUT_LT(LTime_0.S,LTime_1,1);
ENO_Label := OUT_LT(LTime_1.S,LTime_2,1);
```

(1) Long timer LTime_0 starts measurement when Flag_Label turns on.

(2) When the contact of timer LTime_0 turns on, the contacts of timers LTime_1 and LTime_2 also turn on.

Operation error

There is no operation error.

Precautions

This section describes the precautions when using the long timer.

■Precautions about long timer usage

- The long timer cannot be used in initial execution type programs.
- Even when the setting value is increased after the long timer time is up, the long timer status does not change (time continues to be up) and the long timer does not operate.

Counter

OUT_C

This instruction increments the current counter value (count value) by one when the operation result up to the OUT instruction turns on. When the count value reaches the set value, the normally open contact of the counter turns on (continuity state) and the normally closed contact turns off (non-continuity state).

ST
ENO:=OUT_C(EN,Coil,Value);

■Execution condition

Instruction	Execution condition
OUT_C	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type
Coil	Counter number	—	ANY_BOOL[*1]
Value	Value set for the counter	0 to 65535	ANY_INT
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

*1 Only counter type labels can be used.

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
Coil	—	—	○	—	—
Value	—	—	○	○	○[*1]

*1 Only K (decimal constant) can be used.

Processing details

- This instruction increments the current counter value (count value) in the device specified by Coil by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- Counting is disabled while the operation result remains on. (Count input does not need to be converted into pulses.)
- After counting-up, the count value and contact status remain unchanged until the RST instruction is executed.
- When the set value is 0, the same processing is performed as when it is set to 1.

Operation error

There is no operation error.

Long counter

OUT_LC

This instruction increments the current long counter value (count value) by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value, the normally open contact of the long counter turns on (continuity state) and the normally closed contact turns off (non-continuity state).

ST
ENO:=OUT_C(EN,Coil,Value);

■Execution condition

Instruction	Execution condition
OUT_LC	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type
Coil	Long counter number	—	ANY_BOOL[*1]
Value	Set value for the long counter	0 to 4294967295	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

*1 Only long counter type labels can be used.

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
Coil	—	—	—	—	—
Value	—	—	○	○	○[*1]

*1 Only K (decimal constant) can be used.

Processing details

- This instruction increments the current long counter value (count value) in the device specified by Coil by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- Counting is disabled while the operation result remains on. (Count input does not need to be converted into pulses.)
- After counting-up, the count value and contact status remain unchanged until the RST instruction is executed.
- When the set value is 0, the same processing is performed as when it is set to 1.

Operation error

There is no operation error.

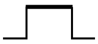
Setting devices

SET

This instruction turns on the specified bit.

ST
ENO:=SET(EN,d);

■Execution condition

Instruction	Execution condition
SET	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(d)	Set target bit device number or bit specification of word device	—	ANY_BOOL
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	—	—	—

Processing details

- This instruction changes the device status as follows when the execution command turns on.

Device	Status
Bit device	Turns on the coil or contact.
Bit-specified word device	Sets the specified bit to 1.

- The device that has been turned on remains on even after the execution command turns off. The device that has been turned on can be turned off by using the RST instruction.

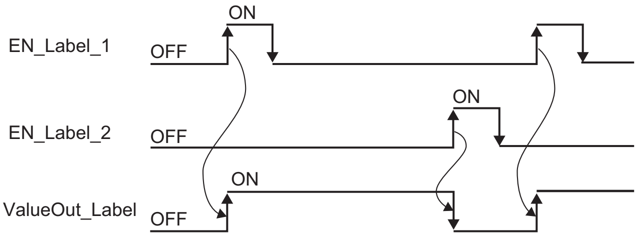
[Label definitions]

Label name	Data type	Class
EN_Label_1	Bit	VAR
EN_Label_2	Bit	VAR
ValueOut_Label	Bit	VAR
ENO_Label	Bit	VAR

[Program]

```
ENO_Label := SET(EN_Label_1,ValueOut_Label);
ENO_Label := RST(EN_Label_2,ValueOut_Label);
```

[Timing chart]



- When the execution command is off, the device status does not change.

Operation error

There is no operation error.

Point

When RX is used, specify a device number that is not used in actual input. If the number that is used in actual input is specified, the data of actual input is written over the input device (RX) specified by the SET instruction.

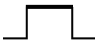
Resetting devices

RST

This instruction turns off the specified device.

ST
ENO:=RST(EN,d);

■Execution condition

Instruction	Execution condition
RST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(d)	Reset target bit device number, bit specification of word device	—	ANY_BOOL
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	—	—	—

Processing details

- This instruction changes the device status as follows when the execution command turns on.

Device	Status
Bit device	Turns off the coil or contact.
Bit-specified word device	Sets the specified bit to 0.

- When the execution command is off, the device status does not change.
- Data except bit type can not be specified in the RST instruction.

Operation error

There is no operation error.

MEMO

PART 4 BASIC INSTRUCTIONS

This part consists of the following chapters.

6 BASIC INSTRUCTIONS

6.1 Arithmetic Operation Instructions

Adding 16-bit binary data

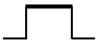
+(_U)

These instructions add the two sets of 16-bit binary data specified.

ST

ENO:=PLUS(EN,s1,s2,d);
ENO:=PLUS_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
+ +_U	

Setting data

■Description, range, data type

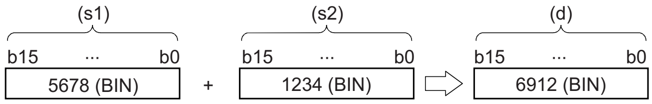
Operand		Description	Range	Data type
(s1)	+	First addend data or the device where the first addend data is stored	-32768 to 32767	ANY16_S
	+_U		0 to 65535	ANY16_U
(s2)	+	Second addend data or the device where the second addend data is stored	-32768 to 32767	ANY16_S
	+_U		0 to 65535	ANY16_U
(d)	+	Device for storing the operation result	—	ANY16_S
	+_U			ANY16_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

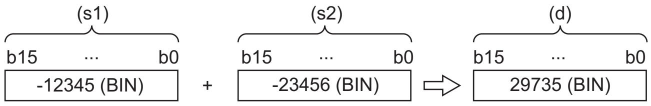
Processing details

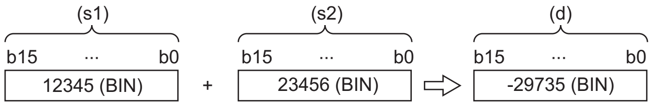
- These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



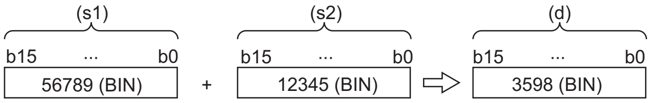
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+ instruction]





[+_U instruction]



Operation error

There is no operation error.

Subtracting 16-bit binary data

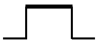
-(_U)

These instructions perform subtraction between the two sets of 16-bit binary data specified.

ST

ENO:=MINUS(EN,s1,s2,d);
ENO:=MINUS_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
- -_U	

Setting data

■Description, range, data type

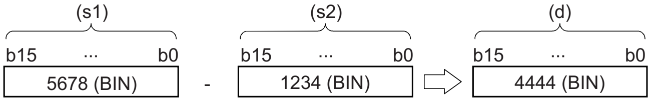
Operand		Description	Range	Data type
(s1)	-	Minuend data or the device where minuend data is stored	-32768 to 32767	ANY16_S
	-_U		0 to 65535	ANY16_U
(s2)	-	Subtrahend data or the device where subtrahend data is stored	-32768 to 32767	ANY16_S
	-_U		0 to 65535	ANY16_U
(d)	-	Device for storing the operation result	—	ANY16_S
	-_U			ANY16_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

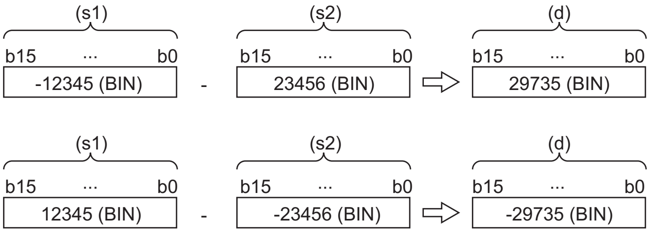
Processing details

- These instructions subtract the 16-bit binary data in the device specified by (s2) from the 16-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).

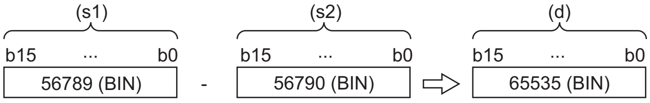


- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[- instruction]



[-_U instruction]



Operation error

There is no operation error.

Adding 32-bit binary data

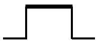
D+(_U)

These instructions add the two sets of 32-bit binary data specified.

ST

ENO:=DPLUS(EN,s1,s2,d);
ENO:=DPLUS_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
D+ D+_U	

Setting data

■Description, range, data type

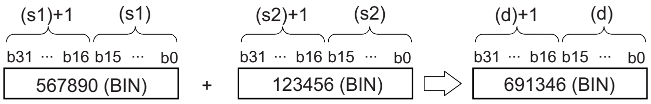
Operand		Description	Range	Data type
(s1)	D+	First addend data or the start device where the first addend data is stored	-2147483648 to 2147483647	ANY32_S
	D+_U		0 to 4294967295	ANY32_U
(s2)	D+	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	ANY32_S
	D+_U		0 to 4294967295	ANY32_U
(d)	D+	Start device for storing the operation result	—	ANY32_S
	D+_U			ANY32_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

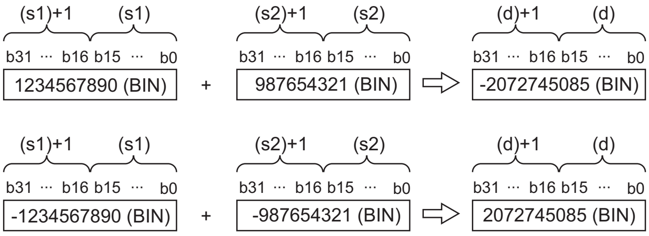
Processing details

- 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 operation result in the device specified by (d).

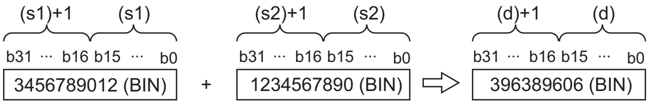


- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+ instruction]



[D+_U instruction]



Operation error

There is no operation error.

Subtracting 32-bit binary data

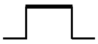
D-(_U)

These instructions perform subtraction between the two sets of 32-bit binary data specified.

ST

ENO:=DMINUS(EN,s1,s2,d);
ENO:=DMINUS_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
D- D-_U	

Setting data

■Description, range, data type

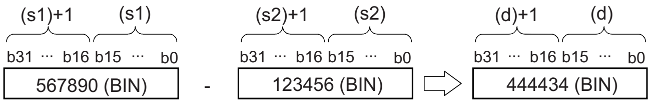
Operand		Description	Range	Data type
(s1)	D-	Minuend data or the start device where minuend data is stored	-2147483648 to 2147483647	ANY32_S
	D-_U		0 to 4294967295	ANY32_U
(s2)	D-	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	ANY32_S
	D-_U		0 to 4294967295	ANY32_U
(d)	D-	Start device for storing the operation result	—	ANY32_S
	D-_U			ANY32_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

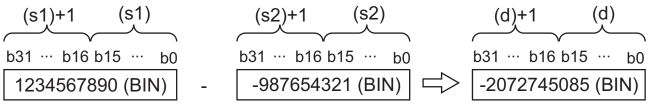
Processing details

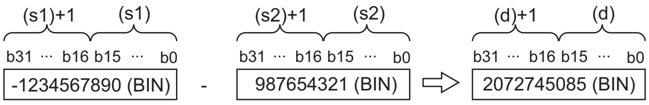
- These instructions subtracts the 32-bit binary data in the device specified by (s2) from the 32-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



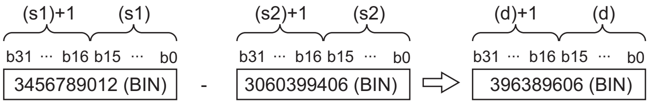
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D- instruction]





[D-_U instruction]



Operation error

There is no operation error.

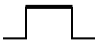
Multiplying 16-bit binary data

*(_U)

These instructions multiply the two sets of 16-bit binary data specified.

ST	
ENO:=MULTI(EN,s1,s2,d);	ENO:=MULTI_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
* *_U	

Setting data

■Description, range, data type

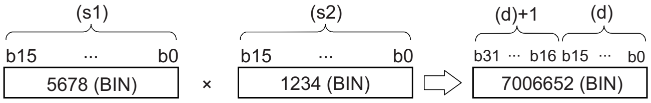
Operand		Description	Range	Data type
(s1)	*	Multiplicand data or the device where multiplicand data is stored	-32768 to 32767	ANY16_S
	*_U		0 to 65535	ANY16_U
(s2)	*	Multiplier data or the device where multiplier data is stored	-32768 to 32767	ANY16_S
	*_U		0 to 65535	ANY16_U
(d)	*	Start device for storing the operation result	—	ANY32_S
	*_U			ANY32_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(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 operation result in the device specified by (d).



- When (d) is a bit device, data should be specified in order from lower bits.

Ex.

Operation 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

There is no operation error.

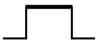
Dividing 16-bit binary data

/(_U)

These instructions perform division between the two sets of 16-bit binary data specified.

ST	
ENO:=DIVISION(EN,s1,s2,d);	ENO:=DIVISION_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
/ /_U	

Setting data

■Description, range, data type

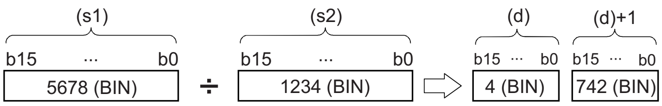
Operand		Description	Range	Data type
(s1)	/	Dividend data or the device where dividend data is stored	-32768 to 32767	ANY16_S
	/_U		0 to 65535	ANY16_U
(s2)	/	Divisor data or the device where divisor data is stored	-32768 to 32767	ANY16_S
	/_U		0 to 65535	ANY16_U
(d)	/	Start device for storing the operation result	—	ANY16_S_ARRAY (Number of elements: 2)
	/_U			ANY16_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(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 operation result in the device specified by (d).



(d): Quotient

(d)+1: Remainder

- As the operation result, the quotient and remainder are stored in 32 bits. When a bit device is specified, the number of digit-specified bits is used to store the quotient and remainder.
  - Quotient···Stored in lower 16 bits.
  - Remainder···Stored in upper 16 bits.

Operation error

Error code	Description
34FFH	The value (divisor) in the device specified by (s2) is 0.

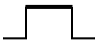
Multiplying 32-bit binary data

D*(_U)

These instructions multiply the two sets of 32-bit binary data specified.

ST	
ENO:=DMULTI(EN,s1,s2,d);	ENO:=DMULTI_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
D* D*_U	

Setting data

■Description, range, data type

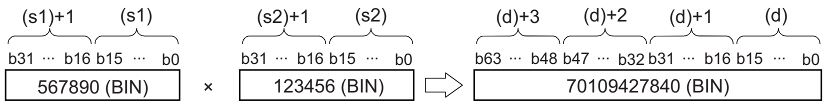
Operand		Description	Range	Data type
(s1)	D*	Multiplicand data or the start device where multiplicand data is stored	-2147483648 to 2147483647	ANY32_S
	D*_U		0 to 4294967295	ANY32_U
(s2)	D*	Multiplier data or the start device where multiplier data is stored	-2147483648 to 2147483647	ANY32_S
	D*_U		0 to 4294967295	ANY32_U
(d)	D*	Start device for storing the operation result	—	ANY32_S_ARRAY (Number of elements: 2)
	D*_U			ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(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 operation result in the device specified by (d).



- When (d) is a bit device, only the lower 32 bits of the operation result are stored. If the upper 32 bits of the operation result are required, temporarily store the result in a word device, and transfer the data stored in (d)+2 and (d)+3 to the specified bit devices.

Ex.

Operation 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

There is no operation error.

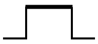
Dividing 32-bit binary data

D/(_U)

These instructions perform division between the two sets of 32-bit binary data specified.

ST	
ENO:=DDIVISION(EN,s1,s2,d);	ENO:=DDIVISION_U(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
D/ D/_U	

Setting data

■Description, range, data type

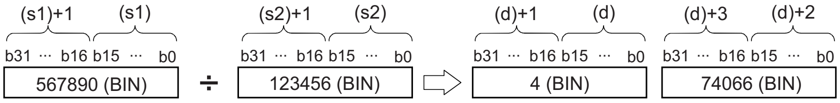
Operand		Description	Range	Data type
(s1)	D/	Dividend data or the start device where dividend data is stored	-2147483648 to 2147483647	ANY32_S
	D/_U		0 to 4294967295	ANY32_U
(s2)	D/	Divisor data or the start device where divisor data is stored	-2147483648 to 2147483647	ANY32_S
	D/_U		0 to 4294967295	ANY32_U
(d)	D/	Start device for storing the operation result	—	ANY32_S_ARRAY (Number of elements: 2)
	D/_U			ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(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 operation result in the device specified by (d).



- As the operation result when a word device is specified, the quotient and remainder are stored in 64 bits. The quotient is stored in lower 32 bits, and the remainder is stored in upper 32 bits. When a bit device is specified, only quotient is stored in 32 bits.

Operation error

Error code	Description
34FFH	The value (divisor) in the device specified by (s2) is 0.

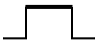
Incrementing 16-bit binary data

INC(_U)

These instructions increment the specified 16-bit binary data by one.

ST	
ENO:=INC(EN,d);	ENO:=INC_U(EN,d);

■Execution condition

Instruction	Execution condition
INC INC_U	

Setting data

■Description, range, data type

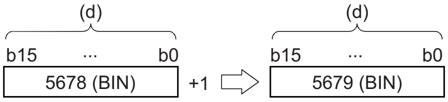
Operand		Description	Range	Data type
(d)	INC	Increment target device	-32768 to 32767	ANY16_S
	INC_U		0 to 65535	ANY16_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- These instructions increment the 16-bit binary data in the device specified by (d) by one.



- When the INC instruction is executed while the data in the device specified by (d) is 32767, -32768 is stored in the device specified by (d).
- When the INC_U instruction is executed while the data in the device specified by (d) is 65535, 0 is stored in the device specified by (d).

Operation error

There is no operation error.

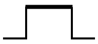
Decrementing 16-bit binary data

DEC(_U)

These instructions decrement the specified 16-bit binary data by one.

ST	
ENO:=DEC(EN,d);	ENO:=DEC_U(EN,d);

■Execution condition

Instruction	Execution condition
DEC DEC_U	

Setting data

■Description, range, data type

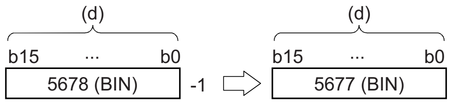
Operand		Description	Range	Data type
(d)	DEC	Decrement target device	-32768 to 32767	ANY16_S
	DEC_U		0 to 65535	ANY16_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- These instructions decrement the 16-bit binary data in the device specified by (d) by one.



- When the DEC instruction is executed while the data in the device specified by (d) is -32768, 32767 is stored in the device specified by (d).
- When the DEC_U instruction is executed while the data in the device specified by (d) is 0, 65535 is stored in the device specified by (d).

Operation error

There is no operation error.

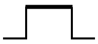
Incrementing 32-bit binary data

DINC(_U)

These instructions increment the specified 32-bit binary data by one.

ST	
ENO:=DINC(EN,d);	ENO:=DINC_U(EN,d);

■Execution condition

Instruction	Execution condition
DINC DINC_U	

Setting data

■Description, range, data type

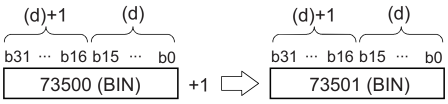
Operand		Description	Range	Data type
(d)	DINC	Increment target start device	-2147483648 to 2147483647	ANY32_S
	DINC_U		0 to 4294967295	ANY32_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- These instructions increment the 32-bit binary data in the device specified by (d) by one.



- When the DINC instruction is executed while the data in the device specified by (d) is 2147483647, -2147483648 is stored in the device specified by (d).
- When the DINC_U instruction is executed while the data in the device specified by (d) is 4294967295, 0 is stored in the device specified by (d).

Operation error

There is no operation error.

Decrementing 32-bit binary data

DDEC(_U)

These instructions decrement the specified 32-bit binary data by one.

ST	
ENO:=DDEC(EN,d);	ENO:=DDEC_U(EN,d);

■Execution condition

Instruction	Execution condition
DDEC DDEC_U	⎍

Setting data

■Description, range, data type

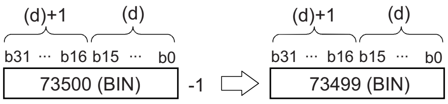
Operand		Description	Range	Data type
(d)	DDEC	Decrement target start device	-2147483648 to 2147483647	ANY32_S
	DDEC_U		0 to 4294967295	ANY32_U
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- These instructions decrement the 32-bit binary data in the device specified by (d) by one.



- When the DDEC instruction is executed while the data in the device specified by (d) is -2147483648, 2147483647 is stored in the device specified by (d).
- When the DDEC instruction is executed while the data in the device specified by (d) is 0, -1 is stored in the device specified by (d).
- When the DDEC_U instruction is executed while the data in the device specified by (d) is 0, 4294967295 is stored in the device specified by (d).

Operation error

There is no operation error.

6.2 Logical Operation Instructions

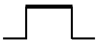
Performing an AND operation on 16-bit data

WAND

This instruction performs an AND operation on the two sets of 16-bit binary data specified.

ST
ENO:=WAND(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
WAND	

Setting data

■Description, range, data type

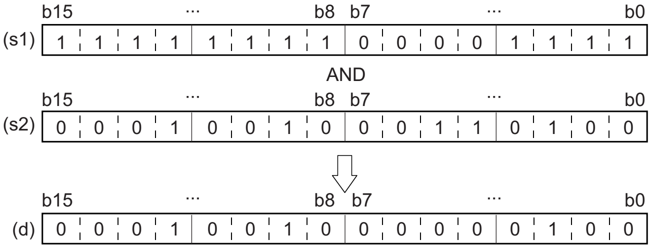
Operand	Description	Range	Data type
(s1)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	ANY16
(s2)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	ANY16
(d)	Device for storing the operation result	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an AND operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



```
      b15         ...        b8  b7        ...        b0
(s1)  1  1  1  1  1  1  1  1  0  0  0  0  1  1  1  1
                            AND
      b15         ...        b8  b7        ...        b0
(s2)  0  0  0  1  0  0  1  0  0  0  1  1  0  1  0  0
                             ⇩
      b15         ...        b8  b7        ...        b0
(d)   0  0  0  1  0  0  1  0  0  0  0  0  0  1  0  0
```

- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

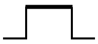
Performing an AND operation on 32-bit data

DAND

This instruction performs an AND operation on the two sets of 32-bit binary data specified.

ST
ENO:=DAND(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
DAND	

Setting data

■Description, range, data type

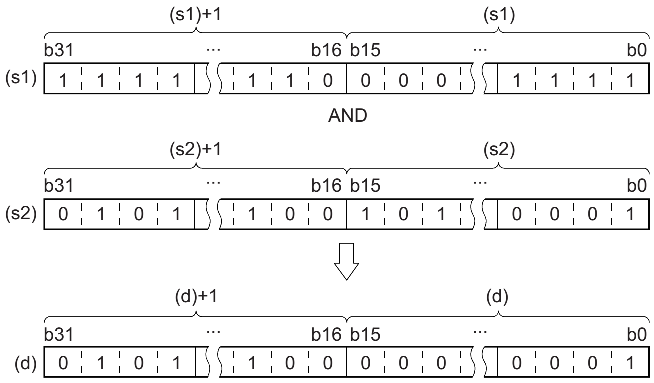
Operand	Description	Range	Data type
(s1)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	ANY32
(s2)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	ANY32
(d)	Start device for storing the operation result	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an AND operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

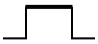
Performing an OR operation on 16-bit data

WOR

This instruction performs an OR operation on the two sets of 16-bit binary data specified.

ST
ENO:=WOR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
WOR	

Setting data

■Description, range, data type

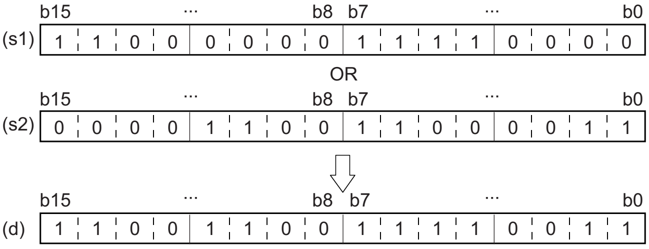
Operand	Description	Range	Data type
(s1)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	ANY16
(s2)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	ANY16
(d)	Device for storing the operation result	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an OR operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



```
      b15          ···          b8  b7          ···          b0
(s1)  1  1  0  0   0  0  0  0   1  1  1  1   0  0  0  0
                                OR
      b15          ···          b8  b7          ···          b0
(s2)  0  0  0  0   1  1  0  0   1  1  0  0   0  0  1  1
                                ⇩
      b15          ···          b8  b7          ···          b0
(d)   1  1  0  0   1  1  0  0   1  1  1  1   0  0  1  1
```

- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

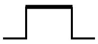
Performing an OR operation on 32-bit data

DOR

This instruction performs an OR operation on the two sets of 32-bit binary data specified.

ST
ENO:=DOR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
DOR	

Setting data

■Description, range, data type

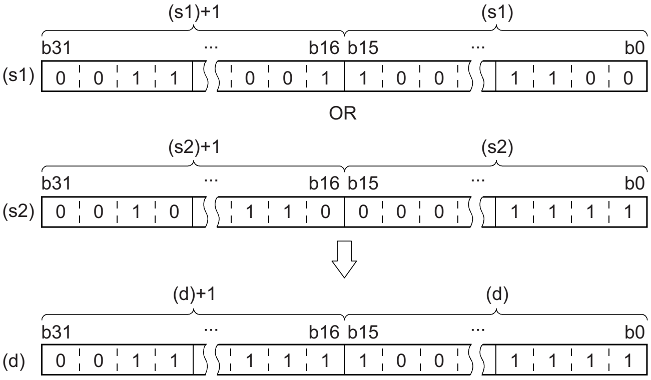
Operand	Description	Range	Data type
(s1)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	ANY32
(s2)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	ANY32
(d)	Start device for storing the operation result	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an OR operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

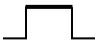
Performing an XOR operation on 16-bit data

WXOR

This instruction performs an XOR operation on the two sets of 16-bit binary data specified.

ST
ENO:=WXOR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
WXOR	

Setting data

■Description, range, data type

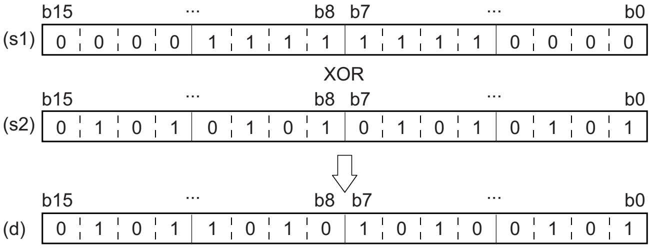
Operand	Description	Range	Data type
(s1)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	ANY16
(s2)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	ANY16
(d)	Device for storing the operation result	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an XOR operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



```
      b15         ...        b8  b7          ...        b0
(s1)  0  0  0  0  1  1  1  1  1  1  1  1  0  0  0  0
                            XOR
      b15         ...        b8  b7          ...        b0
(s2)  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1
                             ⇩
      b15         ...        b8  b7          ...        b0
(d)   0  1  0  1  1  0  1  0  1  0  1  0  0  1  0  1
```

- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

Performing an XOR operation on 32-bit data

DXOR

This instruction performs an XOR operation on the two sets of 32-bit binary data specified.

ST
ENO:=DXOR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
DXOR	⎍

Setting data

■Description, range, data type

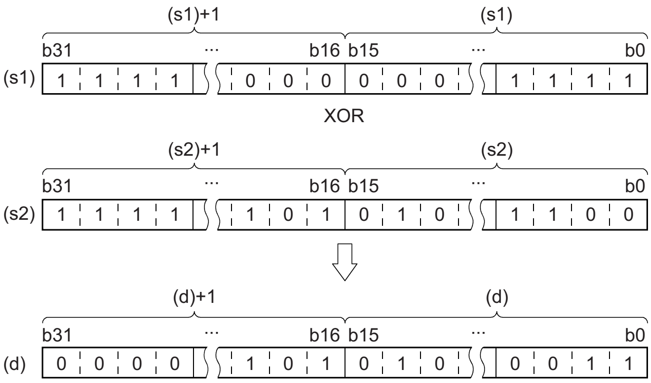
Operand	Description	Range	Data type
(s1)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	ANY32
(s2)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	ANY32
(d)	Start device for storing the operation result	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an XOR operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and stores the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

Performing an XNOR operation on 16-bit data

WXNR

This instruction performs an XNOR operation on the two sets of 16-bit binary data specified.

ST
ENO:=WXNR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
WXNR	⎍

Setting data

■Description, range, data type

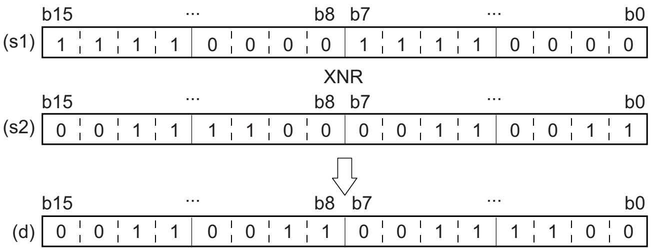
Operand	Description	Range	Data type
(s1)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	ANY16
(s2)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	ANY16
(d)	Device for storing the operation result	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an exclusive NOR 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), and stores the operation result in the device specified by (d).



```
      b15          ...       b8  b7          ...         b0
(s1)  1  1  1  1  0  0  0  0  1  1  1  1  0  0  0  0
                             XNR
      b15          ...       b8  b7          ...         b0
(s2)  0  0  1  1  1  1  0  0  0  0  1  1  0  0  1  1
                              ⇩
      b15          ...       b8  b7          ...         b0
(d)   0  0  1  1  0  0  1  1  0  0  1  1  1  1  0  0
```

- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

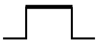
Performing an XNOR operation on 32-bit data

DXNR

This instruction performs an XNOR operation on the two sets of 32-bit binary data specified.

ST
ENO:=DXNR(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
DXNR	

Setting data

■Description, range, data type

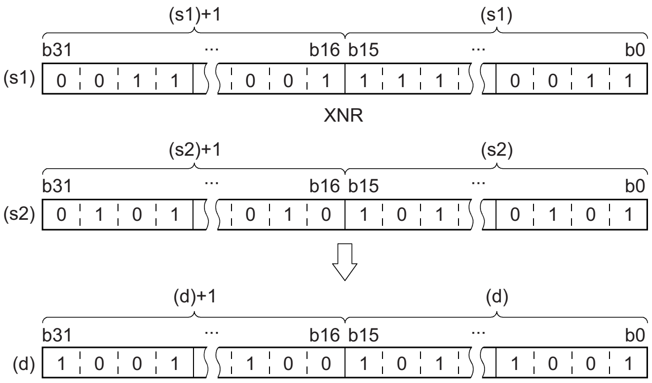
Operand	Description	Range	Data type
(s1)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	ANY32
(s2)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	ANY32
(d)	Start device for storing the operation result	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	○	○	○	○	○
(s2)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction performs an XNOR 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), and stores the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

There is no operation error.

6.3 Data Conversion Instructions

Two's complement of 16-bit binary data (sign inversion)

NEG

Invert the sign of 16-bit binary device.

ST
ENO:=NEG(EN,d);

■Execution condition

Instruction	Execution condition
NEG	⎍

Setting data

■Description, range, data type

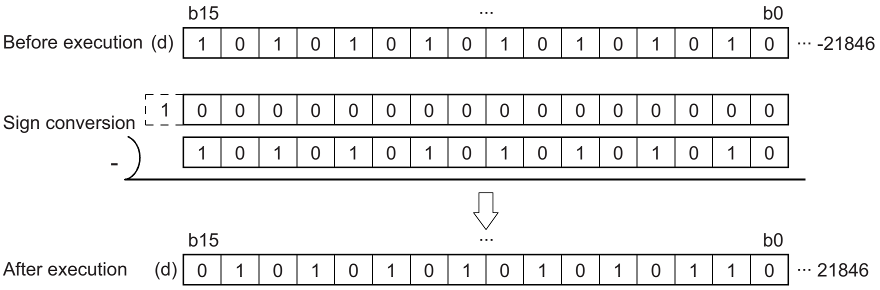
Operand	Description	Range	Data type
(d)	Device where the data subjected to two's complement is stored	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- This instruction inverts the sign of the 16-bit binary data in the device specified by (d), and stores the inverted data in the device specified by (d).
- The instructions are used to invert positive and negative signs.



```
                        b15                    ···                    b0
Before execution  (d)   1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0   ··· -21846

                      1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Sign conversion
                 -)     1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
                 ------------------------------------------
                                       ⇩
                        b15                    ···                    b0
After execution   (d)   0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0   ··· 21846
```

Operation error

There is no operation error.

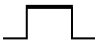
Two's complement of 32-bit binary data (sign inversion)

DNEG

This instruction inverts the sign of 32-bit binary device.

ST
ENO:=DNEG(EN,d);

■Execution condition

Instruction	Execution condition
DNEG	

Setting data

■Description, range, data type

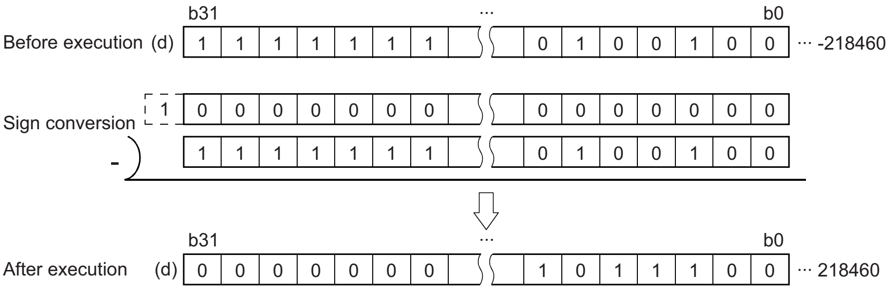
Operand	Description	Range	Data type
(d)	Start device where the data subjected to two's complement is stored	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	○	○	○	○	—

Processing details

- This instruction inverts the sign of the 32-bit binary data in the device specified by (d), and stores the inverted data in the device specified by (d).
- The instructions are used to invert positive and negative signs.



Operation error

There is no operation error.

6.4 Data Transfer Instructions

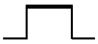
Transferring 16-bit binary data

MOV

This instruction transfers the 16-bit binary data in the device specified.

ST
ENO:=MOV(EN,s,d);

■Execution condition

Instruction	Execution condition
MOV	

Setting data

■Description, range, data type

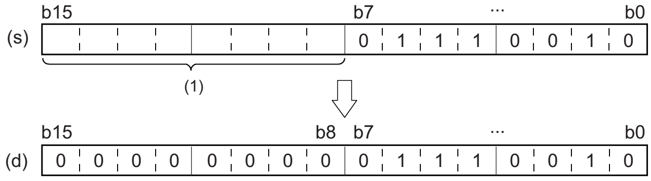
Operand	Description	Range	Data type
(s)	Transfer source data or the number of the device where the transfer source data is stored	-32768 to 32767	ANY16
(d)	Transfer destination device number	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction transfers the 16-bit binary data in the device specified by (s) to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and transferred.



(1) If data specified by (s) is less than 16 bits, 0s are added and transferred.

Operation error

There is no operation error.

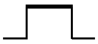
Transferring 32-bit binary data

DMOV

This instruction transfers the 32-bit binary data in the device specified.

ST
ENO:=DMOV(EN,s,d);

■Execution condition

Instruction	Execution condition
DMOV	

Setting data

■Description, range, data type

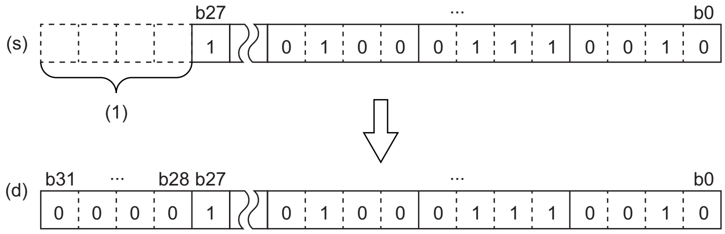
Operand	Description	Range	Data type
(s)	Transfer source data or the number of the device where the transfer source data is stored	-2147483648 to 2147483647	ANY32
(d)	Transfer destination device number	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction transfers the 32-bit binary data in the device specified by (s) to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and transferred.



(1) If data specified by (s) is less than 32 bits, 0s are added and transferred.

Operation error

There is no operation error.

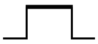
Inverting and transferring 16-bit binary data

CML

This instruction inverts the specified 16-bit binary data bit by bit, and transfer the inverted data.

ST
ENO:=CML(EN,s,d);

■Execution condition

Instruction	Execution condition
CML	

Setting data

■Description, range, data type

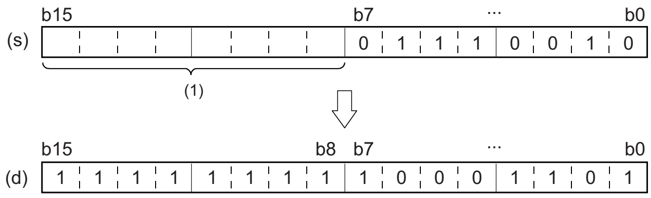
Operand	Description	Range	Data type
(s)	Inversion target data or the number of the device where the inversion target data is stored	-32768 to 32767	ANY16
(d)	Number of the device for storing the inverted data	—	ANY16
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction inverts the 16-bit binary data in the device specified by (s) bit by bit, and transfer the inverted data to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and inverted.



(1) If data specified by (s) is less than 16 bits, 0s are added and inverted.

Operation error

There is no operation error.

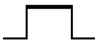
Inverting and transferring 32-bit binary data

DCML

This instruction inverts the specified 32-bit binary data bit by bit, and transfer the inverted data.

ST
ENO:=DCML(EN,s,d);

■Execution condition

Instruction	Execution condition
DCML	

Setting data

■Description, range, data type

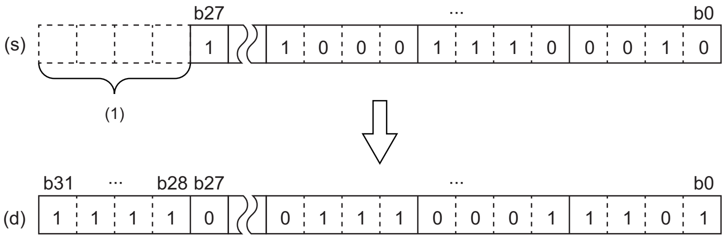
Operand	Description	Range	Data type
(s)	Inversion target data or the number of the device where the inversion target data is stored	-2147483648 to 2147483647	ANY32
(d)	Number of the device for storing the inverted data	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- This instruction inverts the 32-bit binary data in the device specified by (s) bit by bit, and transfer the inverted data to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and inverted.



(1) If data specified by (s) is less than 32 bits, 0s are added and inverted.

Operation error

There is no operation error.

Inverting and transferring 1-bit data

CMLB

This instruction inverts the specified bit data, and transfer the inverted data.

ST
ENO:=CMLB(EN,s,d);

■Execution condition

Instruction	Execution condition
CMLB	⎍

Setting data

■Description, range, data type

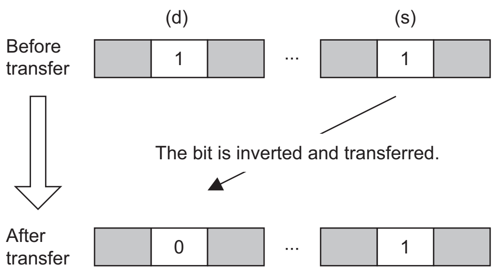
Operand	Description	Range	Data type
(s)	Inversion target data or the number of the device where the inversion target data is stored	—	ANY_BOOL
(d)	Transfer destination device number	—	ANY_BOOL
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	—	—
(d)	○	○	○	—	—

Processing details

- This instruction inverts the bit data in the device specified by (s), and transfer the inverted data to the device specified by (d).



Operation error

There is no operation error.

Transferring 1-bit data

MOVB

This instruction transfers the specified 1-bit data.

ST
ENO:=MOVB(EN,s,d);

■Execution condition

Instruction	Execution condition
MOVB	⎍

Setting data

■Description, range, data type

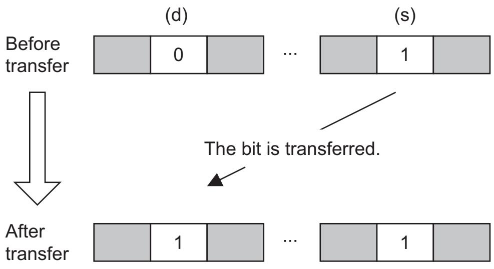
Operand	Description	Range	Data type
(s)	Number of the device where the transfer target data is stored	—	ANY_BOOL
(d)	Transfer destination device number	—	ANY_BOOL
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	—	—
(d)	○	○	○	—	—

Processing details

- This instruction transfers the bit data in the device specified by (s) to the device specified by (d).



Operation error

There is no operation error.

MEMO

PART 5 APPLICATION INSTRUCTIONS

This part consists of the following chapters.

7 PROGRAM CONTROL

7.1 Program Execution Control Instructions

Disabling/enabling interrupt programs

DI, EI

- DI: This instruction disables execution of fixed scan execution type programs.
- EI: This instruction clears the fixed scan execution type programs execution disabled state.

ST

```
ENO:=DI(EN);
ENO:=EI(EN);
```

■Execution condition

Instruction	Execution condition
DI EI	Every scan

Processing details

■DI

- This instruction disables execution of fixed scan execution type programs.
- When the system is powered on or the CPU module is reset, the system is in the state where the DI instruction has been executed.
- The DI (Disabling interrupt programs) instruction cannot be executed in fixed scan execution type programs. If executed, no processing is performed.
- The execution of the EI instruction enables the interrupt that has been disabled by a single DI (Disabling interrupt programs) instruction. Note that if the DI (Disabling interrupt programs) instruction is nested, the interrupt will not be enabled unless executing the EI instruction, including the nesting instruction.

[Program Example (DI nesting)]

```
DI(TRUE);//1st nesting of DI instruction
DI(TRUE);//2nd nesting of DI instruction
EI(TRUE);//2nd nesting of DI instruction interrupted
EI(TRUE);//Interrupt enabled
```

■EI

- This instruction clears the fixed scan execution type programs execution disabled state that has been set by the DI (Disabling interrupt programs) instruction, and enables execution of fixed scan execution type programs.

Operation error

Error code	Description
34FBH	More than 16 DI (Disabling interrupt programs) instructions are nested.

7.2 Program Control Instructions

Changing the program execution type to standby type

PSTOP

This instruction changes the execution type of the program with the specified program name to a standby type.

ST
ENO:=PSTOP(EN,program name);

■Execution condition

Instruction	Execution condition
PSTOP	⎍

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(Program name)	Character string data of the program name to be changed to a standby type, or the start device where the character string data is stored	—	ANYSTRING_DOUBLE
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(Program name)	—	—	○	—	○

Processing details

- This instruction changes the execution type of the program stored in the device specified by (program name) to a standby type.
- This instruction is accepted during END processing of the program that executed the instruction, and the execution type changes to a standby type during END processing of the specified program.
- The PSTOP instruction takes precedence even when the execution type is specified in parameter.

Operation error

Error code	Description
2840H	The program specified by (program name) does not exist.

Point

For how to change the program execution type, refer to "Motion Module Programs" in the following manual.
📖MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

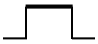
Changing the program execution type to scan execution type

PSCAN

This instruction changes the execution type of the program with the specified program name to a normal execution type.

ST
ENO:=PSCAN(EN,program name);

■Execution condition

Instruction	Execution condition
PSCAN	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(Program name)	The program name to be changed to a normal execution type, or the start device where the program name is stored	—	ANYSTRING_DOUBLE
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(Program name)	—	—	○	—	○

Processing details

- This instruction changes the execution type of the program stored in the device specified by (program name) to a normal execution type.
- This instruction is accepted during END processing of the program that executed the instruction, and the execution type changes to a normal execution type during END processing of the specified program.
- The PSCAN instruction takes precedence even when the execution type is specified in parameter.

Operation error

Error code	Description
2840H	The program specified by (program name) does not exist.

Point

For how to change the program execution type, refer to "Motion Module Programs" in the following manual.
MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

8 DATA PROCESSING

8.1 Data Processing Instructions

Adding 16-bit binary data

WSUM(_U)

These instructions add the (n) points of 16-bit binary data from the specified device.

ST	
ENO:=WSUM(EN,s,n,d);	ENO:=WSUM_U(EN,s,n,d);

■Execution condition

Instruction	Execution condition
WSUM WSUM_U	⎍

Setting data

■Description, range, data type

Operand		Description	Range	Data type
(s)	WSUM	Start device where the data for calculating the total value are stored	—	ANY16_S[*1]
	WSUM_U			ANY16_U[*1]
(d)	WSUM	Start device for storing the total value	—	ANY32_S
	WSUM_U			ANY32_U
(n)		Number of data	0 to 65535	ANY16
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

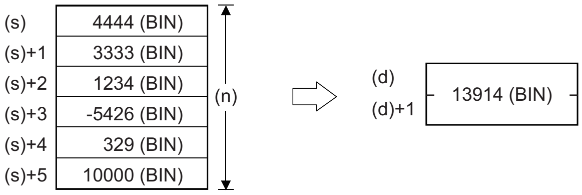
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices/labels

Operand	Bit		Word		Constant
(s)	—	—	○	—	—
(d)	○	○	○	○	—
(n)	○	○	○	○	○

Processing details

- 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).



(s)	4444 (BIN)
(s)+1	3333 (BIN)
(s)+2	1234 (BIN)
(s)+3	-5426 (BIN)
(s)+4	329 (BIN)
(s)+5	10000 (BIN)

Operation error

There is no operation error.

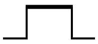
Adding 32-bit binary data

DWSUM(_U)

These instructions add the (n) points of 32-bit binary data in the devices starting from the specified one.

ST	
ENO:=DWSUM(EN,s,n,d);	ENO:=DWSUM_U(EN,s,n,d);

■Execution condition

Instruction	Execution condition
DWSUM DWSUM_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type
(s)	DWSUM	Start device where the data for calculating the total value are stored	—	ANY32_S*1
	DWSUM_U			ANY32_U*1
(d)	DWSUM	Start device for storing the total value	—	ANY32_ARRAY (Number of elements: 2)
	DWSUM_U			
(n)		Number of data	0 to 65535	ANY16
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

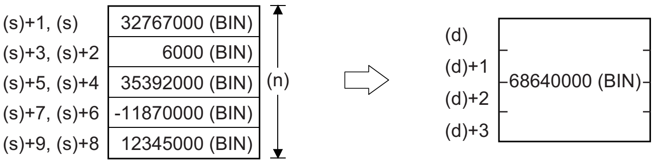
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices/labels

Operand	Bit		Word		Constant
(s)	—	—	○	—	—
(d)	—	—	○	—	—
(n)	○	○	○	○	○

Processing details

- 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).



Operation error

There is no operation error.

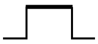
Calculating the mean value of 16-bit binary data

MEAN(_U)

These instructions calculate the average value of the (n) points of 16-bit data in the devices starting from the specified one.

ST	
ENO:=MEAN(EN,s,n,d);	ENO:=MEAN_U(EN,s,n,d);

■Execution condition

Instruction	Execution condition
MEAN MEAN_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type
(s)	MEAN	Start device where the data for calculating the average value are stored	—	ANY16_S[*1]
	MEAN_U			ANY16_U[*1]
(d)	MEAN	Device for storing the mean value	—	ANY16_S
	MEAN_U			ANY16_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	ANY16
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

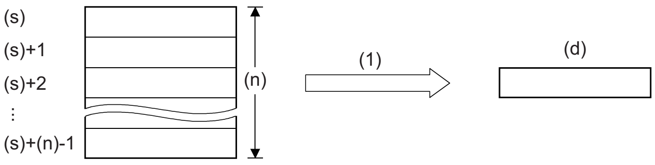
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices/labels

Operand	Bit		Word		Constant
(s)	—	—	○	—	—
(d)	—	—	○	—	—
(n)	○	○	○	○	○

Processing details

- These instructions calculate the average value of the (n) points of 16-bit binary data in the devices starting from the one specified by (s), and stores the average value in the device specified by (d).



(1) Mean value

- If the calculation result is not an integer, the first decimal place is rounded down.
- When (n) is 0, the processing is not performed.

Operation error

There is no operation error.

Calculating the mean value of 32-bit binary data

DMEAN(_U)

These instructions calculate the average value of the (n) points of 32-bit data in the devices starting from the specified one.

ST	
ENO:=DMEAN(EN,s,n,d);	ENO:=DMEAN_U(EN,s,n,d);

■Execution condition

Instruction	Execution condition
DMEAN DMEAN_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type
(s)	DMEAN	Start device where the data for calculating the average value are stored	—	ANY32_S[*1]
	DMEAN_U			ANY32_U[*1]
(d)	DMEAN	Start device for storing the average value	—	ANY32_S
	DMEAN_U			ANY32_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	ANY16
EN		Execution condition	—	BOOL
ENO		Execution result	—	BOOL

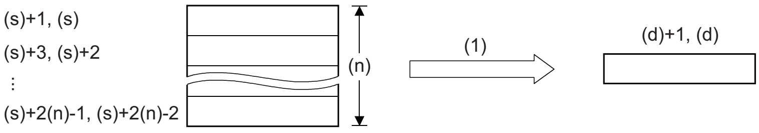
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices/labels

Operand	Bit		Word		Constant
(s)	—	—	○	—	—
(d)	—	—	○	—	—
(n)	○	○	○	○	○

Processing details

- These instructions calculate the average value of the (n) points of 32-bit binary data in the devices starting from the one specified by (s), and stores the average value in the device specified by (d).



(1) Mean value

- If the calculation result is not an integer, the first decimal place is rounded down.
- When (n) is 0, the processing is not performed.

Operation error

There is no operation error.

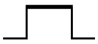
Calculating the square root of 32-bit binary data

DSQRT

These instructions perform a square root operation of the specified 32-bit binary data.

ST
ENO:=DSQRT(EN,s,d);

■Execution condition

Instruction	Execution condition
DSQRT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s)	Device where the data whose square root is to be calculated is stored	0 to 4294967295	ANY32
(d)	Device where the obtained square root is stored	—	ANY32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	○	○	○	○	○
(d)	○	○	○	○	—

Processing details

- These instructions perform a square root operation of the 32-bit binary data specified by (s), and stores the result in (d). The obtained square root is an integer because the decimal places are rounded down.

√(s)+1, (s)→(d)

Operation error

There is no operation error.

9 STRING PROCESSING

9.1 String Processing Instructions

Transferring string data

$MOV

This instruction transfers string data to the specified device number and later.

ST
ENO:=STRINGMOV(EN,s,d);

■Execution condition

Instruction	Execution condition
$MOV	⎍

Setting data

■Descriptions, ranges, and data types

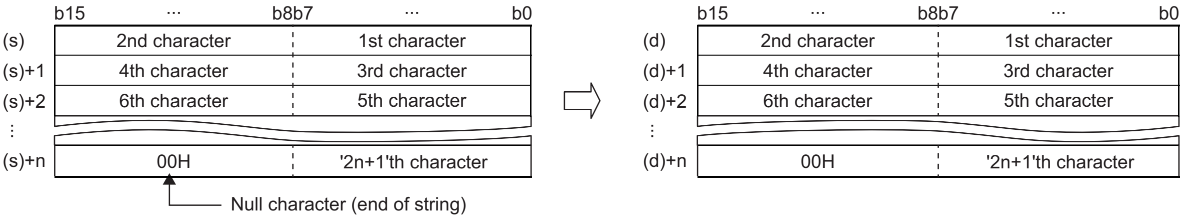
Operand	Description	Range	Data type
(s)	Character string to be transferred (maximum of 255 characters) or the start device containing such character string	—	ANYSTRING_SINGLE
(d)	Start device for storing the transferred character string	—	ANYSTRING_SINGLE
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

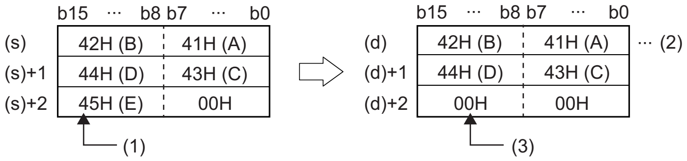
Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction transfers the character string data in the device specified by (s) to the device number specified by (d) and later. The character strings specified by (s) or the character strings from the device number specified by (s) to the device number containing 00H are transferred all at once.



- When 00H is stored in the lower byte of (s)+n, 00H will be stored in both upper and lower bytes of (d)+n.



(1) Data (upper byte) is not transferred.

(2) Data remain the same.

(3) 00H is automatically stored in the upper byte.

Operation error

Error code	Description
3506H	There is no NULL code (00H) in the setting area specified by (s) and later in the device/label memory.
3507H	The number of characters in the string specified by (s) exceeds 16383.
3508H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Transferring Unicode string data

$MOV_WS

This instruction transfers character string [Unicode] data to the specified device number and later.

ST
ENO:=STRINGMOV_WS(EN,s,d);

■Execution condition

Instruction	Execution condition
$MOV_WS	⎍

Setting data

■Descriptions, ranges, and data types

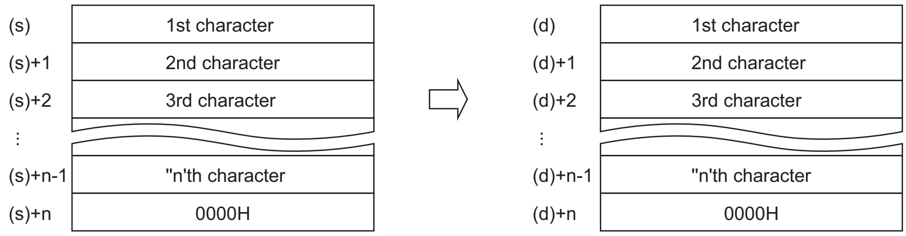
Operand	Description	Range	Data type
(s)	Character string [Unicode] to be transferred (maximum of 255 characters) or the start device containing the character string [Unicode]	—	ANYSTRING_DOUBLE
(d)	Start device for storing the transferred character string [Unicode]	—	ANYSTRING_DOUBLE
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction transfers the character string [Unicode] data in the device specified by (s) to the device number specified by (d) and later. The character strings [Unicode] specified by (s) or the character strings [Unicode] from the device number specified by (s) to the device number containing 0000H are transferred all at once.



(s)	1st character
(s)+1	2nd character
(s)+2	3rd character
⋮	
(s)+n-1	"n'th character
(s)+n	0000H

(d)	1st character
(d)+1	2nd character
(d)+2	3rd character
⋮	
(d)+n-1	"n'th character
(d)+n	0000H

Operation error

Error code	Description
3506H	There is no 0000H in the setting area specified by (s) and later in the device/label memory.
3507H	The number of characters in the character string [Unicode] specified by (s) exceeds 16383.
3508H	The entire character string [Unicode] cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

10 REAL VALUE PEOCESSING

10.1 Floating-point instruction

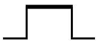
Adding single-precision real numbers

E+

This instruction adds single-precision real numbers.

ST
ENO:=EPLUS(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
E+	

Setting data

■Description, range, data type

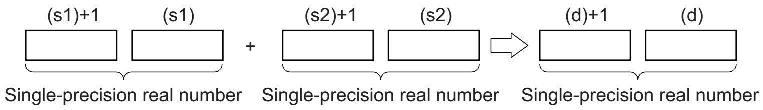
Operand	Description	Range	Data type
(s1)	First addend data or the start device where the first addend data is stored	0, 2^{-126}≤|(s1)|<2^{128}	ANYREAL_32
(s2)	Second addend data or the start device where the second addend data is stored	0, 2^{-126}≤|(s2)|<2^{128}	ANYREAL_32
(d)	Start device for storing the operation result	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	○	—
(s2)	—	—	○	○	—
(d)	—	—	○	○	—

Processing details

- This instruction adds 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).


(s1)+1
(s1)
+
(s2)+1
(s2)
(d)+1
(d)
Single-precision real number
Single-precision real number
Single-precision real number

- Value 0 or 2^{-126}≤|specified value (stored value)| <2^{128} can be specified or stored in the devices specified by (s1), (s2), and (d).

Operation error

Error code	Description
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) |(d)|<2^{128}

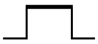
Subtracting single-precision real numbers

E-

This instruction performs subtraction between single-precision real numbers.

ST
ENO:=EMINUS(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
E-	

Setting data

■Description, range, data type

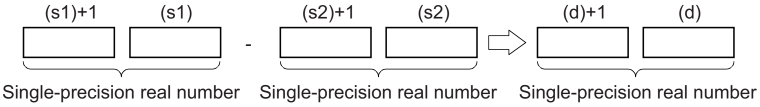
Operand	Description	Range	Data type
(s1)	Minuend data or the start device where minuend data is stored	0, 2^{-126}≤|(s1)|<2^{128}	ANYREAL_32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, 2^{-126}≤|(s2)|<2^{128}	ANYREAL_32
(d)	Start device for storing the operation result	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	○	—
(s2)	—	—	○	○	—
(d)	—	—	○	○	—

Processing details

- This instruction subtracts 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).


(s1)+1
(s1)
-
(s2)+1
(s2)
(d)+1
(d)
Single-precision real number
Single-precision real number
Single-precision real number

- Value 0 or 2^{-126}≤|specified value (stored value)| <2^{128} can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) |(d)|<2^{128}

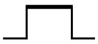
Adding double-precision real numbers

ED+

This instruction adds double-precision real numbers.

ST
ENO:=EDPLUS(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
ED+	

Setting data

■Description, range, data type

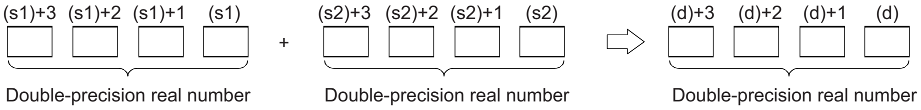
Operand	Description	Range	Data type
(s1)	First addend data or the start device where the first addend data is stored	0, 2^{-1022}≤|(s1)|<2^{1024}	ANYREAL_64
(s2)	Second addend data or the start device where the second addend data is stored	0, 2^{-1022}≤|(s2)|<2^{1024}	ANYREAL_64
(d)	Start device for storing the operation result	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	—	—
(s2)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction adds the double-precision real number in the device specified by (s1) to the double-precision real number in the device specified by (s2), and store the result in the device specified by (d).


(s1)+3
(s1)+2
(s1)+1
(s1)
+
(s2)+3
(s2)+2
(s2)+1
(s2)
(d)+3
(d)+2
(d)+1
(d)
Double-precision real number
Double-precision real number
Double-precision real number

- Value 0 or 2^{-1022}≤|specified value (stored value)| <2^{1024} can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) |(d)|<2^{1024}

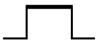
Subtracting double-precision real numbers

ED-

This instruction performs subtraction between double-precision real numbers.

ST
ENO:=EDMINUS(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
ED-	

Setting data

■Description, range, data type

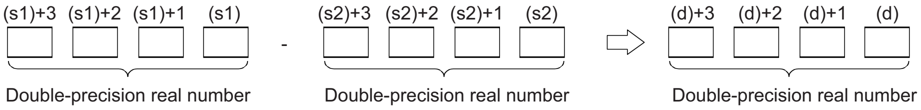
Operand	Description	Range	Data type
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-1022} \leq \|(s1)\| < 2^{1024}$	ANYREAL_64
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022} \leq \|(s2)\| < 2^{1024}$	ANYREAL_64
(d)	Start device for storing the operation result	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	—	—
(s2)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction subtracts the double-precision real number in the device specified by (s2) from the double-precision real number in the device specified by (s1), and store the result in the device specified by (d).


(s1)+3
(s1)+2
(s1)+1
(s1)
-
(s2)+3
(s2)+2
(s2)+1
(s2)
(d)+3
(d)+2
(d)+1
(d)
Double-precision real number
Double-precision real number
Double-precision real number

- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

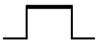
Multiplying single-precision real numbers

E*

This instruction multiplies single-precision real numbers.

ST
ENO:=EMULTI(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
E*	

Setting data

■Description, range, data type

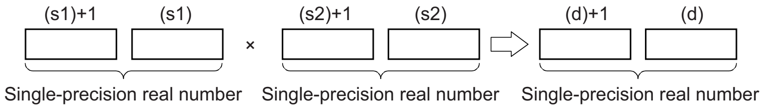
Operand	Description	Range	Data type
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, 2^{-126}≤|(s1)|<2^{128}	ANYREAL_32
(s2)	Multiplier data or the start device where multiplier data is stored	0, 2^{-126}≤|(s2)|<2^{128}	ANYREAL_32
(d)	Start device for storing the operation result	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	○	—
(s2)	—	—	○	○	—
(d)	—	—	○	○	—

Processing details

- This instruction multiplies 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 multiplication result in the device specified by (d).


(s1)+1
(s1)
×
(s2)+1
(s2)
(d)+1
(d)
Single-precision real number
Single-precision real number
Single-precision real number

- Value 0 or 2^{-126}≤|specified value (stored value)| <2^{128} can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) |(d)|<2^{128}

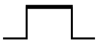
Dividing single-precision real numbers

E/

This instruction performs division between single-precision real numbers.

ST
ENO:=EDIVISION(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
E/	

Setting data

■Description, range, data type

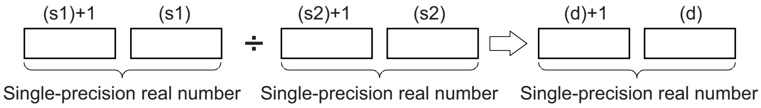
Operand	Description	Range	Data type
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-126}\leq\|(s1)\|<2^{128}$	ANYREAL_32
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-126}\leq\|(s2)\|<2^{128}$	ANYREAL_32
(d)	Start device for storing the operation result	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	○	—
(s2)	—	—	○	○	—
(d)	—	—	○	○	—

Processing details

- This instruction divides 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 division result in the device specified by (d).


(s1)+1
(s1)
÷
(s2)+1
(s2)
(d)+1
(d)
Single-precision real number
Single-precision real number
Single-precision real number

- Value 0 or $2^{-126}\leq$|specified value (stored value)| $<2^{128}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
34FFH	The data (divisor) in the device specified by (s2) is 0.
3502H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

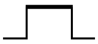
Multiplying double-precision real numbers

ED*

This instruction multiplies double-precision real numbers.

ST
ENO:=EDMULTI(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
ED*	

Setting data

■Description, range, data type

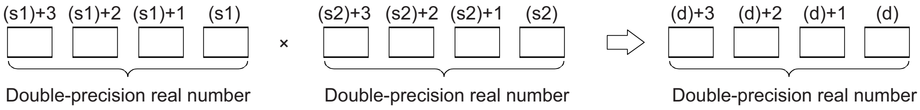
Operand	Description	Range	Data type
(s1)	Multiplicand data or the start device where multiplicand data is stored	$0, 2^{-1022} \leq \|(s1)\| < 2^{1024}$	ANYREAL_64
(s2)	Multiplier data or the start device where multiplier data is stored	$0, 2^{-1022} \leq \|(s2)\| < 2^{1024}$	ANYREAL_64
(d)	Start device for storing the operation result	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	—	—
(s2)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction multiplies the double-precision real number in the device specified by (s1) by the double-precision real number in the device specified by (s2), and store the multiplication result in the device specified by (d).


(s1)+3
(s1)+2
(s1)+1
(s1)
×
(s2)+3
(s2)+2
(s2)+1
(s2)
(d)+3
(d)+2
(d)+1
(d)
Double-precision real number
Double-precision real number
Double-precision real number

- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $<2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

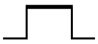
Dividing double-precision real numbers

ED/

This instruction performs division between double-precision real numbers.

ST
ENO:=EDDIVISION(EN,s1,s2,d);

■Execution condition

Instruction	Execution condition
ED/	

Setting data

■Description, range, data type

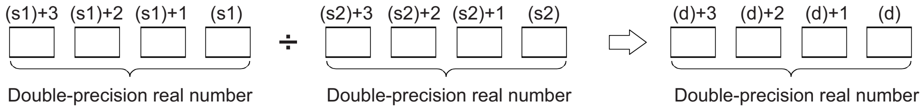
Operand	Description	Range	Data type
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-1022}\leq|(s1)|<2^{1024}$	ANYREAL_64
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-1022}\leq|(s2)|<2^{1024}$	ANYREAL_64
(d)	Start device for storing the operation result	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s1)	—	—	○	—	—
(s2)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction divides the double-precision real number in the device specified by (s1) by the double-precision real number in the device specified by (s2), and store the division result in the device specified by (d).


(s1)+3
(s1)+2
(s1)+1
(s1)
÷
(s2)+3
(s2)+2
(s2)+1
(s2)
(d)+3
(d)+2
(d)+1
(d)
Double-precision real number
Double-precision real number
Double-precision real number

- Value 0 or $2^{-1022}\leq$|specified value (stored value)| $<2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 24 Precautions

Operation error

Error code	Description
34FFH	The data (divisor) in the device specified by (s2) is 0.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)|<2^{1024}$

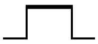
Inverting the sign of single-precision real number

ENEG

This instruction inverts the sign of single-precision real number data.

ST
ENO:=ENEG(EN,d);

■Execution condition

Instruction	Execution condition
ENEG	

Setting data

■Description, range, data type

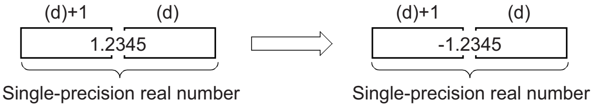
Operand	Description	Range	Data type
(d)	Start device containing the single-precision real number data subject to sign inversion	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	—	—	○	○	—

Processing details

- This instruction inverts the sign of the single-precision real number in the device specified by (d) and store the inverted data in the device specified by (d).



- The instructions are used to invert positive and negative signs.

Operation error

Error code	Description
3501H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

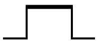
Inverting the sign of double-precision real number

EDNEG

This instruction inverts the sign of double-precision real number data.

ST
ENO:=EDNEG(EN,d);

■Execution condition

Instruction	Execution condition
EDNEG	

Setting data

■Description, range, data type

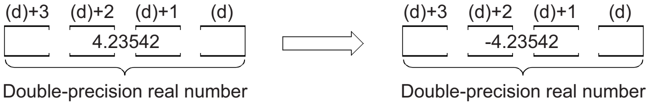
Operand	Description	Range	Data type
(d)	Start device containing the double-precision real number subject to sign inversion	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(d)	—	—	○	—	—

Processing details

- This instruction inverts the sign of the double-precision real number data in the device specified by (d) and store the inverted data in the device specified by (d).



- The instructions are used to invert positive and negative signs.

Operation error

Error code	Description
3501H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

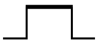
Transferring single-precision real number

EMOV

This instruction transfers single-precision real number data to the specified device.

ST
ENO:=EMOV(EN,s,d);

■Execution condition

Instruction	Execution condition
EMOV	

Setting data

■Description, range, data type

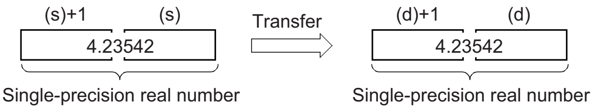
Operand	Description	Range	Data type
(s)	Data to be transferred or start device containing the data to be transferred	0, $2^{-126}\leq|(s)|<2^{128}$	ANYREAL_32
(d)	Start device for storing transferred data	—	ANYREAL_32
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	—	—	○	○	—
(d)	—	—	○	○	—

Processing details

- This instruction transfers 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.

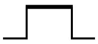
Transferring double-precision real number

EDMOV

This instruction transfers double-precision real number data to the specified device.

ST
ENO:=EDMOV(EN,s,d);

■Execution condition

Instruction	Execution condition
EDMOV	

Setting data

■Description, range, data type

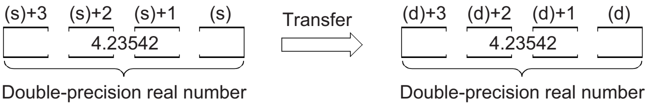
Operand	Description	Range	Data type
(s)	Data to be transferred or start device containing the data to be transferred	0, $2^{-1022}\leq\|(s)\|<2^{1024}$	ANYREAL_64
(d)	Start device for storing transferred data	—	ANYREAL_64
EN	Execution condition	—	BOOL
ENO	Execution result	—	BOOL

■Applicable devices/labels

Operand	Bit		Word		Constant
	SB	RX, RY, LB	SW	G, RWw, RWr, LW	K, H
(s)	—	—	○	—	—
(d)	—	—	○	—	—

Processing details

- This instruction transfers the double-precision real number data stored in the device specified by (s) to the device specified by (d).



Operation error

There is no operation error.

PART 6 STANDARD FUNCTIONS

This part consists of the following chapters.

11 TYPE CONVERSION FUNCTIONS

11.1 Converting BOOL to WORD

BOOL_TO_WORD

This function converts a value from BOOL data type to WORD data type.

ST
d:=BOOL_TO_WORD(s);

Setting data

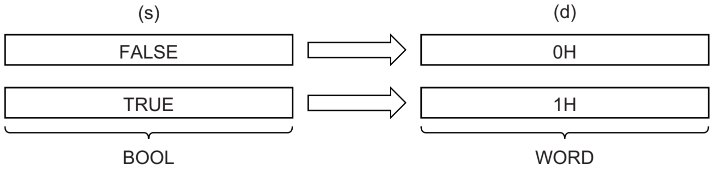
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to WORD data type, and output the converted value from (d).
- When the input value is FALSE, 0H (WORD data type) is output.
- When the input value is TRUE, 1H (WORD data type) is output.



- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.2 Converting BOOL to DWORD

BOOL_TO_DWORD

This function converts a value from BOOL data type to DWORD data type.

Structured text
d:=BOOL_TO_DWORD(s);

Setting data

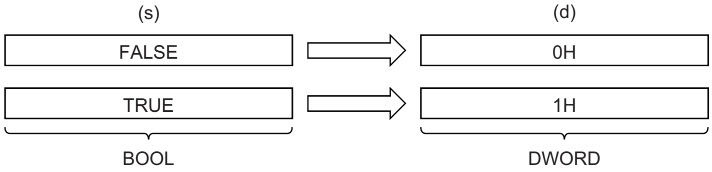
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to DWORD data type, and output the converted value from (d).
- When the input value is FALSE, 0H (DWORD data type) is output.
- When the input value is TRUE, 1H (DWORD data type) is output.



- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.3 Converting BOOL to INT

BOOL_TO_INT

This function converts a value from BOOL data type to INT data type.

Structured text
d:=BOOL_TO_INT(s);

Setting data

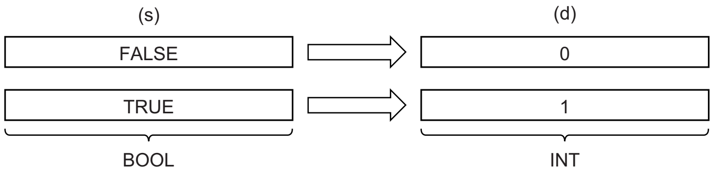
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to INT data type, and output the converted value from (d).
- When the input value is FALSE, 0 (INT data type) is output.
- When the input value is TRUE, 1 (INT data type) is output.



- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.4 Converting BOOL to DINT

BOOL_TO_DINT

This function converts a value from BOOL data type to DINT data type.

Structured text
d:=BOOL_TO_DINT(s);

Setting data

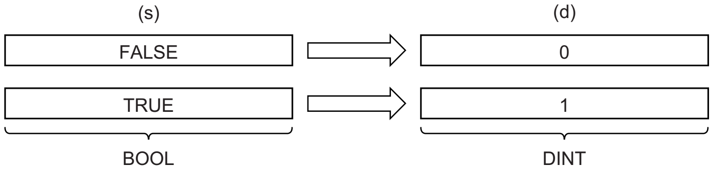
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to DINT data type, and output the converted value from (d).
- When the input value is FALSE, 0 (DINT data type) is output.
- When the input value is TRUE, 1 (DINT data type) is output.



- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.5 Converting BOOL to TIME

BOOL_TO_TIME

This function converts a value from BOOL data type to TIME data type.

Structured text
d:=BOOL_TO_TIME(s);

Setting data

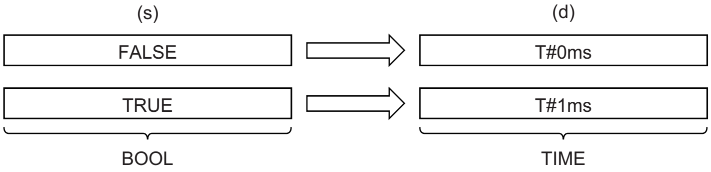
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- This function converts the value input to (s) from BOOL data type to TIME data type, and output the converted value from (d).
- When the input value is FALSE, 0 (TIME data type) is output.
- When the value is TRUE, 1 (TIME data type) is output.



- Input a BOOL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.6 Converting WORD to BOOL

WORD_TO_BOOL

This function converts a value from WORD data type to BOOL data type.

Structured text
d:=WORD_TO_BOOL(s);

Setting data

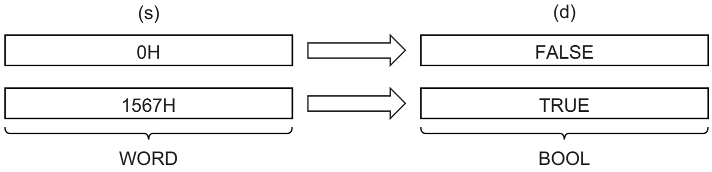
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	WORD
d	Output	Output variable	BOOL

Processing details

■Operation processing

- This function converts the value input to (s) from WORD data type to BOOL data type, and output the converted value from (d).
- When the input value is 0H, FALSE is output.
- When the input value is other than 0H, TRUE is output.



- Input a WORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.7 Converting WORD to DWORD

WORD_TO_DWORD

This function converts a value from WORD data type to DWORD data type.

Structured text
d:=WORD_TO_DWORD(s);

Setting data

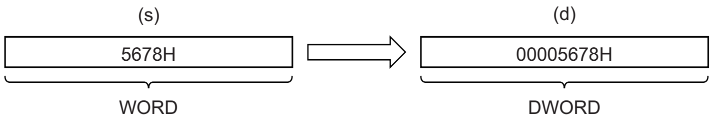
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	WORD
d	Output	Output variable	DWORD

Processing details

■Operation processing

- This function converts the value input to (s) from WORD data type to DWORD data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input a WORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.8 Converting WORD to INT

WORD_TO_INT

This function converts a value from WORD data type to INT data type.

Structured text
d:=WORD_TO_INT(s);

Setting data

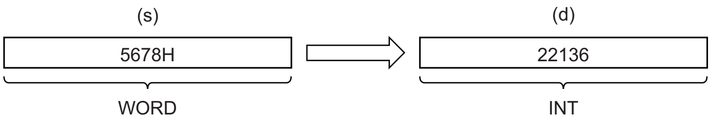
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	WORD
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from WORD data type to INT data type, and output the converted value from (d).



- Input a WORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.9 Converting WORD to DINT

WORD_TO_DINT

This function converts a value from WORD data type to DINT data type.

Structured text
d:=WORD_TO_DINT(s);

Setting data

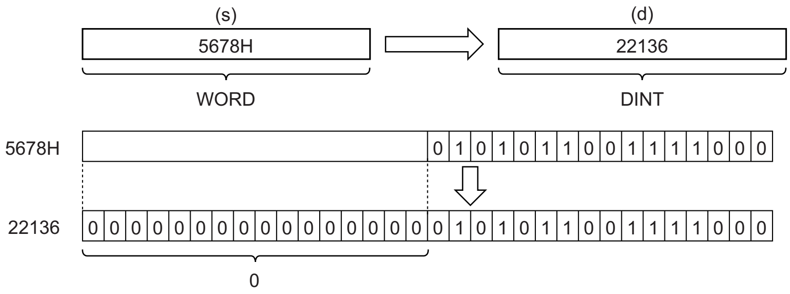
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	WORD
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from WORD data type to DINT data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input a WORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.10 Converting WORD to TIME

WORD_TO_TIME

This function converts a value from WORD data type to TIME data type.

Structured text
d:=WORD_TO_TIME(s);

Setting data

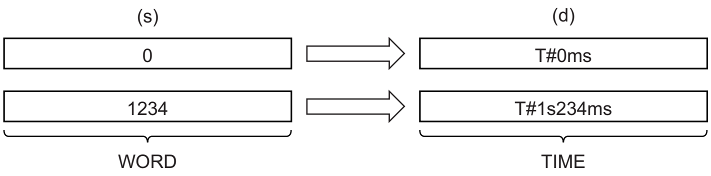
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	WORD
d	Output	Output variable	TIME

Processing details

■Operation processing

- This function converts the value input to (s) from WORD data type to TIME data type, and output the converted value from (d).



- Input a WORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.11 Converting DWORD to BOOL

DWORD_TO_BOOL

This function converts a value from DWORD data type to BOOL data type.

Structured text
d:=DWORD_TO_BOOL(s);

Setting data

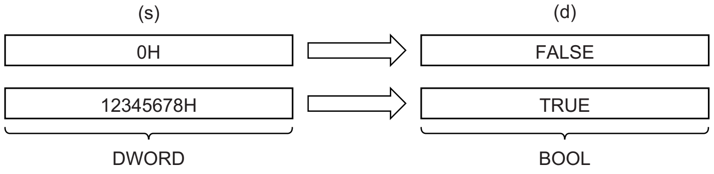
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DWORD
d	Output	Output variable	BOOL

Processing details

■Operation processing

- This function converts the value input to (s) from DWORD data type to BOOL data type, and output the converted value from (d).
- When the input value is 0H, FALSE is output.
- When the input value is other than 0H, TRUE is output.



- Input a DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.12 Converting DWORD to WORD

DWORD_TO_WORD

This function converts a value from DWORD data type to WORD data type.

Structured text
d:=DWORD_TO_WORD(s);

Setting data

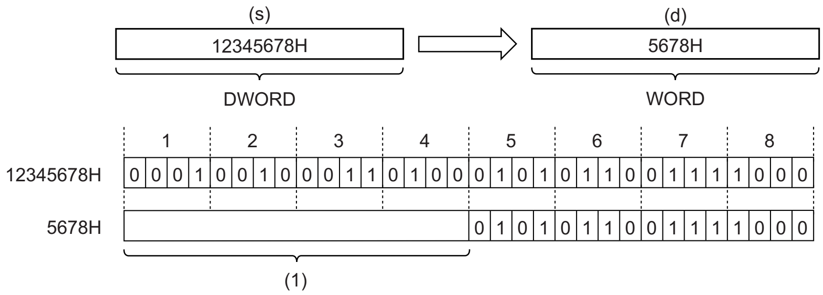
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DWORD
d	Output	Output variable	WORD

Processing details

■Operation processing

- This function converts the value input to (s) from DWORD data type to WORD data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DWORD data type) are discarded. (Refer to (1) in the figure below.)



- Input a DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Point

When the DWORD_TO_WORD function is executed, the upper 16-bit data of the input value (DWORD data type) are discarded.

Operation error

There is no operation error.

11.13 Converting DWORD to INT

DWORD_TO_INT

This function converts a value from DWORD data type to INT data type.

Structured text
d:=DWORD_TO_INT(s);

Setting data

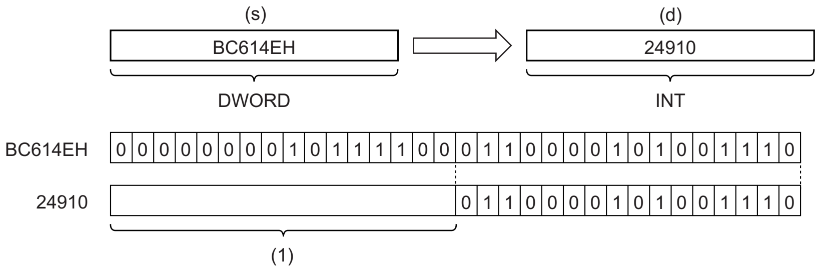
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DWORD
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from DWORD data type to INT data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DWORD data type) are discarded. (Refer to (1) in the figure below.)



- Input a DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Point

When the DWORD_TO_INT function is executed, the upper 16-bit data of the input value (DWORD data type) are discarded.

Operation error

There is no operation error.

11.14 Converting DWORD to DINT

DWORD_TO_DINT

This function converts a value from DWORD data type to DINT data type.

Structured text
d:=DWORD_TO_DINT(s);

Setting data

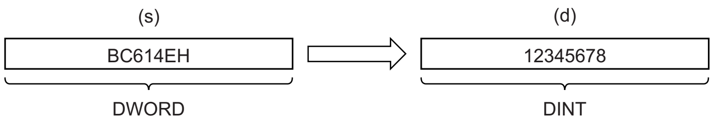
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DWORD
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from DWORD data type to DINT data type, and output the converted value from (d).



- Input a DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.15 Converting DWORD to TIME

DWORD_TO_TIME

This function converts a value from DWORD data type to TIME data type.

Structured text
d:=DWORD_TO_TIME(s);

Setting data

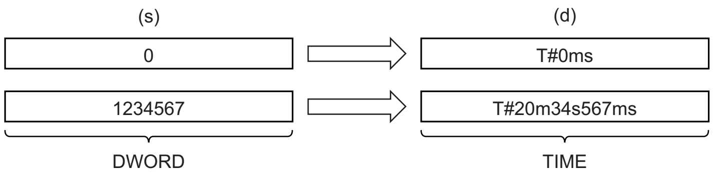
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DWORD
d	Output	Output variable	TIME

Processing details

■Operation processing

- This function converts the value input to (s) from DWORD data type to TIME data type, and output the converted value from (d).



- Input a DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.16 Converting INT to BOOL

INT_TO_BOOL

This function converts a value from INT data type to BOOL data type.

Structured text
d:=INT_TO_BOOL(s);

Setting data

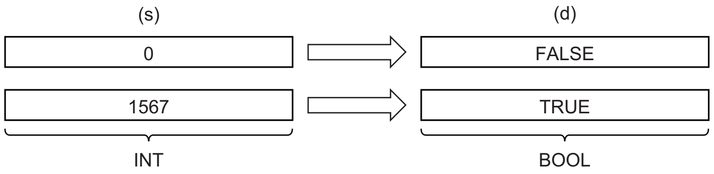
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	BOOL

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to BOOL data type, and output the converted value from (d).
- When the value 0 is input, FALSE is output.
- When the value other than 0 is input, TRUE is output.



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.17 Converting INT to WORD

INT_TO_WORD

This function converts a value from INT data type to WORD data type.

Structured text
d:=INT_TO_WORD(s);

Setting data

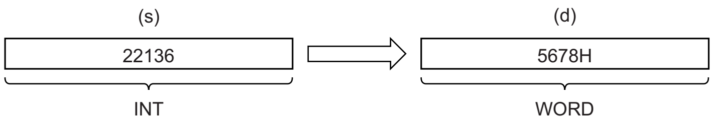
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	WORD

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to WORD data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.18 Converting INT to DWORD

INT_TO_DWORD

This function converts a value from INT data type to DWORD data type.

Structured text
d:=INT_TO_DWORD(s);

Setting data

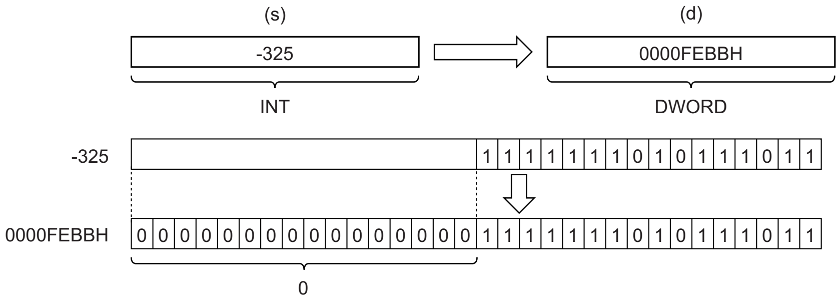
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	DWORD

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to DWORD data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.19 Converting INT to DINT

INT_TO_DINT

This function converts a value from INT data type to DINT data type.

Structured text
d:=INT_TO_DINT(s);

Setting data

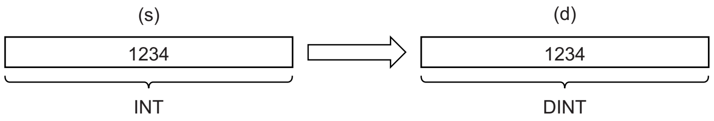
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to DINT data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.20 Converting INT to REAL

INT_TO_REAL

This function converts a value from INT data type to REAL data type.

Structured text
d:=INT_TO_REAL(s);

Setting data

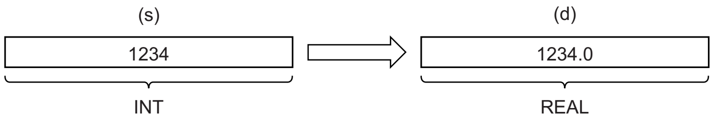
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	REAL

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to REAL data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.21 Converting INT to LREAL

INT_TO_LREAL

This function converts a value from INT data type to LREAL data type.

Structured text
d:=INT_TO_LREAL(s);

Setting data

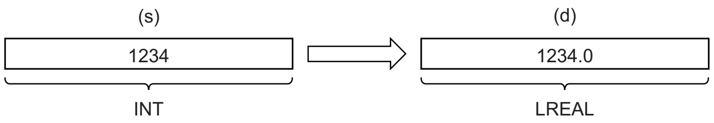
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	LREAL

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to LREAL data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.22 Converting INT to TIME

INT_TO_TIME

This function converts a value from INT data type to TIME data type.

Structured text
d:=INT_TO_TIME(s);

Setting data

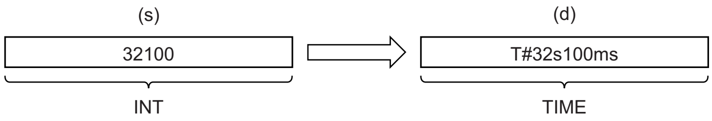
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	INT
d	Output	Output variable	TIME

Processing details

■Operation processing

- This function converts the value input to (s) from INT data type to TIME data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.23 Converting DINT to BOOL

DINT_TO_BOOL

This function converts a value from DINT data type to BOOL data type.

Structured text
d:=DINT_TO_BOOL(s);

Setting data

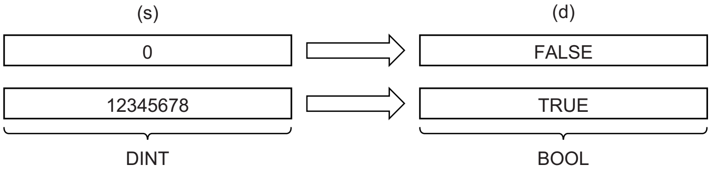
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	BOOL

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to BOOL data type, and output the converted value from (d).
- When the value 0 is input, FALSE is output.
- When the value other than 0 is input, TRUE is output.



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.24 Converting DINT to WORD

DINT_TO_WORD

This function converts a value from DINT data type to WORD data type.

Structured text
d:=DINT_TO_WORD(s);

Setting data

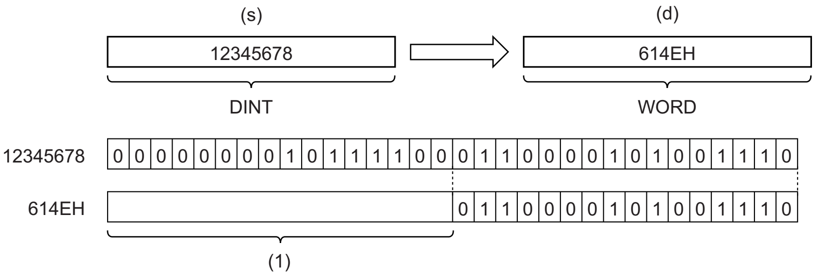
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	WORD

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to WORD data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DINT data type) are discarded. (Refer to (1) in the figure below.)



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Point

When the DINT_TO_WORD function is executed, the upper 16-bit data of the input value (DINT data type) are discarded.

Operation error

There is no operation error.

11.25 Converting DINT to DWORD

DINT_TO_DWORD

This function converts a value from DINT data type to DWORD data type.

Structured text
d:=DINT_TO_DWORD(s);

Setting data

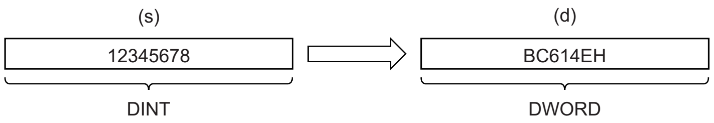
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	DWORD

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to DWORD data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.26 Converting DINT to INT

DINT_TO_INT

This function converts a value from DINT data type to INT data type.

Structured text
d:=DINT_TO_INT(s);

Setting data

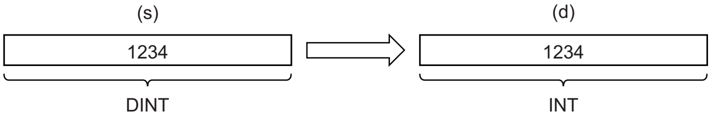
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to INT data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3500H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.

11.27 Converting DINT to REAL

DINT_TO_REAL

This function converts a value from DINT data type to REAL data type.

Structured text
d:=DINT_TO_REAL(s);

Setting data

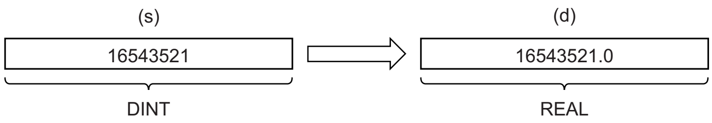
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	REAL

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to REAL data type, and output the converted value from (d).



- Input a DINT data type value to (s).
- The number of significant digits is about seven because a REAL data type value is processed in 32-bit single precision.
- If the integer value exceeds the range of -16777216 to 16777215, a rounding error occurs in the converted value.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.28 Converting DINT to LREAL

DINT_TO_LREAL

This function converts a value from DINT data type to LREAL data type.

Structured text
d:=DINT_TO_LREAL(s);

Setting data

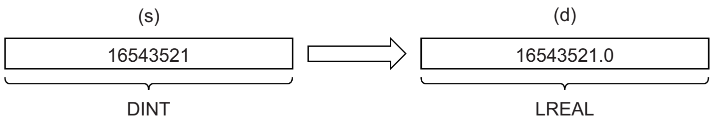
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	LREAL

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to LREAL data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.29 Converting DINT to TIME

DINT_TO_TIME

This function converts a value from DINT data type to TIME data type.

Structured text
d:=DINT_TO_TIME(s);

Setting data

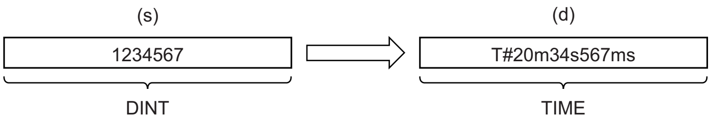
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	DINT
d	Output	Output variable	TIME

Processing details

■Operation processing

- This function converts the value input to (s) from DINT data type to TIME data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.30 Converting REAL to INT

REAL_TO_INT

This function converts a value from REAL data type to INT data type.

Structured text
d:=REAL_TO_INT(s);

Setting data

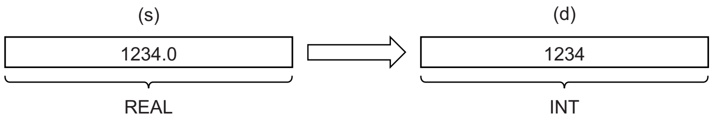
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	REAL
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from REAL data type to INT data type, and output the converted value from (d).



- Input a REAL data type value to (s) within the range of -32768 to 32767.
- After conversion, the first digit after the decimal point of the input value (REAL data type) is rounded off.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3500H	The single-precision real number input to (s) is out of the range, -32768 to 32767.
3501H	• An unusual number is input to (s). • The single-precision real number input to (s) is not within the following range: $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

11.31 Converting REAL to DINT

REAL_TO_DINT

This function converts a value from REAL data type to DINT data type.

Structured text
d:=REAL_TO_DINT(s);

Setting data

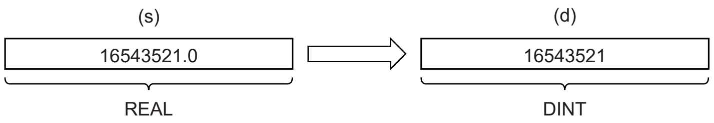
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	REAL
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from REAL data type to DINT data type, and output the converted value from (d).



- Input a REAL data type value to (s) within the range of -2147483648 to 2147483647.
- After conversion, the first digit after the decimal point of the input value (REAL data type) is rounded off.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3500H	The single-precision real number input to (s) is out of the range, -2147483648 to 2147483647.
3501H	• An unusual number is input to (s). • The single-precision real number input to (s) is not within the following range: $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

11.32 Converting REAL to LREAL

REAL_TO_LREAL

This function converts a value from REAL data type to LREAL data type.

Structured text
d:=REAL_TO_LREAL(s);

Setting data

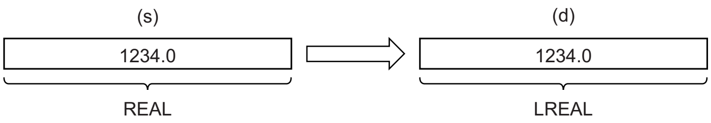
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	REAL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- This function converts the value input to (s) from REAL data type to LREAL data type, and output the converted value from (d).



- Input a REAL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

11.33 Converting LREAL to INT

LREAL_TO_INT

This function converts a value from LREAL data type to INT data type.

Structured text
d:=LREAL_TO_INT(s);

Setting data

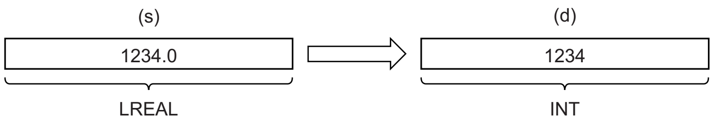
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	LREAL
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from LREAL data type to INT data type, and output the converted value from (d).



- Input an LREAL data type value to (s).
- After conversion, the first digit after the decimal point of the input value (LREAL data type) is rounded off.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data input to (s) is -0 or out of the following range. $-2^{1024}<$(s), (d)$\leq-2^{-1022}$, 0, $2^{-1022}\leq$(s), (d)$<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The data input to (s) is other than -32768 to 32767.

11.34 Converting LREAL to DINT

LREAL_TO_DINT

This function converts a value from LREAL data type to DINT data type.

Structured text
d:=LREAL_TO_DINT(s);

Setting data

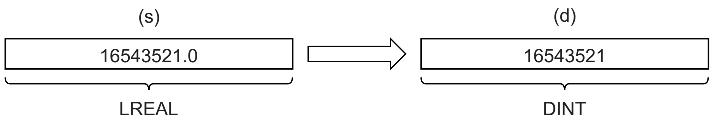
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	LREAL
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from LREAL data type to DINT data type, and output the converted value from (d).



- Input an LREAL data type value to (s).
- After conversion, the first digit after the decimal point of the input value (LREAL data type) is rounded off.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data input to (s) is -0 or out of the following range. -2^{1024}<(s), (d)≤-2^{-1022}, 0, 2^{-1022}≤(s), (d)<2^{1024} (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The data input to (s) is other than -2147483648 to 2147483647.

11.35 Converting LREAL to REAL

LREAL_TO_REAL

This function converts a value from LREAL data type to REAL data type.

Structured text
d:=LREAL_TO_REAL(s);

Setting data

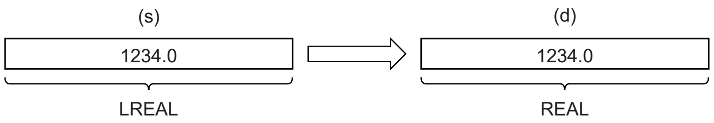
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	LREAL
d	Output	Output variable	REAL

Processing details

■Operation processing

- This function converts the value input to (s) from LREAL data type to REAL data type, and output the converted value from (d).



- Input an LREAL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)|<2^{128}$

11.36 Converting TIME to BOOL

TIME_TO_BOOL

This function converts a value from TIME data type to BOOL data type.

Structured text
d:=TIME_TO_BOOL(s);

Setting data

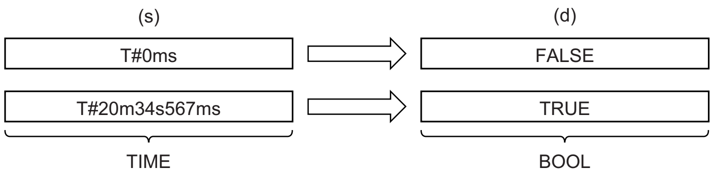
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	TIME
d	Output	Output variable	BOOL

Processing details

■Operation processing

- This function converts the value input to (s) from TIME data type to BOOL data type, and output the converted value from (d).



■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.37 Converting TIME to WORD

TIME_TO_WORD

This function converts a value from TIME data type to WORD data type.

Structured text
d:=TIME_TO_WORD(s);

Setting data

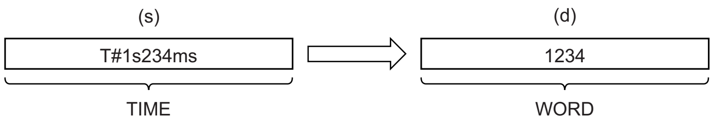
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	TIME
d	Output	Output variable	WORD

Processing details

■Operation processing

- This function converts the value input to (s) from TIME data type to WORD data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.38 Converting TIME to DWORD

TIME_TO_DWORD

This function converts a value from TIME data type to DWORD data type.

Structured text
d:=TIME_TO_DWORD(s);

Setting data

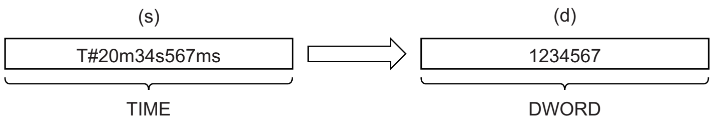
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	TIME
d	Output	Output variable	DWORD

Processing details

■Operation processing

- This function converts the value input to (s) from TIME data type to DWORD data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.39 Converting TIME to INT

TIME_TO_INT

This function converts a value from TIME data type to INT data type.

Structured text
d:=TIME_TO_INT(s);

Setting data

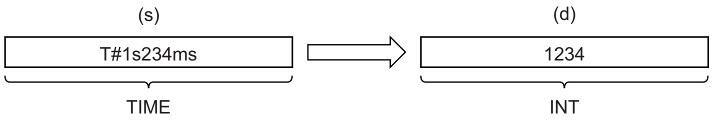
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	TIME
d	Output	Output variable	INT

Processing details

■Operation processing

- This function converts the value input to (s) from TIME data type to INT data type, and output the converted value from (d).



- Input a TIME data type value to (s).
- The upper 16-bit data of the input value (TIME data type) are discarded.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

11.40 Converting TIME to DINT

TIME_TO_DINT

This function converts a value from TIME data type to DINT data type.

Structured text
d:=TIME_TO_DINT(s);

Setting data

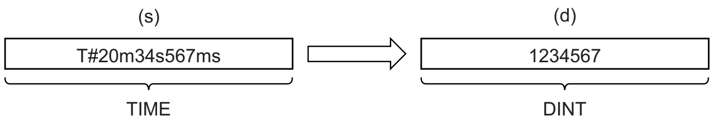
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	TIME
d	Output	Output variable	DINT

Processing details

■Operation processing

- This function converts the value input to (s) from TIME data type to DINT data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

12 SINGLE VARIABLE FUNCTIONS

12.1 Calculating the Absolute Value

ABS

This function outputs the absolute value of an input value.

Structured text
d:=ABS(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_NUM
d	Output	Output variable	ANY_NUM

Processing details

■Operation processing

- This function outputs the absolute value of the INT, DINT, REAL, or LREAL data type value input to (s), in the same type of data as (s), from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B = |A|

- Input an INT, DINT, REAL, or LREAL data type value to (s).
- If -32768 in INT data type is input to (s), (d) will output -32768.
- If -2147483648 in DINT data type is input to (s), (d) will output -2147483648. (No operation error occurs.)

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s) is of REAL data type

Error code	Description
3501H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

- When (s) is of LREAL data type

Error code	Description
3501H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

12.2 Calculating the Square Root

SQRT

These functions calculate the square root of an input value.

Structured text
d:=SQRT(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the square root of the REAL/LREAL data type value input to (s) and store the operation result in (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B = \sqrt{A}$

- Input a positive REAL/LREAL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3507H	The input value is negative.

12.3 Calculating the Natural Logarithm

LN

These functions output the natural logarithm (logarithm with base e) of an input value.

Structured text
d:=LN(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the natural logarithm of the REAL/LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=\log_{e}A$

- Natural logarithm operation is performed with the base (e) defined as 2.71828.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3507H	The input value is 0 or negative.

12.4 Calculating the Common Logarithm

LOG

These functions output the common logarithm (logarithm with base 10) of an input value.

Structured text
d:=LOG(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the common logarithm of the REAL or LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=\log_{10}A$

- Input a REAL or LREAL data type value to (s).
- Input a positive value only. (Calculation cannot be performed with a negative value.)
- If the operation result is -0 or an underflow occurs, 0 will be output as the operation result.

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s) is of REAL data type

Error code	Description
3501H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3507H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

- When (s) is of LREAL data type

Error code	Description
3501H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3507H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

12.5 Calculating the Exponent

EXP

These functions output the exponent of an input value.

Structured text
d:=EXP(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the exponent of the REAL/LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=e^A$

- Exponent operation is performed with the base (e) defined as 2.71828.
- Input a REAL or LREAL data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)|<2^{128}$

12.6 Calculating the Sine

SIN

These functions output the sine of an input value.

Structured text
d:=SIN(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the sine of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=SIN A

- Input a REAL data type value to (s). Input a value (angle) in radians (angle×π/180).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.

12.7 Calculating the Cosine

COS

These functions output the cosine of an input value.

Structured text
d:=COS(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the cosine of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=COS A

- Input a REAL data type value to (s). Input a value (angle) in radians (angle×π/180).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.

12.8 Calculating the Tangent

TAN

These functions output the tangent of an input value.

Structured text
d:=TAN(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the tangent of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=TAN A

- Note that even if the input value is $\pi/2$ radian or $(3/2)\pi$ radian, no error will be issued because of the truncation error in the radian value.
- Input a REAL data type value to (s). Input a value (angle) in radians (angle$\times\pi/180$).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.

12.9 Calculating the Arc Sine

ASIN

These functions output the arc sine (SIN^{-1}) of an input value.

Structured text
d:=ASIN(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the arc sine (SIN^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=SIN^{-1} A$

- Input a REAL data type value to (s) within the following range.

ASIN: -1.0 to 1.0

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.
3507H	The value input with ASIN is other than -1.0 to 1.0.

12.10 Calculating the Arc Cosine

ACOS

These functions output the arc cosine (COS^{-1}) of an input value.

Structured text
d:=ACOS(s);

Setting data

■**Description, type, data type**

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■**Operation processing**

- These functions calculate the arc cosine (COS^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=COS^{-1} A$

- Input a REAL data type value to (s) within the following range.

ACOS: -1.0 to 1.0

- The value (angle) is output from (d) in radians (angle×π/180).

■**Operation result**

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.
3507H	The value input with ACOS is other than -1.0 to 1.0.

12.11 Calculating the Arc Tangent

ATAN

These functions output the arc tangent (TAN^{-1}) of an input value.

Structured text
d:=ATAN(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_REAL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the arc tangent (TAN^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=TAN^{-1} A

- Input a REAL data type value to (s) within the following range.

ATAN: $\pm1.17549^{-38}$ to $\pm3.40282^{+38}$

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3501H	The data specified by (s) is -0.

13 ARITHMETIC OPERATION FUNCTIONS

13.1 Addition

ADD

This function outputs the sum of input values ((s1)+(s2)+···+(s28)).

Structured text[*1]
d:=ADD(s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
d	Output	Output variable	ANY_NUM

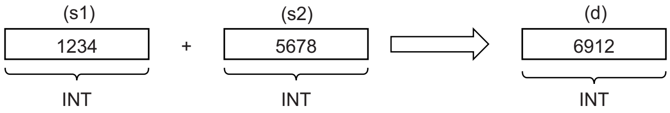
Processing details

■Operation processing

- This function adds the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) to (s28) ((s1)+(s2)+···+(s28)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) to (s28).

- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT	Even if an underflow or overflow occurs, no operation error is issued. [Example 1] 32767+2=-32767 (7FFFH)+(0002H)=(8001H) A negative value results because the most significant bit is 1. [Example 2] -32767+(-2)=32766 (8000H)+(FFFEH)=(7FFEH) A positive value results because the most significant bit is 0.
DINT	Even if an underflow or overflow occurs, no operation error is issued. [Example 1] 2147483647+2=-2147483647 (7FFFFFFFH)+(00000002H)=(80000001H) A negative value results because the most significant bit is 1. [Example 2] -2147483648+(-2)=2147483646 (80000000H)+(FFFEH)=(7FFFFFFEH) A positive value results because the most significant bit is 0.
WORD	Even if an overflow occurs, no operation error is issued. [Example] 65535 + 1 = 0 (FFFFH) + (0001H) = (0000H)
DWORD	Even if an overflow occurs, no operation error is issued. [Example] 4294967295 + 1 = 0 (FFFFFFFFH) + (00000001H) = (00000000H)
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s1) to (s28) are of REAL data type

Error code	Description
3501H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

- When (s1) to (s28) are of LREAL data type

Error code	Description
3501H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

13.2 Multiplication

MUL

This function outputs the product of input values ((s1)×(s2)×···×(s28)).

Structured text[*1]
d:=MUL(s1,s2);

[*1] The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
d	Output	Output variable	ANY_NUM

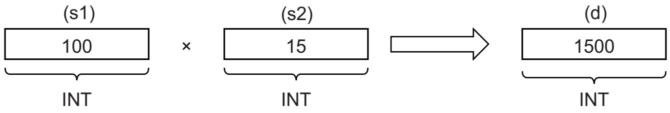
Processing details

■Operation processing

- This function multiplies the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) to (s28) ((s1)×(s2)×···×(s28)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) to (s28).
- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT WORD	• Even if an underflow or overflow occurs, no operation error is issued. • Even if the operation result is outside the INT or WORD data type range, the INT or WORD data type value is output; (In this case, the output value is of INT or WORD data type with the upper 16 bits deleted although the operation result is a DINT or DWORD data type value.) • If the operation result is outside the INT or WORD data type range, convert the input value to the DINT or DWORD data type by using the INT_TO_DINT or WORD_TO_DWORD function, and then perform operation.
DINT DWORD	• Even if an underflow or overflow occurs, no operation error is issued. • Even if the operation result is outside the DINT or DWORD data type range, the DINT or DWORD data type value is output; (In this case, the output value is of DINT or DWORD data type with the upper 32 bits deleted although the operation result is 64-bit data.) • If the operation result is outside the DINT or DWORD data type range, convert the input value to the REAL data type by using the DINT_TO_REAL function, and then perform operation.
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

The operation processing is performed. The operation result is output from (d).

Point

If the operation result is outside the data type range, convert the input value as appropriate before operation.

Operation error

- When (s1) to (s28) are of REAL data type

Error code	Description
3501H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

- When (s1) to (s28) are of LREAL data type

Error code	Description
3501H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

13.3 Subtraction

SUB

This function outputs the difference between input values ((s1)-(s2)).

Structured text
d:=SUB(s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1)	Input	Input variable	ANY_NUM
s2 (IN2)	Input	Input variable	ANY_NUM
d	Output	Output variable	ANY_NUM

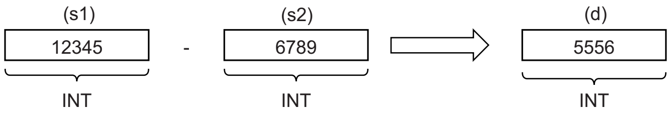
Processing details

■Operation processing

- This function performs subtraction between the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) and (s2) ((s1)-(s2)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) and (s2).
- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT	Even if an underflow or overflow occurs, no operation error is issued. [Example 1] 32767-(-2)=-32767 (7FFFH)-(FFFEH)=(8001H) A negative value results because the most significant bit is 1. [Example 2] -32767-2=32766 (8000H)-(0002H)=(7FFEH) A positive value results because the most significant bit is 0.
DINT	Even if an underflow or overflow occurs, no operation error is issued. [Example 1] 2147483647-(-2)=-2147483647 (7FFFFFFFH)-(0000FFFEH)=(80000001H) A negative value results because the most significant bit is 1. [Example 2] -2147483648-2=2147483646 (80000000H)-(00000002H)=(7FFFFFFEH) A positive value results because the most significant bit is 0.
WORD	Even if an underflow occurs, no operation error is issued. [Example] 0 - 1 = 65535 (0000H) - (0001H) = (FFFFH)
DWORD	Even if an underflow occurs, no operation error is issued. [Example] 0 - 1 = 4294967295 (00000000H) - (00000001H) = (FFFFFFFFH)
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s1) and (s2) are of REAL data type

Error code	Description
3501H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

- When (s1) and (s2) are of LREAL data type

Error code	Description
3501H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

13.4 Division

DIV

This function outputs the quotient of input values ((s1)÷(s2)).

Structured text
d:=DIV(s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1)	Dividend	Input variable	ANY_NUM
s2 (IN2)	Divisor	Input variable	ANY_NUM
d	Output	Output variable	ANY_NUM

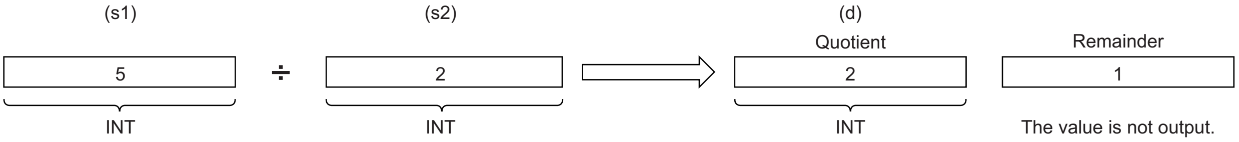
Processing details

■Operation processing

- This function performs division between the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) and (s2) ((s1)÷(s2)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) and (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s1) and (s2) are of INT or WORD data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of REAL data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.
3501H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

- When (s1) and (s2) are of LREAL data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.
3501H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3502H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

13.5 Remainder

MOD

This function outputs the remainder of input values ((s1)÷(s2)).

Structured text
The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design))

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1)	Dividend	Input variable	ANY_INT
s2 (IN2)	Divisor	Input variable	ANY_INT
d	Output	Output variable	ANY_INT

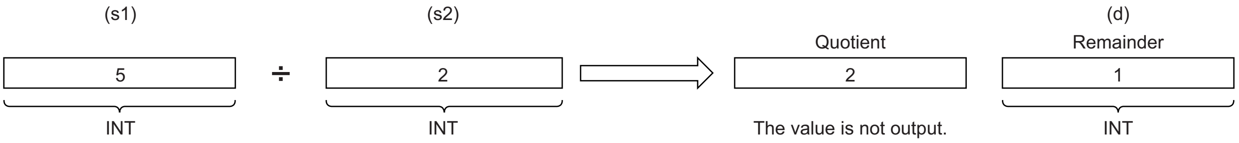
Processing details

■Operation processing

- This function performs division between the INT, DINT, WORD, or DWORD data type values input to (s1) and (s2) ((s1)÷(s2)), and output the remainder of the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, or DWORD data type value to (s1) and (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

- When (s1) and (s2) are of INT or WORD data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

Error code	Description
34FFH	The value (divisor) input to (s2) is 0.

13.6 Assignment (Move Operation)

MOVE

This function outputs the assignment value of an input value.

Structured text
d:=MOVE(s);

Setting data

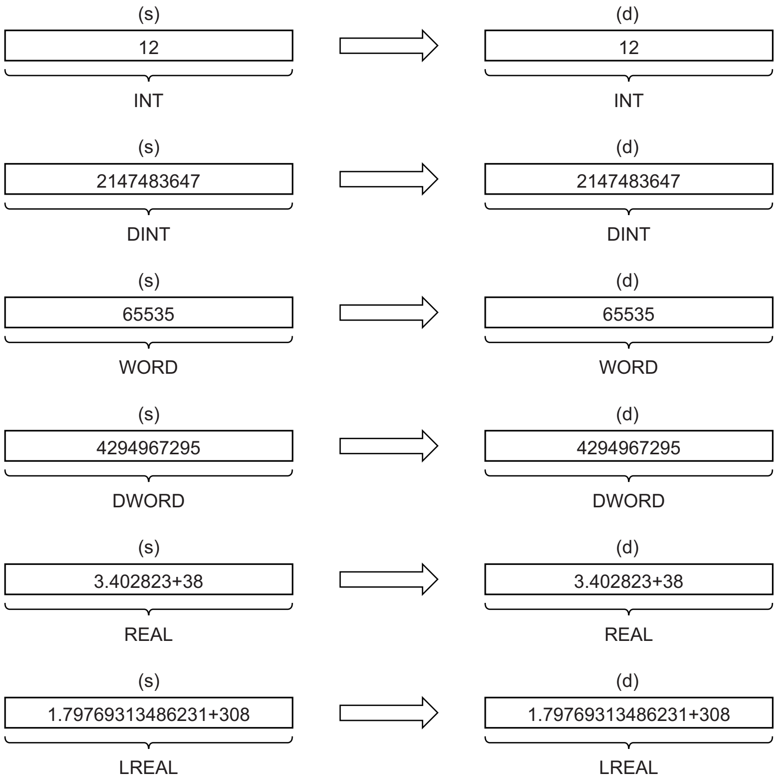
■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY
d	Output	Output variable	ANY

Processing details

■Operation processing

- This function assigns the value of the input variable specified by (s) to the output variable specified by (d).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s) and (d). The values input to (s) and (d) must be of the same data type.



■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3506H	There is no NULL code (00H) in the setting area specified by (s) in the device/label memory.
3507H	The number of characters in the string input to (s) exceeds 16383.
3508H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

14 BOOLEAN FUNCTIONS

14.1 NOT Operation

NOT

This function outputs the logical NOT of input values.

Structured text
The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design))

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Input	Input variable	ANY_BIT
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

- This function performs a NOT operation (bit-by-bit) on the BOOL, WORD, or DWORD data type value input to (s), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: 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

- Input a BOOL, WORD, or DWORD data type value to (s).

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

15 SELECTION FUNCTIONS

15.1 Selecting the Maximum/Minimum Value

MAX, MIN

- MAX: This function outputs the maximum input value.
- MIN: This function outputs the minimum input value.

Structured text[*1]

d:=MAX(s1,s2);
d:=MIN(s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_ELEMENTARY
d	Output	Output variable	ANY_ELEMENTARY

Processing details

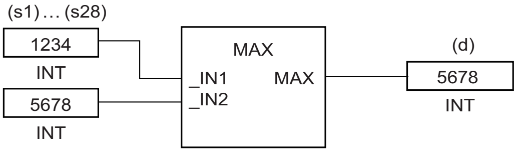
■Operation processing

- MAX

This function outputs the maximum value of the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type values input to (s1) to (s28), in the same data type as (s), from (d).

Ex.

Data type: INT

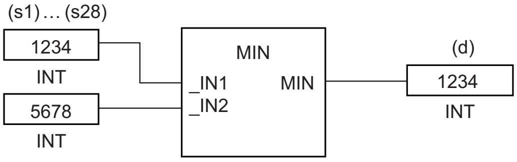


- MIN

This function outputs the minimum value of the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type values input to (s1) to (s28), in the same data type as (s), from (d).

Ex.

Data type: INT



- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value to (s1) to (s28).
- Conditions for comparing the STRING data type values are as follows:

Match:	• All characters matched
Bigger string:	• The one having a character with a bigger code (when strings consist of different characters) • The one having a longer length (when strings are of different lengths)
Smaller string:	• The one having a character with a smaller code (when strings consist of different characters) • The one having a shorter length (when strings are of different lengths)

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

Error code	Description
3506H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
3507H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3508H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

MEMO

PART 7 STANDARD FUNCTION BLOCKS

This part consists of the following chapters.

16 BISTABLE FUNCTION BLOCKS

16.1 Bistable Function Block (Set-Dominant)

SR

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Structured text
Instance name(S1:=s1,R:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (S1)	Set command	Input variable	BOOL
s2 (R)	Reset command	Input variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

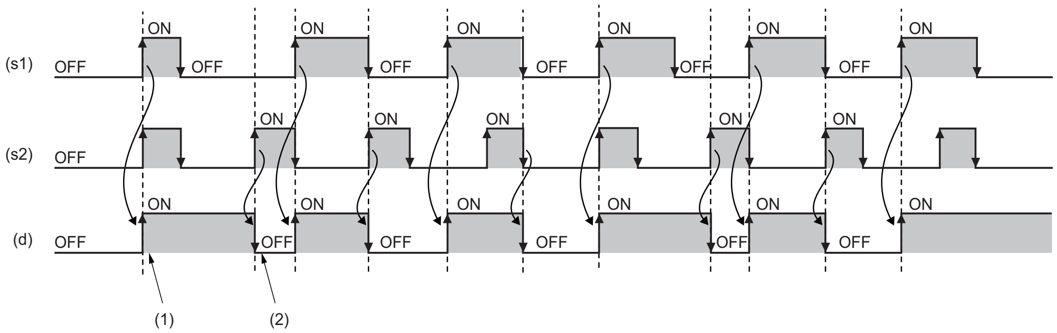
■Operation processing

- When (s1) turns on, (d) is set. Turning on (s2) while (s1) is off resets (d).
- Even when (s2) turns on while (s1) is on, (d) is not reset.

■Operation result

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) When (s1) turns on, (d) turns on.

(2) When (s2) turns on while (s1) is off, (d) turns off.

Operation error

There is no operation error.

16.2 Bistable Function Block (Reset-Dominant)

RS

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Structured text
Instance name(S:=s1,R1:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (S)	Set command	Input variable	BOOL
s2 (R1)	Reset command	Input variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

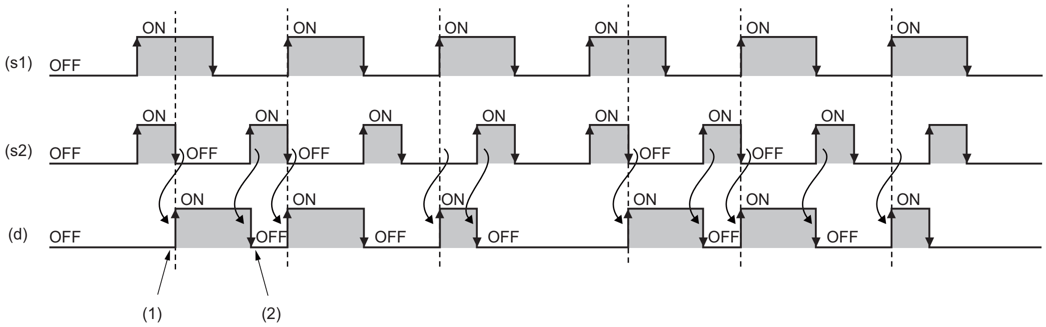
■Operation processing

- When (s1) turns on, (d) is set. When (s2) turns on, (d) is reset.
- Even when (s1) turns on while (s2) is on, (d) is not set.

■Operation result

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) When (s2) turns off while (s1) is on, (d) turns on.

(2) When (s2) turns on, (d) turns off.

Operation error

There is no operation error.

MEMO

17 EDGE DETECTION FUNCTION BLOCKS

17.1 Detecting a Rising Edge

R_TRIG

These function blocks detect a signal rising edge, and outputs the pulse signal.

Structured text
Instance name(CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (CLK)	Rising edge detection input	Input variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

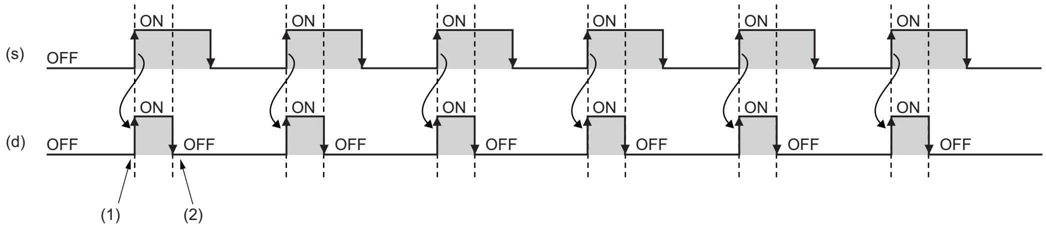
■Operation processing

When (s) turns on, (d) turns on only for one scan.

■Operation result

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) (d) turns on at the rising edge of (s).

(2) (d) turns off in the next scan.

Operation error

There is no operation error.

17.2 Detecting a Falling Edge

F_TRIG

These function blocks detect a signal falling edge, and outputs the pulse signal.

Structured text
Instance name(CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (CLK)	Falling edge detection input	Input variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

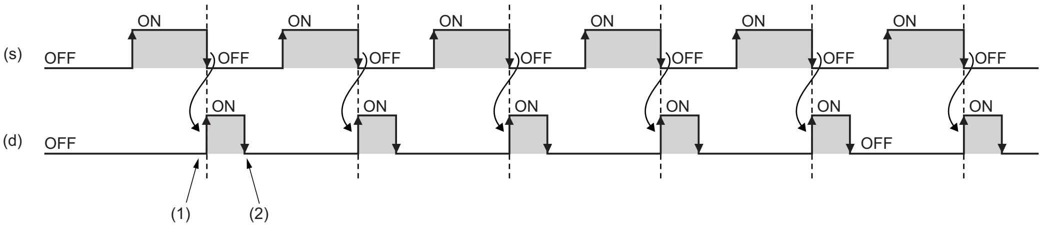
■Operation processing

When (s) turns off, (d) turns on only for one scan.

■Operation result

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) (d) turns on at the falling edge of (s).

(2) (d) turns off in the next scan.

Operation error

There is no operation error.

18 TIMER FUNCTION BLOCKS

18.1 Pulse Timer

TP

These function blocks keep the signal on for the specified period of time.

Structured text
Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Start of output	Input variable	BOOL
n (PT)	Output time setting value	Input variable	TIME
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 period of time set by (n). The time elapsed after (d1) turns on is set to (d2).
- Use the long timer to count the elapsed time.

2. End of output

- Once the elapsed time reaches the setting time, (d1) turns off.
- If (s) is off after (d1) turns off, the elapsed time is reset.
- Even when (s) turns off while (d1) is on, (d1) does not turn off.

3. Output time setting

The valid setting range of (n) is T#1 ms to T#2147483 ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1 ms, the minimum value will be 1 ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the output time setting value is of time type (32-bit value). • Output time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001 ms: T#1 ms to T#2147483 ms • If the long timer setting value is 1000 ms: T#1000 ms to T#2147483000 ms

The value at the rising edge (off to on) of (d1) is used for the setting value of (n). When the (n) value is changed when (d1) is on, the new value will be enabled at the next output start timing.

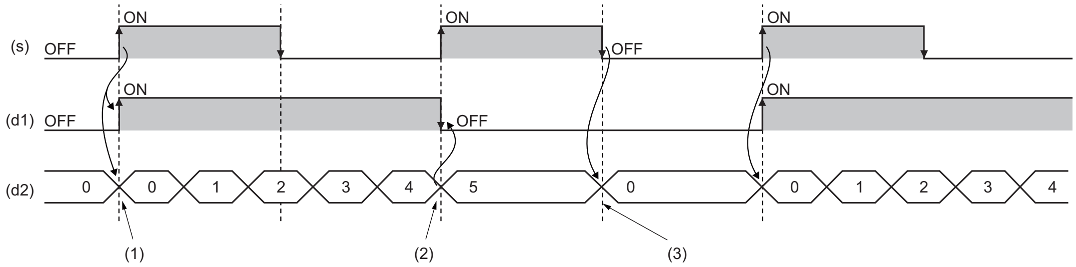
■Operation result

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns on, (d1) turns on. When (s) turns on, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns off.

(3) When both (s) and (d1) are off, the value in (d2) is initialized.

Operation error

Error code	Description
3500H	The output time setting value exceeds the valid range.

18.2 On Delay Timer

TON

These function blocks turn on a signal after the specified period of time.

Structured text
Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
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 after the time that was set by (n). The delay time elapsed after (d1) turns on is set to (d2).
- When (s) turns off, (d1) turns off and the delay elapsed time is also reset.
- Use the long timer to count the elapsed time.

2. Delay time setting

The valid setting range of (n) is T#1 ms to T#2147483 ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1 ms, the minimum value will be 1 ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the delay time setting value is of time type (32-bit value). • Delay time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value of in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001 ms: T#1ms to T#2147483 ms • If the long timer setting value is 1000 ms: T#1000 ms to T#2147483000 ms

The value at the rising edge (off to on) of (d) is used for the setting value of (n). When the (n) value is changed while (s) is on, the new value will be enabled at the next rising edge of (s).

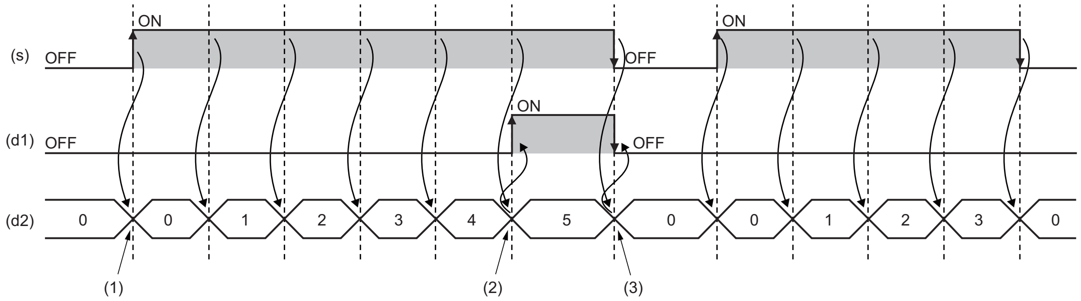
■Operation result

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns on, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns on.

(3) When both (s) and (d1) turn off, the value in (d2) is initialized.

Operation error

Error code	Description
3500H	The output time setting value exceeds the valid range.

18.3 Off Delay Timer

TOF

These function blocks turn off a signal after the specified period of time.

Structured text
Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
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.
- When (s) changes from on to off, (d1) turns off after the time that was set by (n). The delay time elapsed after (d1) turns off is set to (d2).
- Use the long timer to count the elapsed time.

2. Delay time setting

The valid setting range of (n) is T#1 ms to T#2147483 ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1 ms, the minimum value will be 1 ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the delay time setting value is of time type (32-bit value). • Delay time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value of in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001 ms: T#1ms to T#2147483 ms • If the long timer setting value is 1000 ms: T#1000 ms to T#2147483000 ms

The value at the falling edge (on to off) of (s) is used for the setting value of (n). When the (n) value is changed when (s) is off, the new value will be enabled at the next falling edge of (s).

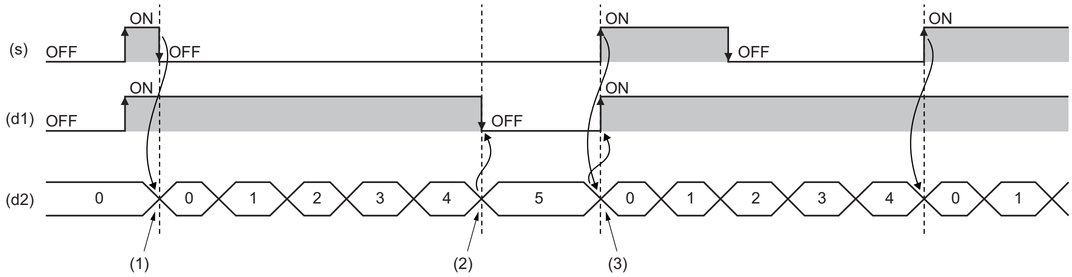
■Operation result

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns off, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns on.

(3) When (s) turns on, the value in (d2) is initialized.

Operation error

Error code	Description
3500H	The output time setting value exceeds the valid range.

18.4 Timer Function Block

TIMER_□_M

These function blocks start counting a timer when the execution condition is satisfied, and continue counting until the timer reaches the set value.

Structured text
Instance name(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (Coil)	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s2 (Preset)	Timer setting value	Input variable	INT
s3 (ValueIn)	Initial timer value	Input variable	INT
d1 (ValueOut)	Current timer value	Output variable	INT
d2 (Status)	Output	Output variable	BOOL

Processing details

■TIMER_10_FB_M

- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×10 ms. When the value reaches (s2)×10 ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- If the unit of measurement of the high-speed timer (in the timer limit setting) is changed from the default value using the engineering tool, a warning will be issued during conversion of modified or newly added programs or all programs in a project.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

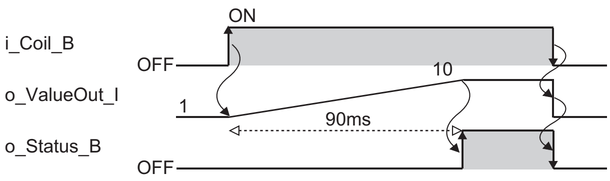
[Label definitions]

Label name	Data type	Class	Description
TIMER_10_FB_M_1	TIMER_10_FB_M	VAR	An instance of standard FB
i_Coil_B	Bit	VAR	Executing condition (TRUE: Execution, FALSE: Stop)
o_ValueOut_I	Word (signed)	VAR	Timer current value
o_Status_B	Bit	VAR	Output

[Program]

```
TIMER_10_FB_M_1(
Coil := i_Coil_B ,
Preset := 10 ,
ValueIn := 1 ,
ValueOut => o_ValueOut_I ,
Status => o_Status_B );
```

[Timing chart]



■TIMER_100_FB_M

- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×100 ms. When the value reaches (s2)×100 ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- If the unit of measurement of the low-speed timer (in the timer limit setting) is changed from the default value using the engineering tool, a warning will be issued during conversion of modified or newly added programs or all programs in a project.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

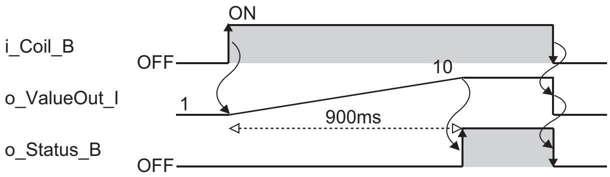
[Label definitions]

Label name	Data type	Class	Description
TIMER_100_FB_M_1	TIMER_100_FB_M	VAR	An instance of standard FB
i_Coil_B	Bit	VAR	Executing condition (TRUE: Execution, FALSE: Stop)
o_ValueOut_I	Word (signed)	VAR	Timer current value
o_Status_B	Bit	VAR	Output

[Program]

```
TIMER_100_FB_M_1(
Coil := i_Coil_B ,
Preset := 10 ,
ValueIn := 1 ,
ValueOut => o_ValueOut_I ,
Status => o_Status_B );
```

[Timing chart]



■TIMER_HIGH_FB_M

- This is a high-speed timer whose unit of measurement is 0.1 to 100 ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×0.1 to 100 ms (variable; set in parameter). When the value reaches (s2)×0.1 to 100 ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- The unit of measurement of the high-speed timer is 10 ms by default. The unit can be changed in the range from 0.01 to 100 ms.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

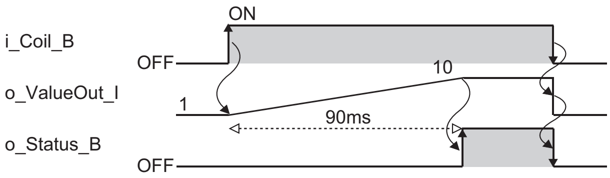
[Label definitions]

Label name	Data type	Class	Description
TIMER_HIGH_FB_M_1	TIMER_HIGH_FB_M	VAR	An instance of standard FB
i_Coil_B	Bit	VAR	Executing condition (TRUE: Execution, FALSE: Stop)
o_ValueOut_I	Word (signed)	VAR	Timer current value
o_Status_B	Bit	VAR	Output

[Program]

```
TIMER_HIGH_FB_M_1(
Coil := i_Coil_B ,
Preset := 10 ,
ValueIn := 1 ,
ValueOut => o_ValueOut_I ,
Status => o_Status_B );
```

[Timing chart]



■TIMER_LOW_FB_M

- This is a low-speed timer whose unit of measurement is 1 to 1000 ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×1 to 1000 ms (variable; set in parameter). When the value reaches (s2)×1 to 1000 ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- The unit of measurement of the low-speed timer is 100 ms by default. The unit can be changed in the range from 1 to 1000 ms (in increments of 1 ms).
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

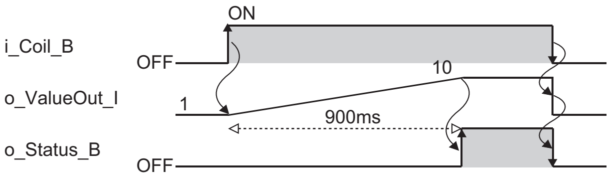
[Label definitions]

Label name	Data type	Class	Description
TIMER_LOW_FB_M_1	TIMER_LOW_FB_M	VAR	An instance of standard FB
i_Coil_B	Bit	VAR	Executing condition (TRUE: Execution, FALSE: Stop)
o_ValueOut_I	Word (signed)	VAR	Timer current value
o_Status_B	Bit	VAR	Output

[Program]

```
TIMER_LOW_FB_M_1(
Coil := i_Coil_B ,
Preset := 10 ,
ValueIn := 1 ,
ValueOut => o_ValueOut_I ,
Status => o_Status_B );
```

[Timing chart]



■TIMER_CONT_FB_M/TIMER_CONTHFB_M

- This is a retentive timer that measures the on time of a variable. When (s1) turns on, measurement of the current value starts. There are two retentive timers: low-speed (TIMER_CONT_FB_M) and high-speed (TIMER_CONTHFB_M) retentive timers.
- The measurement starts from (s3)×1 to 1000 ms (0.1 to 100 ms for the high-speed retentive timer) (variable; set in parameter). When the value reaches (s2)×1 to 1000 ms (0.1 to 100 ms for the high-speed retentive timer), (d2) turns on. The measured current value is output to (d1).
- Even when (s1) is off, the on/off states of (d1) and (d2) are held. When (s1) turns on again, the measurement resumes with the measured value that has been held.
- The unit of measurement (time limit) for the retentive timers is common to both the low-speed timer (TIMER_LOW_FB_M) and high-speed timer (TIMER_HIGH_FB_M).
  - Low-speed retentive timer: Low-speed timer
  - High-speed retentive timer: High-speed timer
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.
- To reset (d1) of a retentive timer, reset (s1) of FB directly.

Ex.

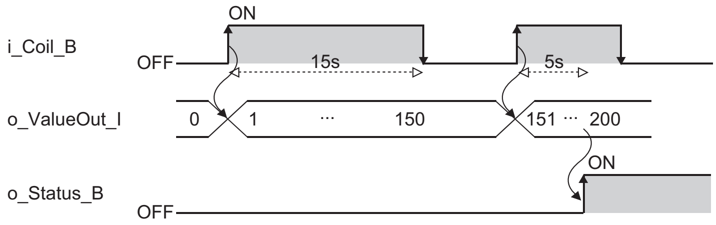
[Label definitions]

Label name	Data type	Class	Description
TIMER_CONT_FB_M_1	TIMER_CONT_FB_M	VAR	An instance of standard FB
i_Coil_B	Bit	VAR	Executing condition (TRUE: Execution, FALSE: Stop)
o_ValueOut_I	Word (signed)	VAR	Timer current value
o_Status_B	Bit	VAR	Output

[Program]

```
TIMER_CONT_FB_M_1(
Coil := i_Coil_B ,
Preset := 200 ,
ValueIn := 0 ,
ValueOut => o_ValueOut_I ,
Status => o_Status_B );
```

[Timing chart]



Operation error

There is no operation error.

MEMO

PART 8 MOTION DEDICATED INSTRUCTIONS

This part consists of the following chapter.

19 MOTION DEDICATED INSTRUCTIONS

19.1 Overview

The module dedicated instruction enables access and execution of instructions to the labels defined in the motion from control CPUs such as a programmable controller CPU.

The dedicated instruction which can be executed from the programmable controller CPU to the Motion module is shown below.

Instruction	Execution condition
G(P).CEXECUTE	Instructs the execution of processing in the Motion module.

The module dedicated instruction executes the processing with the dedicated instruction execution task in the motion module. The priority of the task is lower than the fixed cycle task (which executes the motion operation, etc.) and higher than the normal task. Therefore, it does not effect to the operation cycle by the instruction execution. However, it may cause the processing time of the normal task gets longer.

Operation of this function for each system status

○: Possible, ×: Not possible

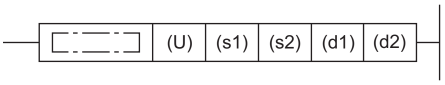
System status	Operation availability
STOP	○
RUN	○
Moderate error	○
Major error	×

19.2 User Function Execution Instruction

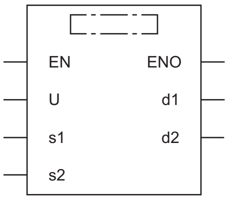
G(P).CEXECUTE

These instructions instruct the execution of processing in the Motion module.

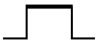
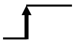
Ladder	ST

	ENO:=G_CEXECUTE(EN,U,s1,s2,d1,d2);
	ENO:=GP_CEXECUTE(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.CEXECUTE	
GP.CEXECUTE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (Label)
(U)	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Start device where control data is stored	Page 204 Control data	Device name	ANY16[*2]
(s2)	Start device where request data is stored	—[*1]	Device name	ANY16[*2]
(d1)	Start device for storing response data	—[*1]	Device name	ANY16[*2]
(d2)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d2)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] The maximum size of response data and request data will be 8K words.

[*2] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices/labels

Operand	Bit		Word			Double Word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	S	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○*1	—	—	—	—	○	—	—	—	—
(s2)	—	—	○*1	—	—	—	—	○	—	—	—	—
(d1)	—	—	○*1	—	—	—	—	○	—	—	—	—
(d2)	○*2	—	○*3	—	—	—	—	—	—	—	—	—

*1 FD cannot be used.

*2 FX and FY cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Allowable number of response data	Sets the allowable number of words of response data that can be stored in (d1).	1 to 8192	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System

■Request data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the request data length. (Number of words)	1 to 8192	User
+1 to +□	Request data	Specify the request data.	—	User

■Response data

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The response data length is stored. (Number of words)	0 to 8192	System
+1 to +□	Response data	The response data is stored.	—	System

Processing details

- The request data stored in the device specified by (s2) and later is handed over to the Motion module specified by (U), and the response data is stored in the device specified by (d1) and later. However, if the received response data is larger than the allowable number of response data specified in (s1), only the allowable number of response data will be stored and the remaining will be discarded. (The dedicated instruction will be completed successfully.) In this case, the response data length (d1) will be the number of data actually stored.
- Set (s2) according to the process which will be executed. The description of (d1) changes depending on the process to be executed. The following shows details.

■Label read request

Reads labels from the Motion module.

The execution of this request requires the SignalIO add-on.

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the request data length. (Number of words from +1 to +□)	3 to 258	User
+1	Request ID	Specify "1".	1	User
+2 to +□	Label name	Specify the label name to be read with character string [Unicode] (including the last NULL character). • Only primitive data type can be read. Specify the end element for structures. • Specify [start...end] when reading out the multiple array elements in a batch. This notation can be used at only one location in the label. Ex. If the array is Label[10...20], the elements [10] to [20] will be read. • Local labels cannot be specified.	Maximum of 257 characters	User

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The response data length is stored. (Number of words) • The response data length depends on the label type to be read. • If an array is specified and the response data length is greater than 8192 points, the data will be stored within the range of not exceeding 8192 points and not separating the data and complete normally. Ex. The WSTRING character string of 100 length is stored up to 81 strings (8100 points), the 82nd string will not be stored.	4 to 8192	System
+1 to +3	Reserved	"0" is stored.	0	System
+4 to +□	Read data	The read data is stored. • Stores in bit 0 for a BOOL type label. When reading out the multiple data with specified array, store in order without space such as bit 0, bit 1, ... • When reading out the array of STRING/WSTRING type array, output the character string in the NULL tab-delimited format type. In the case of STRING type, adjust the number of NULL to set each head of the string at word border. (Insert NULL x 1 when the length is odd number without NULL and insert NULL x 2 in the case of even number.)	—	System

■Label write request

Writes labels to the Motion module.

The execution of this request requires the SignalIO add-on.

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the request data length. (Number of words from +1 to +□)	4 to 8192	User
+1	Request ID	Specify "2".	2	User
+2 to +M	Label name	Specify the label name to be written with character string [Unicode] (including the last NULL character). • Only primitive data type can be written. Specify the end element for structures. • Specify [start...end] when writing out the multiple array elements in a batch. This notation can be used at only one location in the label. Ex. If the array is Label[10...20], the elements [10] to [20] will be written. • Local labels cannot be specified.	Maximum of 256 characters	User
+(M+1) to +□	Write data	Specify the data to be written. • If an array is specified and the request data length exceeds the request data length specified in (s2+0), the data will be written within the range of not exceeding the request data length and not seperated, and complete normally. • Stores in bit 0 for a BOOL type label. When reading out the multiple data with specified array, store in order without space such as bit 0, bit 1, …. • When writing in the array of STRING/WSTRING type label, set the character string in the NULL tab-delimited format type. In the case of STRING type, adjust the number of NULL to set each head of the string at word border.	—	User

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The response data length is stored. (Number of words)	1	System
+1	Number of write points	The number of points (number of words) written is stored. • The bool type label is 16 pieces/1 point.	—	System

■Motion instruction execution

Executes the program instructions of the Motion module. For the processing details of each instruction, refer to the following. The execution of this request requires Addon_Program_ST.

☞ Page 95 Program Control Instructions

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the request data length. (Number of words from +1 to +□)	3 to 8192	User
+1	Request ID	Specify "3".	3	User
+2 to +M	Instruction name	Specify the instruction to be executed with ASCII character string (including the last NULL character).	Maximum of 256 characters	User
+(M+1) to +□	Argument	Specify the argument to be passed to the instruction.	—	User

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The response data length is stored. (Number of words)	1 to 8192	System
+1 to +□	Response data	The response data is stored.	—	System

Instruction names which can be specified, the arguments, and the response data are shown below.

Instruction name	Argument	Response data
PSCAN[*1]	+0 to: program name (character string [Unicode], including the last NULL character)	+0: bit 0 ...Execution result (ENO) is stored.[*2] bit 1-F ...0 is stored.
PSTOP[*1]	+0 to: program name (character string [Unicode], including the last NULL character)	+0: bit 0 ... Execution result (ENO) is stored.[*2] bit 1-F ...0 is stored.

[*1] Return the response data by executing the acceptance of instruction with dedicated instruction task. After that, execute the change processing with the normal task cycle.

[*2] It is always 1 when this instruction is executed. The judgement of the success and failure by execution, check the PROGRAM_INFO.Status.

Point

The instruction can be executed only while the Motion module is set to RUN. If the instruction is executed while the Motion module is set to STOP, an error will occur.

Precautions

- The G(P).CEXECUTE instruction cannot be executed additionally while another G(P).CEXECUTE instruction is being executed. If two G(P).CEXECUTE instructions are executed at the same time, their operation will not be guaranteed. Implement measures such as executing the next G(P).CEXECUTE instruction after the completion device (d2) of the first instruction turns ON to prevent two instructions from being executed at the same time.
- The operand must be specified even when request data and response data are not required.
- Do not change each data (control data and request data, etc.) specified in the dedicated instruction until the dedicated instruction process is completed.
- The character string type or the structure including the character string type of the Motion module is not made public. To read/write from the PLC program, use the G(P).CEXECUTE instruction.
- The operation when reading or writing public labels using the G(P).CEXECUTE instruction is as follows. Depending on the timing of the operation, the consistency of data may not be maintained. It is recommended to use the instruction depending on the purpose, such as using public labels for things that need to be updated in fixed cycles, and accessing all other things with the G(P).CEXECUTE instruction.

G(P).CEXECUTE	**Label read**	**Label write**
Public label Motion control type		
Programmable controller CPU ⇨ Motion module	The read value and the module label value may not match depending on the operation timing.[1]	Will be overwritten with the public label.
Motion module ⇨ Programmable controller CPU	The read value and the module label value may not match depending on the operation timing.[1]	The write value and the module label value may not match depending on the operation timing.[1]

[1] The priority of the dedicated instruction execution task (☞ Page 202 Overview) is lower than buffer memory refresh processing task, and will not synchronize with public label refresh.

Operation error

Error code ((s1)+1)	**Description**
1800H	The out of the range value is specified for the request ID.
1801H	The G(P).CEXECUTE instruction is executed in the dedicated instruction disabled status.
1802H	Multiple instructions are executed.
1803H	• The system memory capacity for the PlcInstruction add-on is insufficient. • The buffer memory capacity (For Motion Control FB area) is insufficient.
1804H	• The specified request data is error. • An error is detected in Motion Control FB related add-on.[2]
1805H	The specified value set to the request data length is out of range.
1806H	The specified value set to the allowable amount of response data is out of range.
1807H	Required add-ons for the instruction execution have not been loaded.
180FH	Dedicated instruction execution errors.

[2] An system error occurs at the same time. Check the details of the error of each function in the event history.

- For above error codes, take corrective actions according to the generated error.

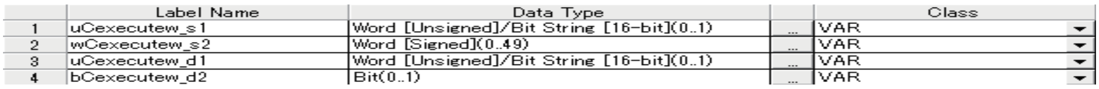
Upon completion with an error, the completion status indication device (d2)+1 is turned TRUE and an error code is stored in the completion status (s1)+1. Response data (d1) will not store the data.

The PlcInstruction add-on does not output the self-diagnostic error. Check the instruction completion status for the instruction execution error.

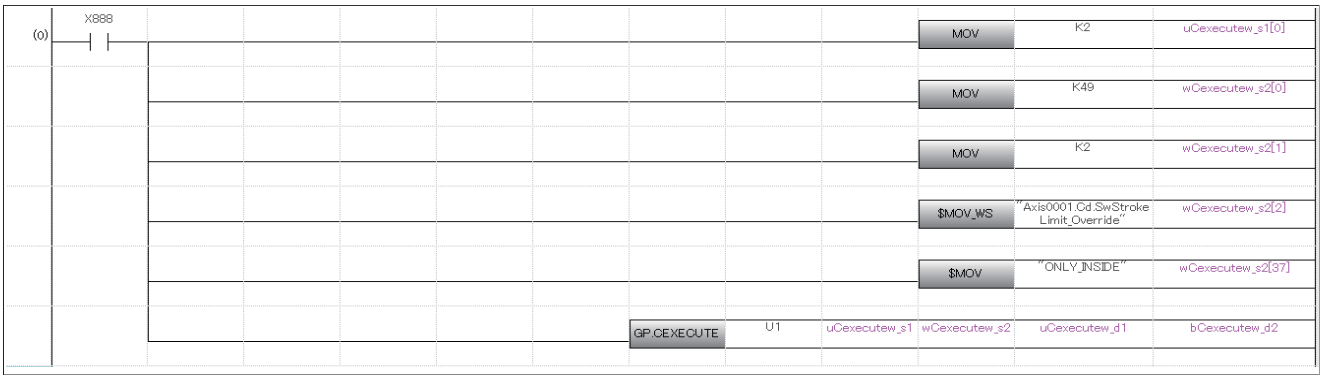
Program example

[Label write request]

A program example is shown below when X888 is turned ON the character string "ONLY_INSIDE" will be written to Software stroke limit override of the axis 1 by the G(P).CEXECUTE instruction.

Classification	Label Name	Description
Module label	Axis0001.Cd.SwStrokeLimit_Override	Software stroke limit override of the axis 1
Local label	Define the local label as follows. The settings of Assign (Device/Label) are not required for the label that the assignment device is not set because the unused internal relay and data device are automatically assigned. 	

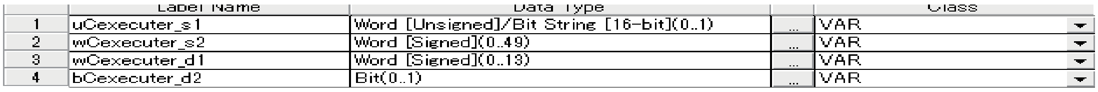
	Label Name	Data Type		Class
1	uCexecutew_s1	Word [Unsigned]/Bit String [16-bit](0..1)	...	VAR
2	wCexecutew_s2	Word [Signed](0..49)	...	VAR
3	uCexecutew_d1	Word [Unsigned]/Bit String [16-bit](0..1)	...	VAR
4	bCexecutew_d2	Bit(0..1)	...	VAR



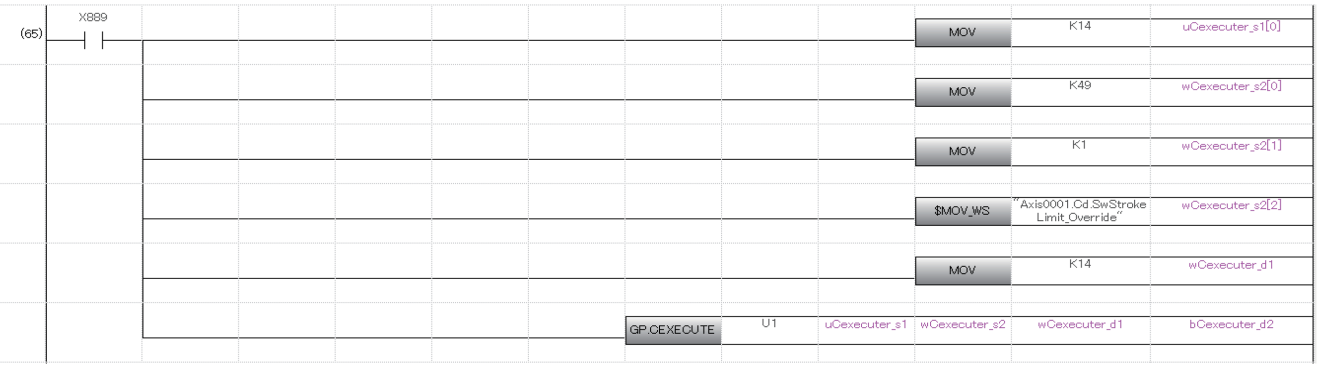
```
(0)  X888
     -| |-+------------------------------------------[MOV      K2                          uCexecutew_s1[0]  ]
          +------------------------------------------[MOV      K49                         wCexecutew_s2[0]  ]
          +------------------------------------------[MOV      K2                          wCexecutew_s2[1]  ]
          +------------------------------------------[$MOV_WS  "Axis0001.Cd.SwStroke
                                                                Limit_Override"            wCexecutew_s2[2]  ]
          +------------------------------------------[$MOV     "ONLY_INSIDE"               wCexecutew_s2[37] ]
          +------------------[GP.CEXECUTE  U1  uCexecutew_s1  wCexecutew_s2  uCexecutew_d1  bCexecutew_d2    ]
```

[Label read request]

A program example is shown below when X889 is turned ON the character string set to Software stroke limit override will be read of the axis 1 by the G(P).CEXECUTE instruction.

Classification	Label Name	Description
Module label	Axis0001.Cd.SwStrokeLimit_Override	Software stroke limit override of the axis 1
Local label	Define the local label as follows. The settings of Assign (Device/Label) are not required for the label that the assignment device is not set because the unused internal relay and data device are automatically assigned. 	

	Label Name	Data Type		Class
1	uCexecuter_s1	Word [Unsigned]/Bit String [16-bit](0..1)	...	VAR
2	wCexecuter_s2	Word [Signed](0..49)	...	VAR
3	wCexecuter_d1	Word [Signed](0..13)	...	VAR
4	bCexecuter_d2	Bit(0..1)	...	VAR



```
(65) X889
     -| |-+------------------------------------------[MOV      K14                         uCexecuter_s1[0]  ]
          +------------------------------------------[MOV      K49                         wCexecuter_s2[0]  ]
          +------------------------------------------[MOV      K1                          wCexecuter_s2[1]  ]
          +------------------------------------------[$MOV_WS  "Axis0001.Cd.SwStroke
                                                                Limit_Override"            wCexecuter_s2[2]  ]
          +------------------------------------------[MOV      K14                         wCexecuter_d1     ]
          +------------------[GP.CEXECUTE  U1  uCexecuter_s1  wCexecuter_s2  wCexecuter_d1  bCexecuter_d2    ]
```

MEMO

INDEX

INSTRUCTION INDEX

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
July 2019	IB(NA)-0300431ENG-A	First edition
January 2020	IB(NA)-0300431ENG-B	■Added models RD78GHV, RD78GHW ■Added or modified parts TERMS, GENERIC TERMS AND ABBREVIATIONS, Section 1.2, 11.1, 11.2, 19.4, WARRANTY, TRADEMARKS
August 2020	IB(NA)-0300431ENG-C	■Added or modified parts RELEVANT MANUAL, MANUAL PAGE ORGANIZATION, Part 2, 6, 7, 8, Chapter 3, 4, 11, 12, 13, 14, 15, 16, 17 ,18, 19, Section 1.1, 1.4, 3.2, 5.1, 6.1, 7.1, 7.2, 9.1, 10.1, 11.26, 11.30, 11.31, 11.32, 11.33, 11.34, 11.35, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, 12.9, 12.10, 12.11, 13.1, 13.2, 13.3, 13.4, 13.5, 13.6, 15.1, 16.2, 18.1, 18.2, 18.3, 18.4, 19.1, 19.2, INSTRUCTION INDEX
August 2021	IB(NA)-0300431ENG-D	■Added or modified parts INTRODUCTION, RELEVANT MANUALS, TERMS, GENERIC TERMS AND ABBREIVIATIONS, Section 1.2, 6.2, 6.3, 7.2, 8.1, 18.2, 19.2, WARRANTY
January 2022	IB(NA)-0300431ENG-E	■Added or modified parts RELEVANT MANUALS, TERMS, FUTURE SUPPORT PLANNED, Section 3.2, 5.1
August 2022	IB(NA)-0300431ENG-F	Partially modified
November 2023	IB(NA)-0300431ENG-G	■Added or modified parts INFORMATION AND SERVICES, TRADEMARKS

Japanese manual number: IB-0300430-H

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

WARRANTY

Warranty

1. Warranty period and coverage

We will repair any failure or defect hereinafter referred to as "failure" in our FA equipment hereinafter referred to as the "Product" arisen during warranty period at no charge due to causes for which we are responsible through the distributor from which you purchased the Product or our service provider. However, we will charge the actual cost of dispatching our engineer for an on-site repair work on request by customer in Japan or overseas countries. We are not responsible for any on-site readjustment and/or trial run that may be required after a defective unit is repaired or replaced.

[Term]

For terms of warranty, please contact your original place of purchase.

[Limitations]

(1) You are requested to conduct an initial failure diagnosis by yourself, as a general rule.
It can also be carried out by us or our service company upon your request and the actual cost will be charged. However, it will not be charged if we are responsible for the cause of the failure.

(2) This limited warranty applies only when the condition, method, environment, etc. of use are in compliance with the terms and conditions and instructions that are set forth in the instruction manual and user manual for the Product and the caution label affixed to the Product.

(3) Even during the term of warranty, the repair cost will be charged on you in the following cases;

1. a failure caused by your improper storing or handling, carelessness or negligence, etc., and a failure caused by your hardware or software problem
2. a failure caused by any alteration, etc. to the Product made on your side without our approval
3. a failure which may be regarded as avoidable, if your equipment in which the Product is incorporated is equipped with a safety device required by applicable laws and has any function or structure considered to be indispensable according to a common sense in the industry
4. a failure which may be regarded as avoidable if consumable parts designated in the instruction manual, etc. are duly maintained and replaced
5. any replacement of consumable parts (battery, fan, smoothing capacitor, etc.)
6. a failure caused by external factors such as inevitable accidents, including without limitation fire and abnormal fluctuation of voltage, and acts of God, including without limitation earthquake, lightning and natural disasters
7. a failure generated by an unforeseeable cause with a scientific technology that was not available at the time of the shipment of the Product from our company
8. any other failures which we are not responsible for or which you acknowledge we are not responsible for

2. Term of warranty after the stop of production

(1) We may accept the repair at charge for another seven (7) years after the production of the product is discontinued. The announcement of the stop of production for each model can be seen in our Sales and Service, etc.

(2) Please note that the Product (including its spare parts) cannot be ordered after its stop of production.

3. Service in overseas countries

Our regional FA Center in overseas countries will accept the repair work of the Product. However, the terms and conditions of the repair work may differ depending on each FA Center. Please ask your local FA center for details.

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 to:

(1) Damages caused by any cause found not to be the responsibility of Mitsubishi.

(2) Loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products.

(3) Special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products.

(4) Replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Change of Product specifications

Specifications listed in our catalogs, manuals or technical documents may be changed without notice.

6. Application and use of the Product

(1) For the use of our Motion module, its applications should be those that may not result in a serious damage even if any failure or malfunction occurs in the Motion module, and a backup or fail-safe function should operate on an external system to the Motion module when any failure or malfunction occurs.

(2) Our Motion module is designed and manufactured as a general purpose product for use at general industries.
Therefore, applications substantially influential on the public interest for such as atomic power plants and other power plants of electric power companies, and also which require a special quality assurance system, including applications for railway companies and government or public offices are not recommended, and we assume no responsibility for any failure caused by these applications when used
In addition, applications which may be substantially influential to human lives or properties for such as airlines, medical treatments, railway service, incineration and fuel systems, man-operated material handling equipment, entertainment machines, safety machines, etc. are not recommended, and we assume no responsibility for any failure caused by these applications when used.
We will review the acceptability of the abovementioned applications, if you agree not to require a specific quality for a specific application. Please contact us for consultation.

(3) Mitsubishi shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

INFORMATION AND SERVICES

For further information and services, please contact your local Mitsubishi Electric sales office or representative.
Visit our website to find our locations worldwide.

MITSUBISHI ELECTRIC Factory Automation Global Website
Locations Worldwide
www.MitsubishiElectric.com/fa/about-us/overseas/

TRADEMARKS

Microsoft, Microsoft Access, Excel, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Windows Vista, and Windows XP are trademarks of the Microsoft group of companies.

The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.

In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

IB(NA)-0300431ENG-G(2311)MEE
MODEL: RD78-P-MF-E
MODEL CODE: 1XB041

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BLDG., 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 1-14, YADA-MINAMI 5-CHOME, HIGASHI-KU, NAGOYA 461-8670, JAPAN

When exported from Japan, this manual does not require application to the Ministry of Economy, Trade and Industry for service transaction permission.

Specifications subject to change without notice.