MITSUBISHI
ELECTRIC

Programmable Controller

MELSEC iQ-R
series

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

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.

CONDITIONS OF USE FOR THE PRODUCT

(1) Mitsubishi programmable controller ("the PRODUCT") shall be used in conditions;

i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and

ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.

MITSUBISHI SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI'S USER, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT.

("Prohibited Application")

Prohibited Applications include, but not limited to, the use of the PRODUCT in;

- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above restrictions, Mitsubishi may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi representative in your region.

- When the Safety CPU is used

(1) Although MELCO has obtained the certification for Product's compliance to the international safety standards IEC61508, EN954-1/ISO13849-1 from TUV Rheinland, this fact does not guarantee that Product will be free from any malfunction or failure. The user of this Product shall comply with any and all applicable safety standard, regulation or law and take appropriate safety measures for the system in which the Product is installed or used and shall take the second or third safety measures other than the Product. MELCO is not liable for damages that could have been prevented by compliance with any applicable safety standard, regulation or law.

(2) MELCO prohibits the use of Products with or in any application involving, and MELCO shall not be liable for a default, a liability for defect warranty, a quality assurance, negligence or other tort and a product liability in these applications.
  (a) power plants,
  (b) trains, railway systems, airplanes, airline operations, other transportation systems,
  (c) hospitals, medical care, dialysis and life support facilities or equipment,
  (d) amusement equipments,
  (e) incineration and fuel devices,
  (f) handling of nuclear or hazardous materials or chemicals,
  (g) mining and drilling,
  (h) and other applications where the level of risk to human life, health or property are elevated.

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.

CONTENTS

PART 1 OVERVIEW

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

PART 3 CPU MODULE INSTRUCTIONS

CHAPTER 9 PID OPERATION INSTRUCTION 1134

CHAPTER 10 PID CONTROL INSTRUCTIONS 1149

CHAPTER 11 PROCESS CONTROL INSTRUCTIONS 1181

CHAPTER 13 SFC PROGRAM INSTRUCTIONS 1411

CHAPTER 14 REDUNDANT SYSTEM INSTRUCTIONS 1440

PART 4 MODULE DEDICATED INSTRUCTIONS

CHAPTER 15 NETWORK COMMON INSTRUCTIONS 1448

CHAPTER 16 ETHERNET INSTRUCTIONS 1535

PART 6 STANDARD FUNCTION BLOCKS

RELEVANT MANUALS

Manual name [manual number]	Description	Available form
MELSEC iQ-R Programming Manual (Instructions, Standard Functions/ Function Blocks) [SH-081266ENG] (this manual)	Instructions for the CPU module, dedicated instructions for the intelligent function modules, and standard functions/function blocks	e-Manual PDF
MELSEC iQ-R Programming Manual (Program Design) [SH-081265ENG]	Program specifications (ladder, ST, FBD/LD, and SFC programs) and labels	e-Manual PDF
MELSEC iQ-R Programming Manual (Process Control Function Blocks) [SH-081749ENG]	Standard process function blocks, tag access function blocks, and tag function blocks designed for the process control	e-Manual PDF
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3	e-Manual PDF

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.

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
A/D converter module	A generic term for MELSEC iQ-R series analog-digital converter module, channel isolated analog-digital converter module, and high speed analog-digital converter module
Analog module	A generic term for the A/D converter module, D/A converter module, and temperature input module
Backup mode	A mode for normal operation in a redundant system
Buffer memory	Memory in an intelligent function module for storing data such as setting values and monitored values. Buffer memory in a CPU module stores setting values and monitored values of the Ethernet function and data used for data communications among the CPU modules in a multiple CPU system.
CC-Link IE Controller Network module	The abbreviation for the MELSEC iQ-R series CC-Link IE Controller Network-equipped module
CC-Link IE Controller Network-equipped module	A generic term for the following modules when the CC-Link IE Controller Network function is used: • RJ71GP21-SX • RJ71EN71 • RnENCPU
CC-Link IE Field Network master/local module	The abbreviation for the CC-Link IE Field Network-equipped master/local module
CC-Link IE Field Network-equipped master/local module	A generic term for the following modules when the CC-Link IE Field Network function is used: • RJ71GF11-T2 • RJ71EN71 • RnENCPU
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.
Control system	A system that takes control and performs network communications in a redundant system
CPU module	The abbreviation for the MELSEC iQ-R series CPU module
D/A converter module	A generic term for the MELSEC iQ-R series digital-analog converter module, channel isolated digital-analog converter module, and high speed digital-analog converter module
Engineering tool	The product name of the software package for the MELSEC programmable controllers
Ethernet interface module with built-in CC-Link IE	A generic term for the following modules when the Ethernet function is used: • RJ71EN71 • CPU module
Ethernet module	The abbreviation for the MELSEC iQ-R series CC-Link IE built-in Ethernet interface module
External device	A generic term for the personal computer and other Ethernet-equipped modules connected over Ethernet for data communications
External device	A generic term for devices that send SLMP request messages to a SLMP-compatible device (personal computers, HMI (Human Machine Interface) and others)
FBD/LD	The abbreviation for the function block diagram/ladder diagram
I/O module	A generic term for the input module, output module, I/O combined module, and interrupt module
Intelligent function module	A module that has functions other than input and output, such as an analog module
Label	A label that represents a device in a given character string
Local station	A station that performs cyclic transmission and transient transmission with the master station and other local stations on CC-Link IE Field Network.
Master station	A station that controls the entire network on CC-Link IE Field Network. This station can perform cyclic transmission and transient transmission with all stations. Only one master station can be used in a network.
MELSECNET/10	The abbreviation for the MELSECNET/10 network system
MELSECNET/H	The abbreviation for the MELSECNET/H network system
Network module	A generic term for the following modules: • Ethernet interface module • CC-Link IE Controller Network module • CC-Link IE Field Network module • MELSECNET/H module • MELSECNET/10 module • RnENCPU (network part)
Operand	A generic term for the devices, such as source data (s), destination data (d), number of devices (n), and others, used as parts to configure instructions and functions
Positioning module	The abbreviation for the MELSEC iQ-R series positioning module

Term	Description
Predefined protocol support function	A function of GX Works3. This function sets protocols appropriate to each external device and reads/writes protocol setting data.
Process CPU	A generic term for the R08PCPU, R16PCPU, R32PCPU, and R120PCPU
Programmable controller CPU	A generic term for the R04CPU, R04ENCPU, R08CPU, R08ENCPU, R16CPU, R16ENCPU, R32CPU, R32ENCPU, R120CPU, and R120ENCPU
Redundant system	A system consisting of two systems that have same configuration (CPU module, power supply module, network module, and other modules). Even after an error occurs in one of the two system, the other system takes over the control of the entire system.
Remote head module	The abbreviation for the RJ72GF15-T2 CC-Link IE Field Network remote head module
Request message	A processing request message sent from external devices to SLMP-compatible devices
Response message	A processing result message sent from SLMP-compatible devices in response to the request message
RnCPU	A generic term for the R04CPU, R08CPU, R16CPU, R32CPU, and R120CPU
RnENCPU	A generic term for the R04ENCPU, R08ENCPU, R16ENCPU, R32ENCPU, and R120ENCPU
RnENCPU (network part)	The right side (network part) of the RnENCPU (📖 MELSEC iQ-R Ethernet/CC-Link IE User's Manual (Startup))
RnPCPU	A generic term for the R08PCPU, R16PCPU, R32PCPU, and R120PCPU
Safety CPU	A generic term for the R08SFCPU, R16SFCPU, R32SFCPU, and R120SFCPU. This module is used with a safety function module as a pair and performs both standard and safety control.
Safety function module	Another term for the R6SFM. This module is used with the Safety CPU as a pair and performs safety control. The module can only be paired with the Safety CPU.
Separate mode	A mode for system maintenance in a redundant system. This mode can maintain a redundant system without stopping control while the system is running.
SFC	The abbreviation for the sequential function chart
SLMP	The abbreviation for Seamless Message Protocol. This protocol is used to access an SLMP-compatible device or a programmable controller connected to an SLMP-compatible device from an external device.
SLMP-compatible device	A generic term for the devices of the Mitsubishi Electric product that can transfer SLMP messages (Ethernet adapter module and Ethernet-equipped module)
ST language	The abbreviation for the structured text language
Standby system	A backup system in a redundant system
System A	A system that is set as system A to distinguish two systems, which are connected with two tracking cables. When the two systems start up at the same time, this system will be a control system. System switching does not affect the system A/B setting.
System B	A system that is set as system B to distinguish two systems, which are connected with two tracking cables. When the two systems start up at the same time, this system will be a standby system. System switching does not affect the system A/B setting.
Temperature input module	A generic term for the MELSEC iQ-R series channel isolated thermocouple input module and channel isolated RTD input module

The following terms are used to explain a safety programmable controller system using the Safety CPU.

Term	Description
Safety cycle processing	Processing of safety input/output and safety program
Safety control	Machine control by safety programs and safety data communications. When an error occurs, the machine in operation is securely stopped.
Safety communications	Communication service that performs send/receive processing in the safety layer of the safety communication protocol
Safety device	A device that can be used in safety programs (📖 MELSEC iQ-R CPU Module User's Manual (Application))
Safety program	A program for performing safety control
Safety label	A generic term for the safety global label, safety local label, and standard/safety shared label (📖 MELSEC iQ-R Programming Manual (Program Design))
Standard CPU	A generic term for MELSEC iQ-R series CPU modules (other than Safety CPU) that perform standard control (This term is used to distinguish from the Safety CPU.)
Standard control	Machine control by standard programs and standard data communications. Programmable controllers other than the safety programmable controller perform only standard control. (This term is used to distinguish from safety control.)
Standard communications	Communications other than safety communications, such as cyclic transmission and transient transmission of CC-Link IE Field Network
Standard device	A device (X, Y, M, D, or others) in a CPU module. (Safety devices are excluded.) This device can be used only in standard programs. (This term is used to distinguish from a safety device.)
Standard program	A program that performs sequence control. (Safety programs are excluded.) (This term is used to distinguish from a safety program.)

Instruction symbols

Unless otherwise specified, this manual uses the following generic symbols for some instructions.

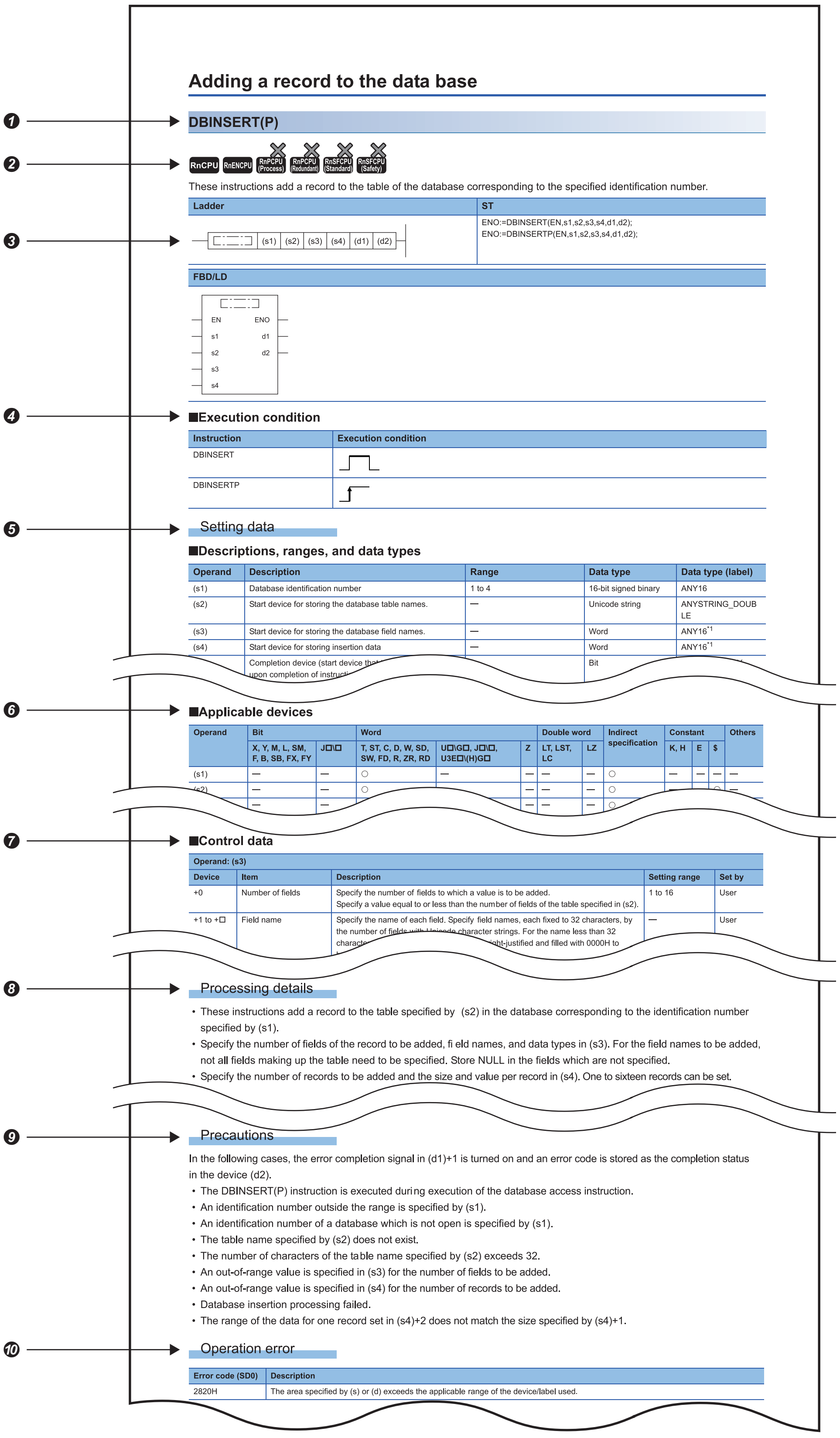
Classification	Instruction symbol	Generic symbol
PID control instruction	S(P).PIDINIT, PIDINIT(P)	PIDINIT
	S(P).PIDCONT, PIDCONT(P)	PIDCONT
	S(P).PIDPRMW, PIDPRMW(P)	PIDPRMW
Multiple CPU dedicated instruction	D(P).DDRD, M(P).DDRD	DDRD
	D(P).DDWR, M(P).DDWR	DDWR
Network common instruction	JP.READ, GP.READ	READ
	JP.SREAD, GP.SREAD	SREAD
	JP.WRITE, GP.WRITE	WRITE
	JP.SWRITE, GP.SWRITE	SWRITE
	JP.SEND, GP.SEND	SEND
	JP.RECV, GP.RECV	RECV
	G.RECVS, Z.RECVS	RECVS
	J(P).REQ, G(P).REQ	REQ
	J(P).RIRD, G(P).RIRD	RIRD
	J(P).RIWT, G(P).RIWT	RIWT
Ethernet instruction	GP.OPEN, ZP.OPEN	OPEN
	GP.CLOSE, ZP.CLOSE	CLOSE
	GP.BUFRCV, ZP.BUFRCV	BUFRCV
	G.BUFRCVS, Z.BUFRCVS	BUFRCVS
	GP.BUFSND, ZP.BUFSND	BUFSND
	G(P).UINI, Z(P).UINI	UINI
CC-Link IE Controller Network instruction	J(P).RRUN, G(P).RRUN, Z(P).RRUN	RRUN
	J(P).RSTOP, G(P).RSTOP, Z(P).RSTOP	RSTOP
	J(P).RTMRD, G(P).RTMRD, Z(P).RTMRD	RTMRD
	J(P).RTMWR, G(P).RTMWR, Z(P).RTMWR	RTMWR
	G(P).UINI, Z(P).UINI	UINI
CC-Link IE Field Network instruction	JP.REMFR, ZP.REMFR	REMFR
	JP.REMTO, ZP.REMTO	REMTO
	JP.REMFRD	REMFRD
	JP.REMTOD	REMTOD
	G(P).UINI, Z(P).UINI	UINI
Positioning instruction	G.ABRST1, G.ABRST2, G.ABRST3, G.ABRST4, Z.ABRST1, Z.ABRST2, Z.ABRST3, Z.ABRST4	ABRST□
	GP.PSTRT1, GP.PSTRT2, GP.PSTRT3, GP.PSTRT4, ZP.PSTRT1, ZP.PSTRT2, ZP.PSTRT3, ZP.PSTRT4	PSTRT□
	GP.TEACH1, GP.TEACH2, GP.TEACH3, GP.TEACH4, ZP.TEACH1, ZP.TEACH2, ZP.TEACH3, ZP.TEACH4	TEACH□
	GP.PFWRT, ZP.PFWRT	PFWRT
	GP.PINIT, ZP.PINIT	PINIT

MANUAL PAGE ORGANIZATION

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

How to read Part 3 and Part 4

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



Adding a record to the data base

DBINSERT(P)

These instructions add a record to the table of the database corresponding to the specified identification number.

Ladder	ST
(s1) (s2) (s3) (s4) (d1) (d2)	ENO:=DBINSERT(EN,s1,s2,s3,s4,d1,d2); ENO:=DBINSERTP(EN,s1,s2,s3,s4,d1,d2);

FBD/LD

■Execution condition

Instruction	Execution condition
DBINSERT	
DBINSERTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the database table names.	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the database field names.	—	Word	ANY16*1
(s4)	Start device for storing insertion data	—	Word	ANY16*1
	Completion device (start device tha upon completion of instructi		Bit	

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○		—	—	—	○				—
	—	—			—	—	—	○				

■Control data

Operand: (s3)				
Device	Item	Description	Setting range	Set by
+0	Number of fields	Specify the number of fields to which a value is to be added. Specify a value equal to or less than the number of fields of the table specified in (s2).	1 to 16	User
+1 to +□	Field name	Specify the name of each field. Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characte ight-justified and filled with 0000H to	—	User

Processing details

- These instructions add a record to the table specified by (s2) in the database corresponding to the identification number specified by (s1).
- Specify the number of fields of the record to be added, field names, and data types in (s3). For the field names to be added, not all fields making up the table need to be specified. Store NULL in the fields which are not specified.
- Specify the number of records to be added and the size and value per record in (s4). One to sixteen records can be set.

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBINSERT(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the number of fields to be added.
- An out-of-range value is specified in (s4) for the number of records to be added.
- Database insertion processing failed.
- The range of the data for one record set in (s4)+2 does not match the size specified by (s4)+1.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s) or (d) exceeds the applicable range of the device/label used.

❶ Instruction symbol

- An instruction symbol followed by parentheses indicates multiple instructions. For example, "GRY(P)(_U)" indicates four instructions: GRY, GRYP, GRY_U, and GRYP_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.

- An instruction symbol followed by "□" indicates multiple instructions. For example, "LDDT□" indicates six instructions: LDDT=, LDDT<>, LDDT>, LDDT<=, LDDT<, and LDDT>=.

❷ Availability by the CPU module type (The instruction cannot be used by the CPU module marked ×.)

❸ Description formats of ladder diagram, structured text language, and FBD/LD

An instruction symbol should be described in the enclosed area of each ladder or FBD/LD program.

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

❹ Execution condition (☞ Page 53 Execution Condition)

❺ Description of operands, setting ranges, data types, and label data types

- For the data type, refer to the following.

☞ Page 36 Data Specification Method

❻ Devices that can be used as operands

Operand	Bit		Word			Double word		Indirect specification	Constant			Others*5
	X, Y, M, L, SM, F, B, SB, S, FX, FY	J□\□*4	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□*4, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
Applicable device*1	X, Y, M, L, SM, F, B, SB, S, FX*2, FY*2	J□\X J□\Y J□\B J□\SB	T*3, ST*3, C*3, D, W, SD, SW, FD*2, R, ZR, RD	U□\G□ U3E□\G□ U3E□\HG□ J□\W J□\SW	Z	LT*3 LST*3 LC*3	LZ	@□ @□.□	K, H	E	$	P, I, J, U, DX, DY, N, V, BL, BL□\S□

The following table lists safety devices that can be used as operands in safety programs executed by the Safety CPU.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
Applicable device*1	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T*3, SA\ST*3, SA\C*3, SA\D, SA\W, SA\SD	K, H

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

*2 FX and FY can be used for bit data only, and FD for word data only.

*3 When T, ST, C, LT, LST, or LC is used for instructions other than those listed below, it can only be used as word data. It cannot be used as bit data.
[Instructions that can be used as bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, BKRST, MOVB(P), CMLB(P)
When SA\T, SA\ST, or SA\C is used for instructions other than those listed below, it can only be used as word data. It cannot be used as bit data.
[Instructions that can be used as bit data] LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, MOVB(P)

*4 This device can be used with a network module with a network number specified.

*5 In the "Others" column, a device(s) that can be set for each instruction is shown.

❼ 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.

❽ Processing details of the instruction. Unless otherwise specified, the following programs are regarded as interrupt programs.

- Interrupt program using the interrupt pointer (I)
- Fixed scan execution type program
- Event execution type program that is triggered by the interrupt pointer (I)

❾ Precautions

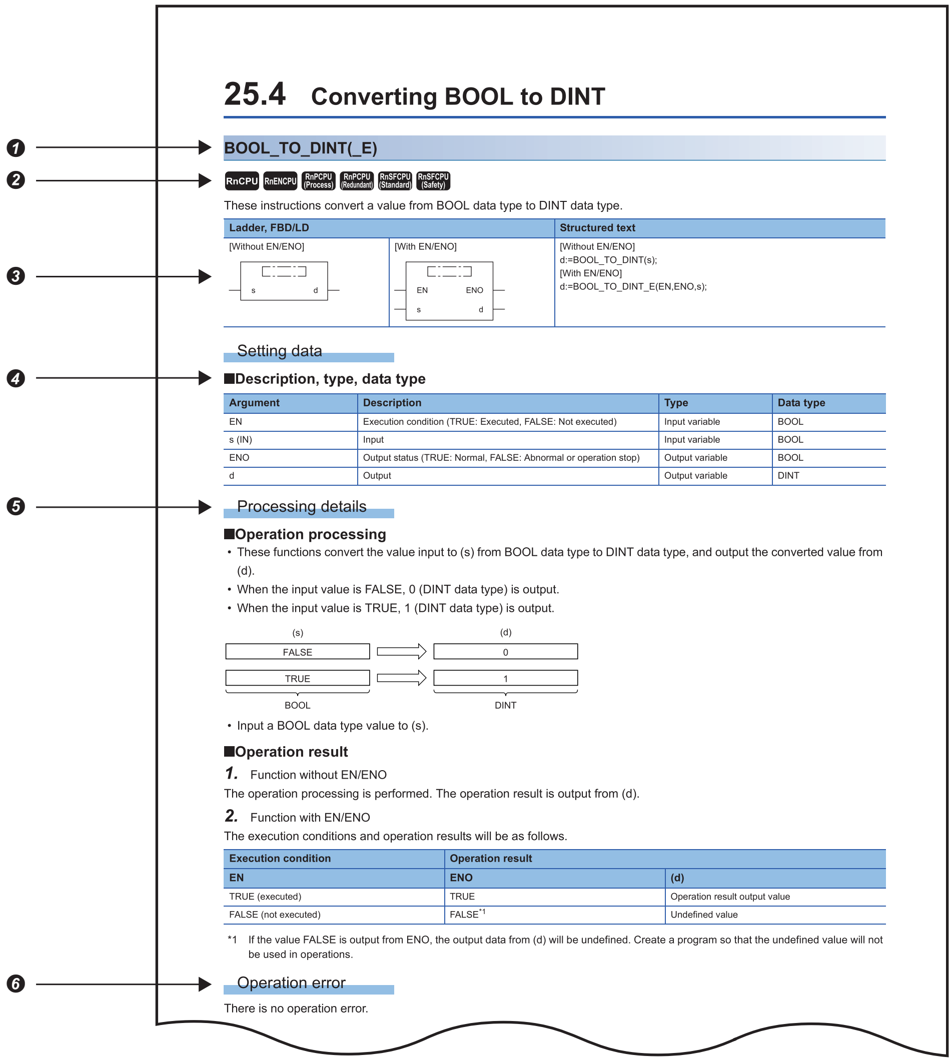
❿ Error code and error details if the instruction has any possible operation error

- A device in which an error code is stored is provided in the error code column. When an error code is stored in SD0, an error flag (SM0) turns on. (The error status can be checked with the module label of the CPU module.)
- For the errors not provided here, refer to the following.

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

How to read Part 5 and Part 6

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



25.4 Converting BOOL to DINT

BOOL_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert a value from BOOL data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_DINT(s);
		[With EN/ENO]
		d:=BOOL_TO_DINT_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

❶ Function symbol

A function symbol followed by parentheses indicates multiple functions or function blocks. For example, "BOOL_TO_DINT(_E)" includes two functions: "BOOL_TO_DINT" and "BOOL_TO_DINT_E".

Function symbol	Meaning
Function symbol followed by "(_E)"	This standard function or standard function block can write program with EN/ENO.

❷ Availability by the CPU module type (The function or function block cannot be used by the CPU module marked ×.)

❸ Description formats of ladder diagram, structured text language, and FBD/LD

In the enclosed area, either of the following symbol should be described.

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

Execution condition is input to EN of each standard function or function block. And, execution result is output from ENO of each standard function or function block.

The return value of functions are not displayed in FBD/LD programs.

For instances, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

❹ Description of operands, types, data types, and label data types

- For the data type, refer to the following.

☞ Page 36 Data Specification Method

❺ Processing details of the standard function or standard function block

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

A device in which an error code is stored is provided in the error code column. When an error code is stored in SD0, an error flag (SM0) turns on. (The error status can be checked with the module label of the CPU module.)

For the errors not provided here, refer to the following.

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

MEMO

PART 1 OVERVIEW

This part consists of the following chapter.

1 OVERVIEW

1.1 Instruction Configuration

Many instructions available for programmable controllers 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. When using constants in variable data, perform index modification.[*1]
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.

[*1] For the index modification, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)

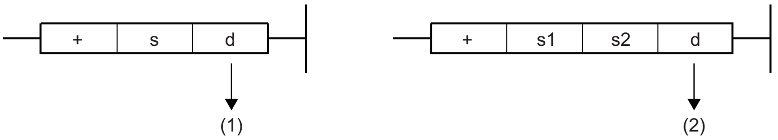
Destination (d)

Data after operation is stored in the destination area.

However, some instructions require the data to be used in the operation to be stored before the operation.

Ex.

Binary 16-bit data addition instruction



(1) The data required for operation is stored before the operation.

(2) Only the operation result is stored.

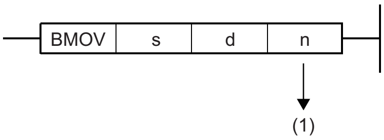
A label or device to store data must be set for the destination.

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.

Ex.

Block transfer instruction



(1) The number of transfers executed by the BMOV instruction is specified.

A numerical value from 0 to 65535 or 0 to 4294967295 can be set for the size such as the number of devices, transfers, or characters.[*1]

Note, however, that when the size specification such as the number of devices, transfers, or characters is 0, the relevant instruction results in non-processing.

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

Point

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

1.2 Data Specification Method

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

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
64-bit data (quad-word data)	64-bit signed binary data
	64-bit unsigned binary data
Real number data (floating-point data)	Single-precision real number data
	Double-precision real number data
BCD data	BCD 4-digit data
	BCD 8-digit data
	BCD 16-digit data
String data	String
	Unicode string

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. ☞ Page 40 Bit data	• Bit device • Bit specification of word device
Word	Word data can be handled. ☞ Page 42 16-bit data (word data)	• 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. ☞ Page 42 16-bit data (word data)	
16-bit unsigned binary		
Double word	Double-word data can be handled. ☞ Page 45 32-bit data (double word data)	• 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. ☞ Page 45 32-bit data (double word data)	
32-bit unsigned binary		
64-bit signed binary	Two consecutive sets of 64-bit data or 32-bit data can be handled. The value range varies depending on whether the value is signed or unsigned.	• Word device • Double-word device • Decimal constant • Hexadecimal constant
64-bit unsigned binary		
BCD 4-digit	BCD 4-digit data can be handled. 16-bit data is divided by 4 digits and each digit is specified in 0 to 9.	• Word device • Digit-specified bit device (K1 to K4)^{*2} • Decimal constant • Hexadecimal constant
BCD 8-digit	BCD 8-digit data can be handled. 32-bit data is divided by 8 digits and each digit is specified in 0 to 9.	• Word device • Double-word device • Digit-specified bit device (K1 to K8)^{*2} • Decimal constant • Hexadecimal constant
BCD 16-digit	BCD 16-digit data can be handled. 64-bit data is divided by 16 digits and each digit is specified in 0 to 9.	• Word device • Double-word device • Decimal constant • Hexadecimal constant
Single-precision real number	Single-precision real number data (single-precision floating-point data) can be handled. ☞ Page 48 Configuration of single-precision real number data	• Word device • Double-word device • Real constant
Double-precision real number	Double-precision real number data (double-precision floating-point data) can be handled. ☞ Page 49 Configuration of double-precision real number data	• Word device • Double-word device • Real constant
Character string	ASCII code and Shift JIS code character string data can be handled. ☞ Page 51 String data	• Word device • Character string constant
Unicode character string	Unicode character string data can be handled. ☞ Page 51 String data	• 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 • Digit-specified bit type label (K1 to K4) • 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 • Digit-specified bit type label (K1 to K8) • Current value of long timer/long retentive timer type label • Current value of long counter type label
Word [signed] (INT)	• Word [signed] type label • Digit-specified bit type label (K1 to K4) • Current value of timer/retentive timer type label • Current value of counter type label
Double word [signed] (DINT)	• Double word [signed] type label • Digit-specified bit type label (K1 to K8) • 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
Pointer (POINTER)	• Pointer type label

Point

For details on individual labels, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

■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 label
ANY[1]	ANY_SIMPLE	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 label
ANYBIT_ARRAY			Bit
ANYWORD_ARRAY	ANY16_ARRAY	ANY16_S_ARRAY	Word [signed]
		ANY16_U_ARRAY	Word [unsigned]/bit string [16 bits]
	ANY32_ARRAY	ANY32_S_ARRAY	Double word [signed], time
		ANY32_U_ARRAY	Double word [unsigned]/bit string [32 bits]
	ANY_REAL_ARRAY	ANY_REAL_32_ARRAY	Single-precision real number
		ANY_REAL_64_ARRAY	Double-precision real number
	ANY_STRING_ARRAY	ANY_STRING_SINGLE_ARRAY	Character string
		ANY_STRING_DOUBLE_ARRAY	Character string [Unicode]
STRUCT_ARRAY			Structures

Bit data

Data size and data range

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

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

Handling bit data with bit devices and labels

Bit data of one point per point can be handled.

Handling bit data with bit word devices

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

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

The following word devices support bit specification.

Item	Device
Word devices which support bit specification	• Data register (D) • Link register (W, J□\W) • Link special register (SW, J□\SW) • Function register (FD) • Special register (SD) • Module access device (U□\G) • CPU buffer memory access device (U3E□\G, U3E□\HG) • File register (R, ZR) • Module refresh register (RD)

A bit number of a safety device used in safety programs executed by the Safety CPU can be specified in hexadecimal within the range from 0 to F.

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

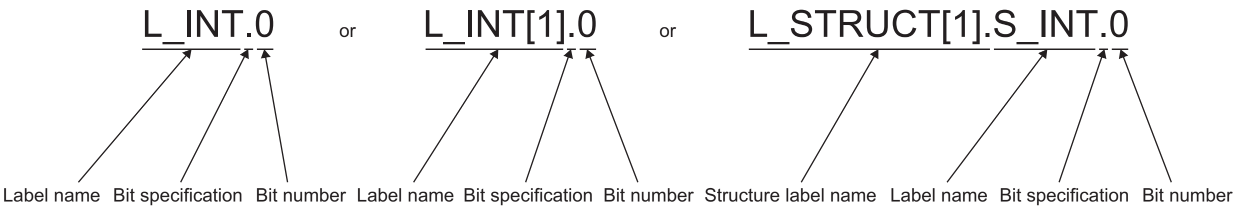
The following word devices support bit specification.

Item	Device
Word devices which support bit specification	• Safety data register (SA\D) • Safety link register (SA\W) • Safety special register (SA\SD)

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)[*1] • Current value (N) of retentive timer (RETENTIVETIMER type)[*1] • Current value (N) of counter (COUNTER type)[*1]

[*1] Cannot be specified in ladder programs.

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	Other than link direct device	K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 4.)	K4X10 K2M113
	Link direct device	J□\K□Bit device start number □ (on the left): Network number □ (on the right): Number of digits (Specify the number within the range of 1 to 4.)	J1\K3B10 J10\K2Y10

A bit device used in safety programs executed by the Safety CPU can be handled as 16-bit data by performing digit specification.

Item	Notation	Example
Bit device	SA\K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 4.)	SA\K4X10 SA\K2M113

Handling 16-bit data with bit type array labels

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

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

Item	Notation	Example
Bit type array label	K□Label name □: Number of digits (Specify the number within the range of 1 to 4.) Specify a label name without an array element.	K1L_BOOL

Digit specification range

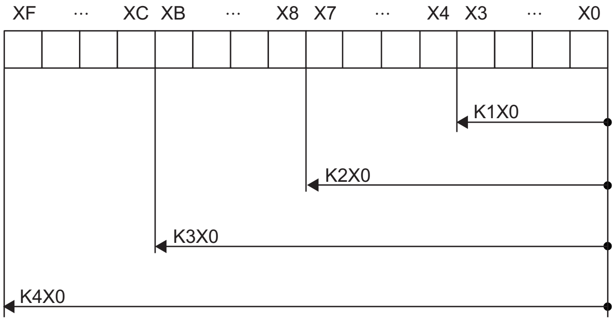
The following table lists the range of 16-bit data for each digit specification.

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.

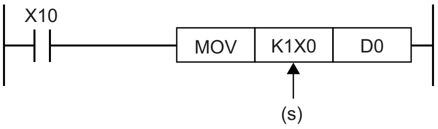
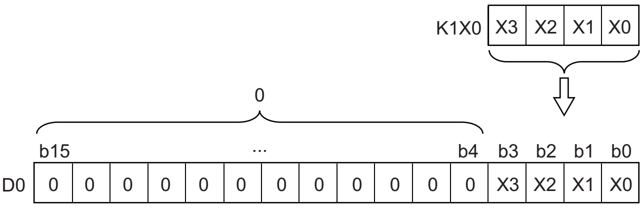
When digit specification is made for X0, the applicable number of points is as follows.

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



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

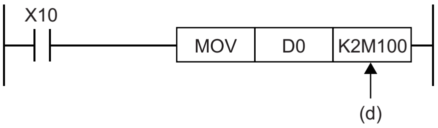
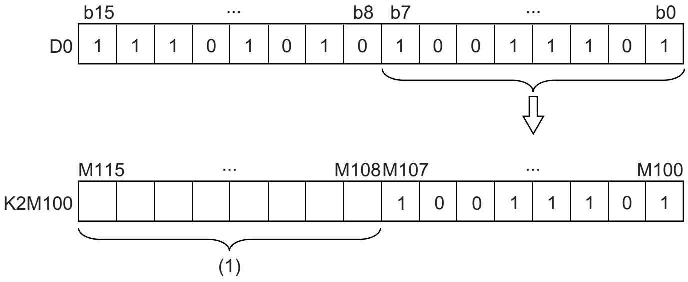
When a bit device with digit specification is specified in the source of an instruction, 0 is stored in the bits, which follow the bit for which digit specification is made in the source, in the word device of the destination.

Ladder example	Processing
• 16-bit data instruction 	

■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 bit devices after the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device 	 (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	Other than link direct device	K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8.	K8X80 K6B018
	Link direct device	J□/K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8. Network number	J1\K7B30 J10\K5X128

A bit device used in safety programs executed by the Safety CPU can be handled as 32-bit data by performing digit specification.

Item	Notation	Example
Bit device	SA\K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8.	SA\K8X80 SA\K6B018

Handling 32-bit data with bit type array labels

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

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

Item	Notation	Example
Bit type array label	K□ Label name Number of digits: Specify the number within the range of 1 to 8. Specify a label name without an array element.	K8L_BOOL

Digit specification range

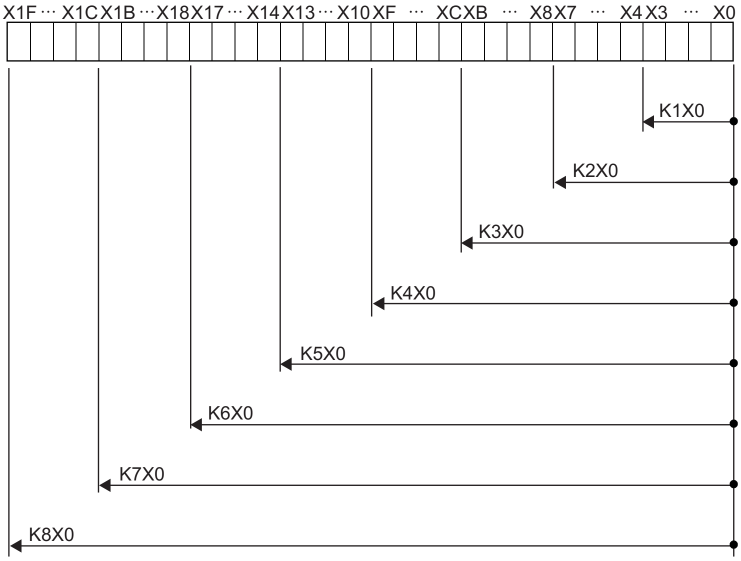
The following table lists the range of 32-bit data for each digit specification.

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.

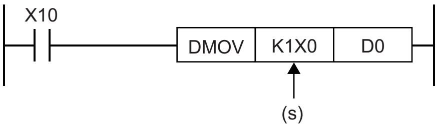
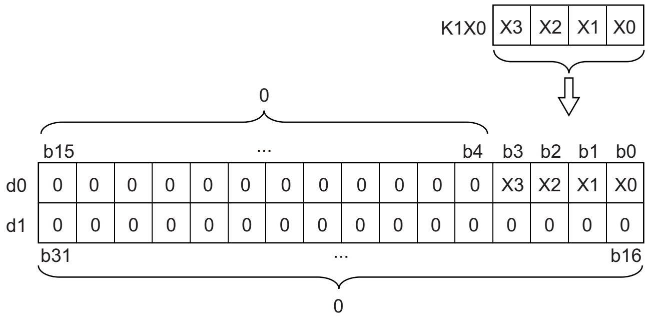
When digit specification is made for X0, the applicable number of points is as follows.

- K1X0→4 points from X0 to X3
- K2X0→8 points from X0 to X7
- K3X0→12 points from X0 to XB
- K4X0→16 points from X0 to XF
- K5X0→20 points from X0 to X13
- K6X0→24 points from X0 to X17
- K7X0→28 points from X0 to X1B
- K8X0→32 points from X0 to X1F



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

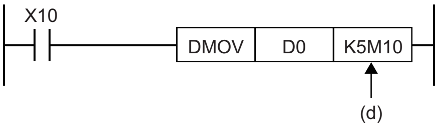
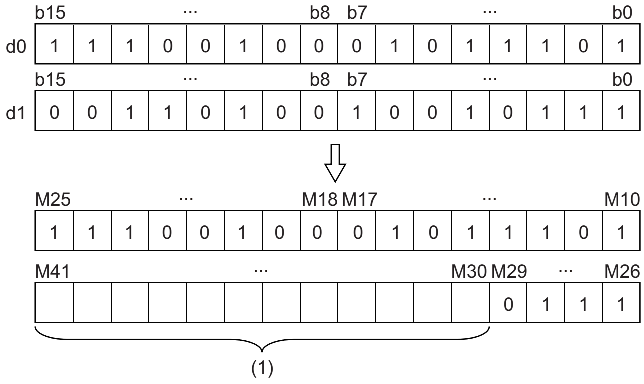
When a bit device with digit specification is specified in the source of an instruction, 0 is stored in the bits, which follow the bit for which digit specification is made in the source, in the word device of the destination.

Ladder example	Processing
• 32-bit data instruction 	

■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 bit devices after the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device 	 (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.

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

- Long timer (LT)
- Long retentive timer (LST)
- Long counter (LC)
- Long index register (LZ)

■Double word type label

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

Real number data (floating-point data)

Data size and data range

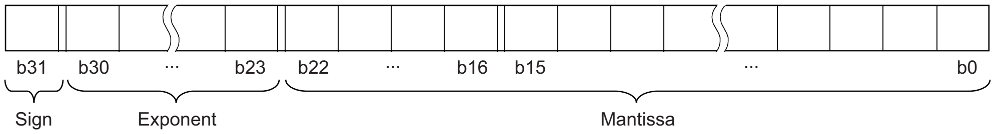
Real number data includes single-precision 32-bit real number data and double-precision 64-bit real number data.
Real number data can be stored only in devices other than bit devices or in single-precision or double-precision real data type labels.

Data name		Data size	Value range
Single-precision real number data (single-precision floating-point data)	Positive number	32 bits (2 word)	$2^{-126} \leq$ real number $< 2^{126}$
	Zero		0
	Negative number		$-2^{128} <$ real number $\leq -2^{-126}$
Double-precision real number data (double-precision floating-point data)	Positive number	64 bits (4 word)	$2^{-1022} \leq$ real number $< 2^{1024}$
	Zero		0
	Negative number		$-2^{1024} <$ real number $\leq -2^{-1022}$

Configuration of single-precision real number data

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

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



■Sign (1 bit)

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

■Mantissa (23 bits)

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

■Exponent (8 bits)

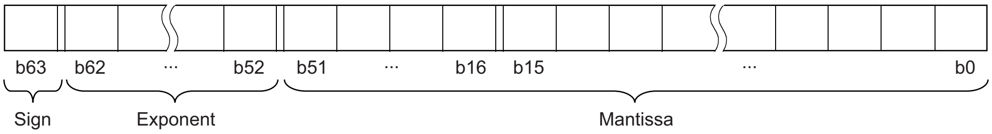
An exponent means N of $1.XXXXX\cdots\times 2^N$ representing a single-precision real number in binary. The following table shows the relationships between the exponent value and N of a single-precision real number.

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

Configuration of double-precision real number data

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

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



■Sign (1 bit)

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

■Mantissa (52 bits)

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

■Exponent (11 bits)

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

Exponent (b52 to b62)	7FFH	7FEH	7FDH	···	401H	400H	3FFH	3FEH	···	02H	01H	00H
N	Not used	1023	1022	···	2	1	0	-1	···	-1021	-1022	Not used

Precautions

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

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

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

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

Ex.

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

Ex.

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

■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.

Point

The monitor function of the engineering tool can monitor real number data of CPU modules.

To represent "0" in real number data, set all numbers in each of the following range to 0.

- Single-precision real number data: b0 to b31
- Double-precision real number data: b0 to b63

The setting range of real number data is as follows.*1

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

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

---

*1 For the operations to be performed when an overflow or underflow occurs or when a special value is input, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)

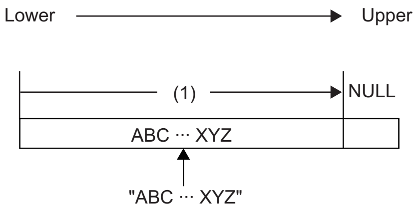
String data

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)
Unicode character string	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

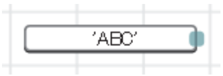
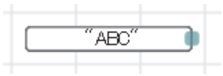
The following shows the notation of character strings in ladder programs.

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

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

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

The following shows the notation of character strings in FBD/LD programs.

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

Data range

The following table summarizes the ranges of character string data.

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

[*1] For the Unicode character string, 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 Unicode character strings
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 D0	Image of storing character string data from word type label array arrayA[0]
Null character string (("") or (''))	D0 NULL ¦ NULL	arrayA[0] NULL ¦ NULL
ABC	D0 B ¦ A D1 NULL ¦ C	arrayA[0] B ¦ A arrayA[1] NULL ¦ C
ABCD	D0 B ¦ A D1 D ¦ C D2 NULL ¦ NULL	arrayA[0] B ¦ A arrayA[1] D ¦ C arrayA[2] NULL ¦ NULL

■Unicode character strings

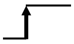
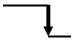
In each Unicode character string storage image, "NULL" indicates a NULL code (0000H).

Character string to be stored	Image of storing character string data from D0	Image of storing character string data from word type label array arrayA[0]
Null character string ("")	D0 NULL	arrayA[0] NULL
ABCD	D0 A D1 B D2 C D3 D D4 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.
Off		An instruction is executed during off. It is executed only while the precondition of the instruction is off. When the precondition is on, the instruction is not executed.
Falling edge		An instruction is executed one time when turned off. It is executed only once on the falling edge (on to off) of the precondition of the instruction and is no longer executed later even when the condition turns off.
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) or function block diagram/ladder diagram (FBD/LD), 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) or function block diagram/ladder diagram (FBD/LD), 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 High-speed Instruction Processing

Subset processing

Subset processing can reduce the number of steps or speed up the instruction processing when the device and label specified by each operand of an instruction satisfy the specified conditions.
Instruction symbols and the number of operands do not change whether subset processing is applicable or not.

Instructions that support subset processing

For the availability of subset processing for each instruction, refer to the following.

☞ Page 2031 Number of Basic Steps and Availability of Subset Processing

Operand condition

The conditions that the operands need to satisfy to enable subset processing are shown.

■When a device is specified in an operand

The following table lists the conditions that an operand which specifies a device needs to satisfy.

Data type of operand	Condition[1]
Bit data	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")[2] • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device[3] • File register • Local device • Refresh data register
Signed 16-bit data Unsigned 16-bit data Signed 32-bit data Unsigned 32-bit data	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")[2] • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device[3] • Index register • File register • Local device • Refresh data register • Constant (decimal, hexadecimal)
Single-precision real number	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")[2] • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device[3] • Index register • File register • Local device • Refresh data register • Constant (single-precision real number)

[1] Including the cases where bit numbers, digits, indirect addresses, or index-modified devices are specified

[2] True when U3En\G□, U3En\G□Zn, U3En\HG□, or U3En\HG□Zn is used in the CPU buffer memory access device of the host CPU module.

[3] True when U3En\HG□ or U3En\HG□Zn is used in the CPU buffer memory access device of another CPU module.

■When the label assigned to a device is specified in an operand

The same conditions as those applicable when a device is specified in an operand apply.

■When the label assigned to each label area is specified in an operand

When the label assigned to a label area or latch label area is specified in an operand, any instruction which supports subset processing performs subset processing regardless of the data type of the operand. (Including the cases where bit numbers or digits are specified.)

1.5 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[*1]	Error code
An I/O number which is out of range (other than 000H to FFFH and 3E0H to 3E3H) is specified.	2800H
An I/O number which corresponds to no module is specified.	2801H
An I/O number of the module that cannot be specified by using the instruction is specified.	2803H
A network number which is out of range (1 to 239) is specified.	2804H
A network number which does not exist is specified.	2805H
• The device or label specified by the instruction exceeds the available range. • The file register is accessed while the file register is not set in the file setting of a CPU parameter or the file register to be used in the program is not set.	2820H
• The range of the buffer memory of the module specified by the instruction is exceeded. • The module specified by the instruction does not have buffer memory.	2823H

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

Checking the ranges of instruction runtime devices and labels

Checking the ranges of devices and labels

When a device or label is specified in an instruction, no range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant device or label.

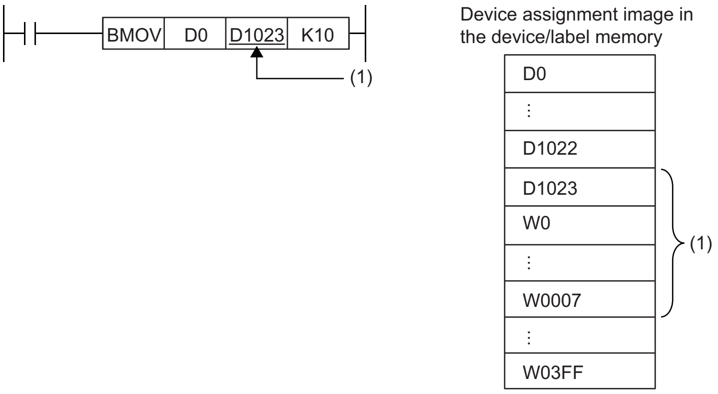
If a range exceeding that of the relevant device or label is specified, no error is detected but data is written to other device or label areas.

However, an error (error code: 2820) occurs if data is written to outside the areas.

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

Ex.

When a global device is specified and W0 is assigned after D1023 in the device/label memory



(1) The transfer destination is in the range corresponding to D1023 to D1032. Even though the range D1024 to D1032 does not exist, the data are written and the data in W0 to W0007 are overwritten.

Checking the range of file register

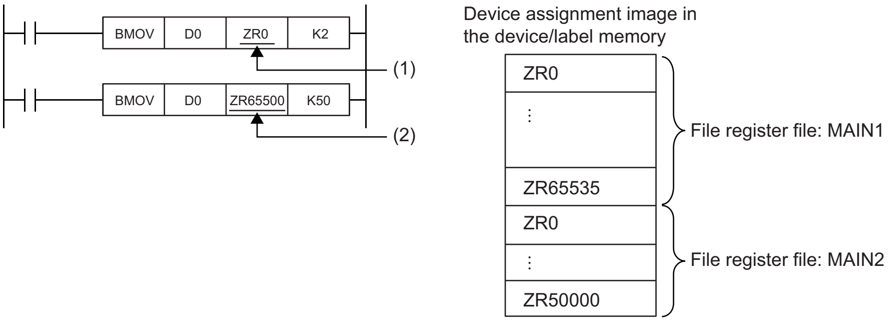
When a file register is specified in an instruction, a range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant file register.

If a range exceeding that of the file register (ZR) is specified, an error (error code: 2820) occurs.

If a range exceeding that of the file register of the block number used by the file register (R) is specified, an error (error code: 2820) occurs.

Ex.

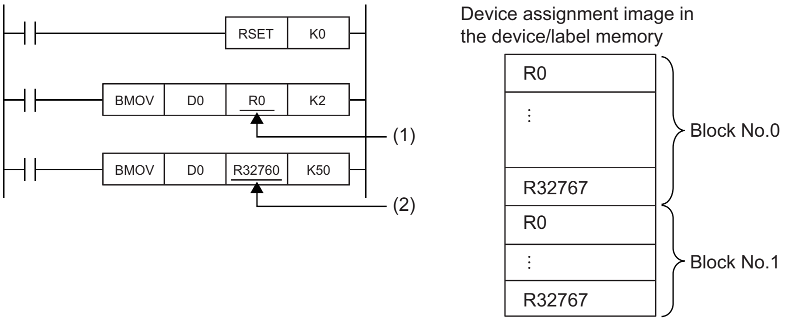
When a file register (ZR) is specified



(1) The transfer destination is in the range of file register MAIN1. Data is written to ZR0 and ZR1.
(2) The transfer destination is out of the range of file register MAIN1. An error occurs because the area range of file register MAIN1 is exceeded.

Ex.

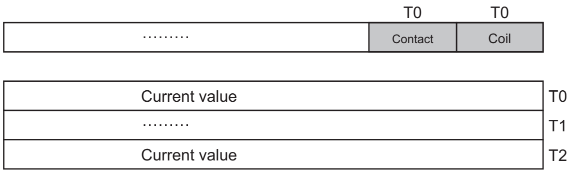
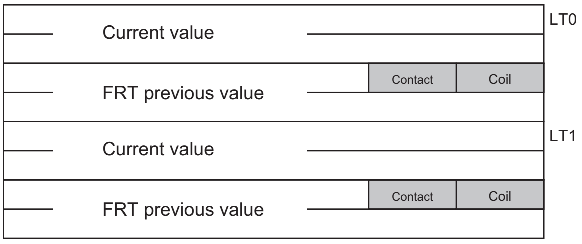
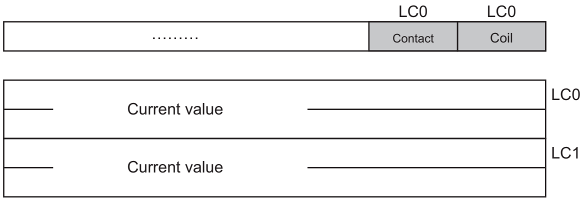
When a file register (R) is specified



(1) The transfer destination is in the range of the R device of block number 0. Data is written to R0 and R1.
(2) The transfer destination is out of the range of the R device of block number 0. An error occurs because the area range of the R device of block number 0.

Operation when a long timer or long retentive timer device is used

When the data to be handled exceeds the width (32 bits) of the current value, the long timer or long retentive timer operates by using not only the area of the current value but also the areas of the previous value, contact, and coil.

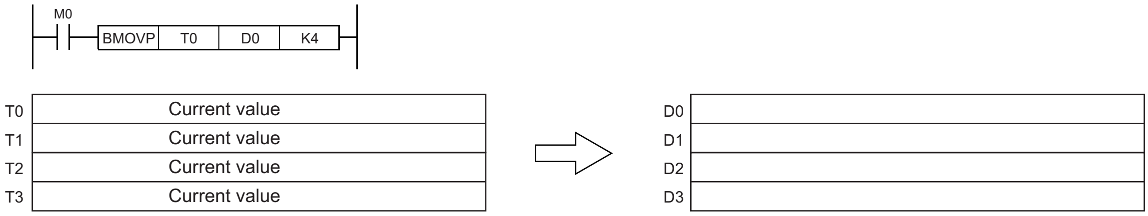
Device	Configuration
Timer (T)	
Retentive timer (ST)	
Counter (C)	
Long timer (LT)	
Long retentive timer (LST)	
Long counter (LC)	

When the BMOV instruction is used to batch-transfer current values, current values alone cannot be batch-transferred. Batch-transfer the current values, contacts, and coils altogether and, after the batch transfer is finished, use only the current values. When the DMOV instruction is used to batch-transfer current values, repeat the transfer of the current values alone using the FOR to NEXT instruction.

Ex.

To batch-transfer the current values of the timer device

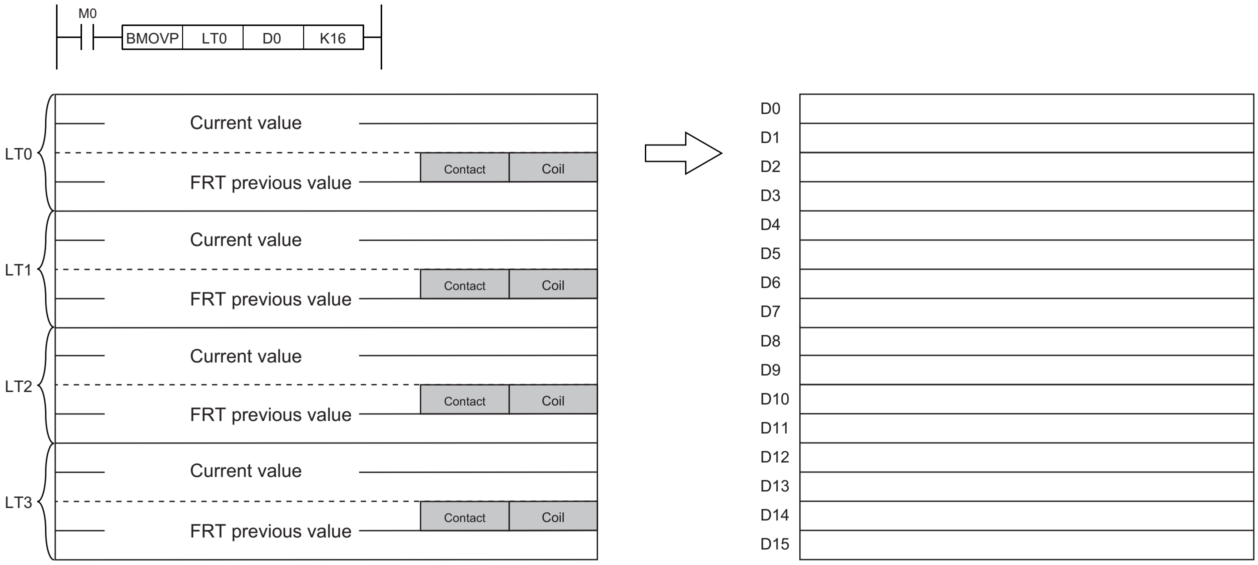
When the BMOV instruction is used, only current values are batch-transferred.



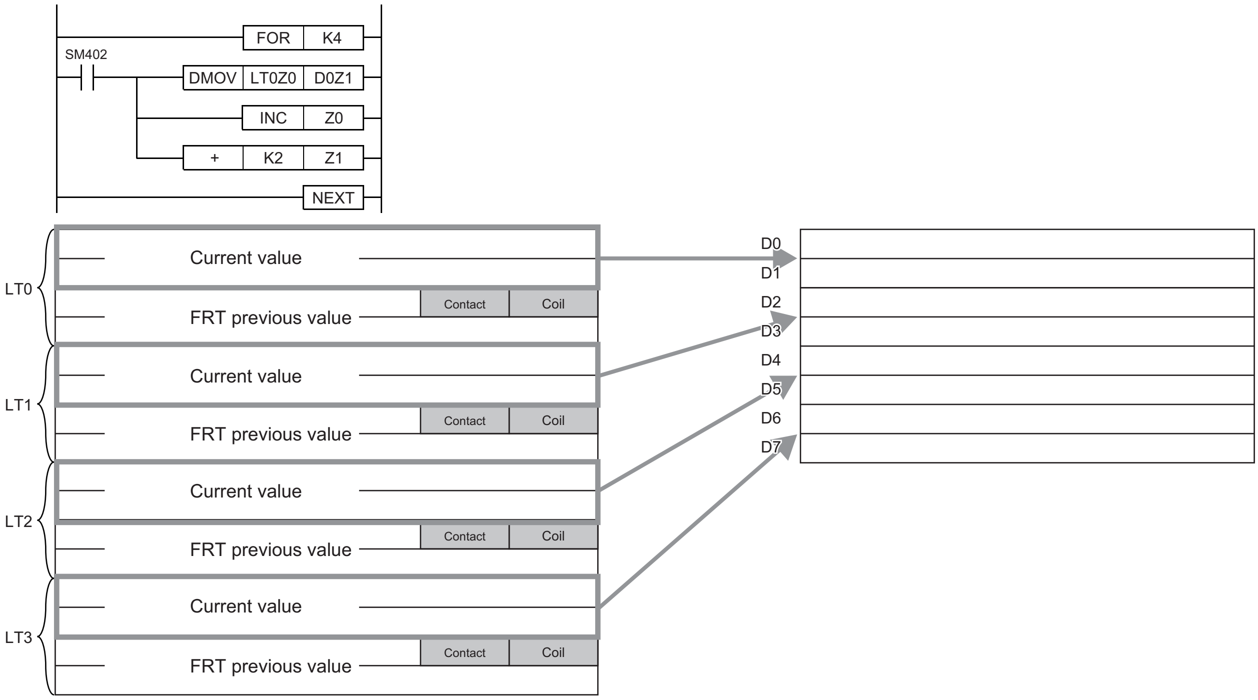
Ex.

To batch-transfer the current values of the long timer device

When the BMOV instruction is used, all current values, contacts, and coils are batch-transferred.



When the DMOV instruction is used, only current values are batch-transferred.



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

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

Point

In safety programs executed by the Safety CPU, replace some words as follows:

- "Scan" → "Safety cycle processing"
- "X0" → "SA\X0", "X1" → "SA\X1", "M0" → "SA\M0"
- "END" on the rising edge of X0 (in Figures) → "Safety cycle processing start"[*1]
- "END" on the falling edge of X0 (in Figures) → "Safety cycle processing end"[*1]

[*1] For the PLF instruction, replace "END" with "safety cycle processing start" regardless of the X0 status (rising edge or falling edge).

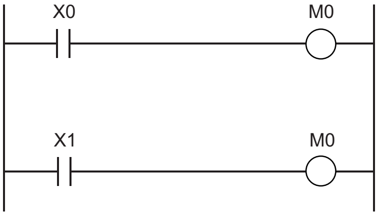
For OUT instructions of the same device

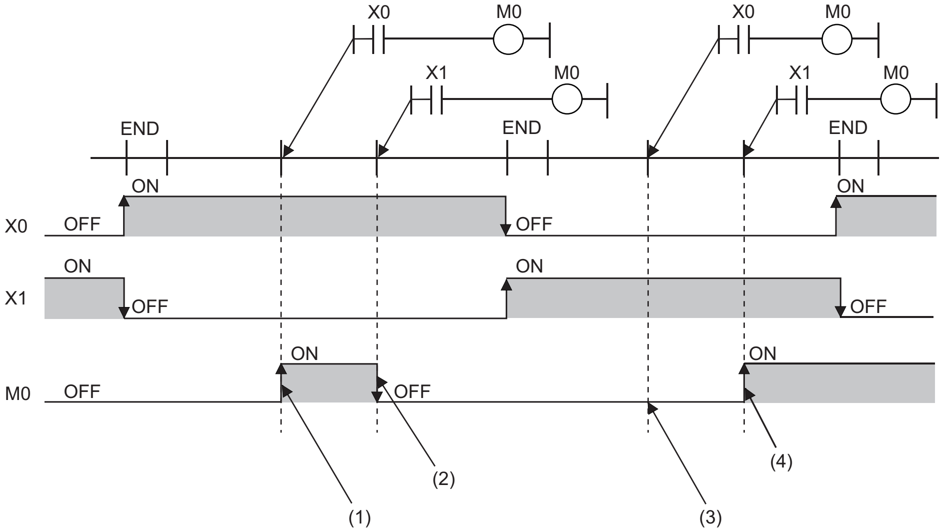
More than one OUT instruction of the same device must not be issued during one scan.

Otherwise, the specified device turns on or off, depending on the operation result up to each OUT instruction while it is in execution.

In this case, the device may turn on/off during one scan because the on/off state of the specified device is determined during execution of each OUT instruction.

The following figure shows the behavior arising when a circuit turning on/off the same internal relay (M0) is created with input X0 and X1.





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

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

If SET/RST instructions of the same device are used

■For SET instructions

The SET instruction turns on the specified device if the execution command is on, and causes no operation if it is off.

Thus, if two or more SET instructions of the same device are executed during one scan, the specified device turns on even if one execution command is on.

■For RST instructions

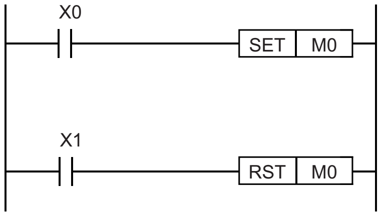
The RST instruction turns on the specified device if the execution command is off, and causes no operation if it is off.

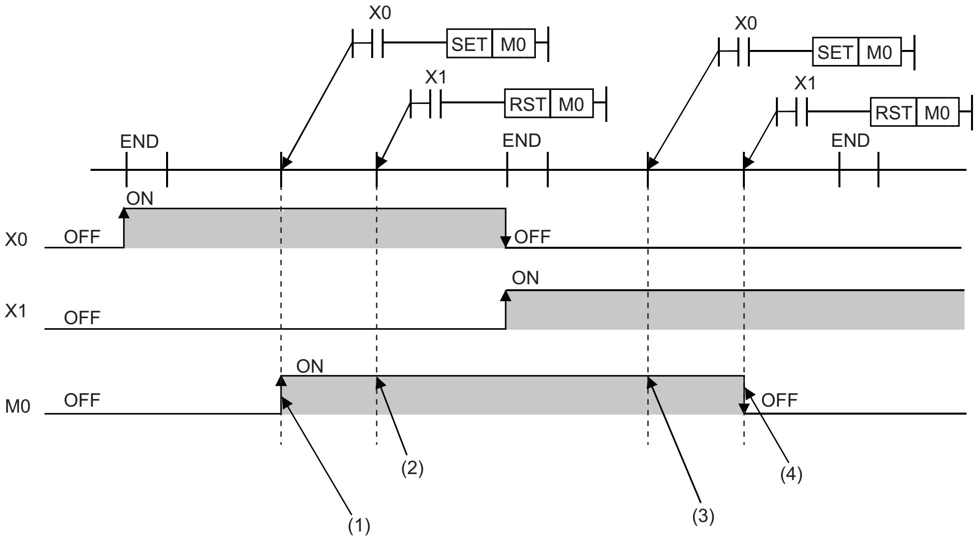
Thus, if two or more RST instructions of the same device are executed during one scan, the specified device turns on even if one execution command is off.

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

If the SET and RST instructions of the same device exist in one scan, the SET instruction turns on the specified device if the execution command is on, and turns off the specified device if it is on.

If both the SET and RST instructions are off, the on/off state of the specified device will be unchanged.





(1) Since X0 is on, M0 turns on.

(2) Since X1 is off, M0 remains on. (The RST instruction results in non-processing.)

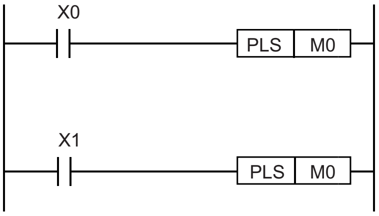
(3) Since X0 is off, M0 remains on. (The SET instruction results in non-processing.)

(4) Since X1 is on, M0 turns off.

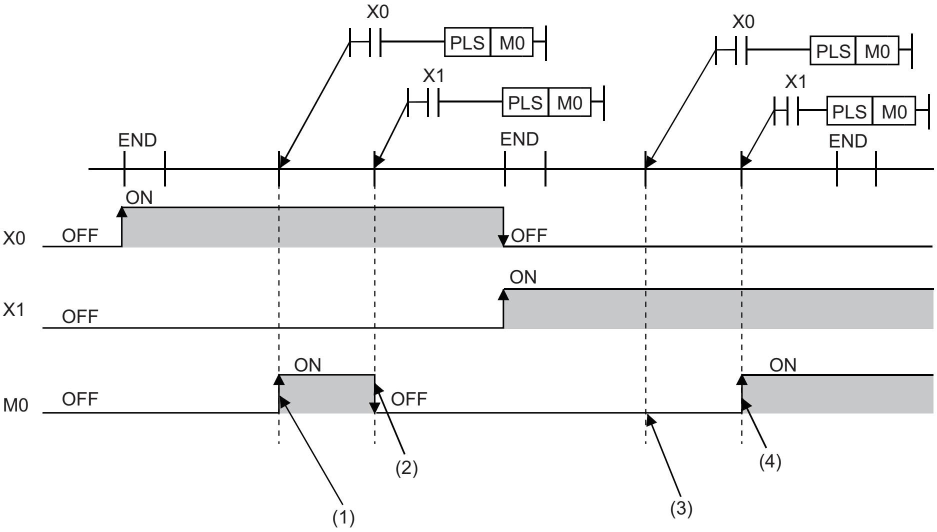
If output (Y) is specified using a SET/RST instruction, the on/off state of the last SET/RST instruction executed during the one scan will be output.

If PLS instructions of the same device are used

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

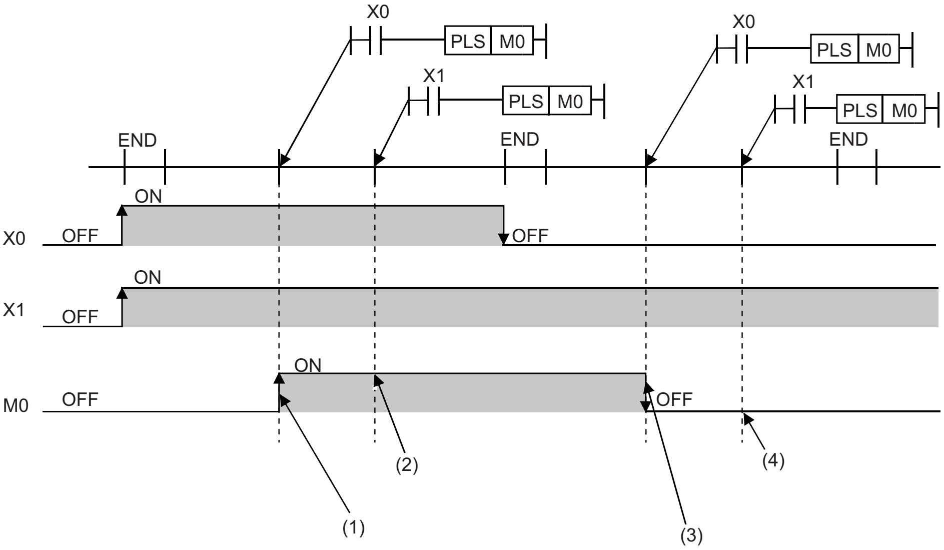


- If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)



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

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

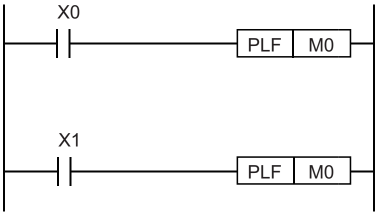


(1) Since X0 turns on, M0 turns on.
(2) Since X1 turns on, M0 remains off.
(3) Since X0 is other than turning on, M0 turns off.
(4) Since X1 is other than turning on, M0 remains off.

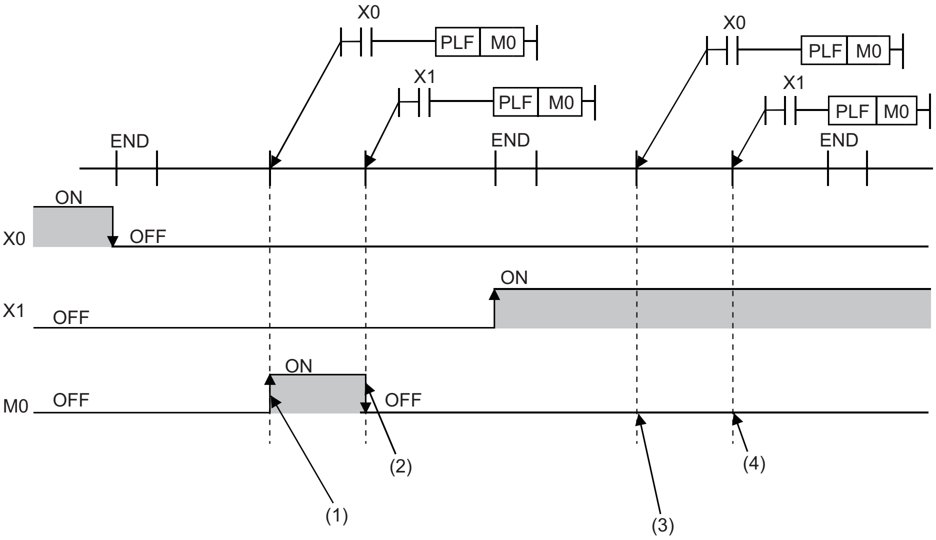
If output (Y) is specified using a PLS instruction, the on/off state of the last PLS instruction executed during the one scan will be output.

If PLF instructions of the same device are used

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

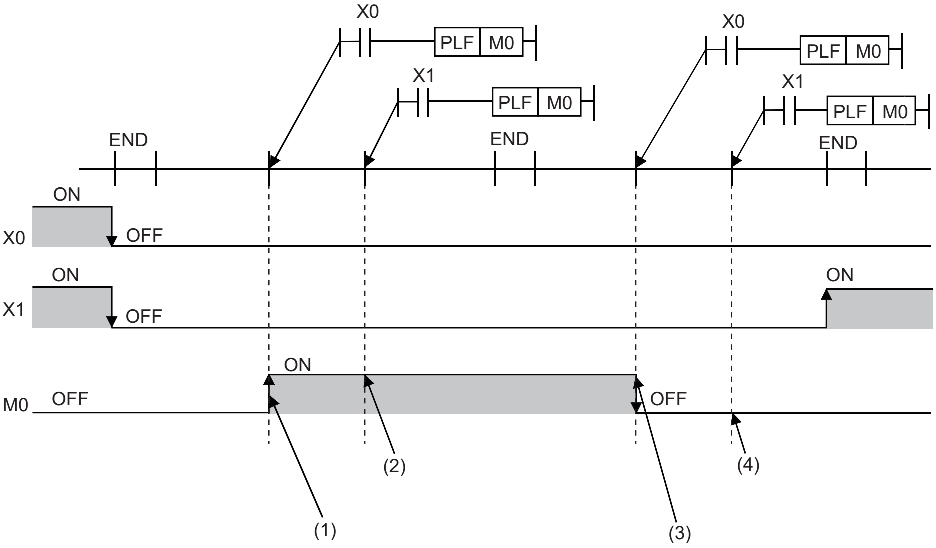


- If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)



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

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



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

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

Restrictions on using file registers

When a file register is specified for the refresh device, note the following restrictions.

When a file register having the same name as a program is specified

If the use of a file register having the same name as a program is specified in the parameter, refresh cannot be performed correctly. When a file register having the same name of a program is used, data is refreshed by the file register having the same name of the program that has been set at the final number in the program settings.

To read or write refresh data, use the QDRSET instruction to switch to the corresponding file register and specify it.

If the file name or drive number is changed by the QDRSET instruction

If the file register file name or drive number is changed by the QDRSET instruction, the setting file is linked immediately before refresh.

To read or write refresh data, specify it in the setting file immediately before refresh.

When the block number is changed by the RSET instruction

When the block number is changed by the RSET instruction, note the following.

- Data is refreshed by the file register (R) of the new block number.
- Data is refreshed by the file register (R) of the block number immediately before refresh.

To read or write refresh data, specify the block number immediately before refresh.

MEMO

PART 2

LISTS OF INSTRUCTIONS AND FUN/FB

This part consists of the following chapters.

2 CPU MODULE 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

Contact instructions

■Operation start, series connection, parallel connection

Instruction symbol	Processing details	Reference
LD	Outputs the on/off information of the specified device as the operation result. (Normally open contact operation start instruction)	Page 160 LD, LDI, AND, ANI, OR, ORI
LDI	Outputs the on/off information of the specified device as the operation result. (Normally closed contact operation start instruction)	
AND	Performs an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Normally open contact series connection instruction)	
ANI	Performs an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Normally closed contact series connection instruction)	
OR	Performs an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Single normally open contact parallel connection instruction)	
ORI	Performs an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Single normally closed contact parallel connection instruction)	

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

Instruction symbol	Processing details	Reference
LDP	Turns on only at the rising edge (off to on) of the specified bit device. (Rising edge pulse operation start instruction)	Page 162 LDP, LDF, ANDP, ANDF, ORP, ORF
LDF	Turns on only at the falling edge (on to off) of the specified bit device. (Falling edge pulse operation start instruction)	
ANDP	Performs an AND operation with the previous operation result. (Rising edge pulse series connection instruction)	
ANDF	Performs an AND operation with the previous operation result. (Falling edge pulse series connection instruction)	
ORP	Performs an OR operation with the previous operation result. (Rising edge pulse parallel connection instruction)	
ORF	Performs an OR operation with the previous operation result. (Falling edge pulse parallel connection instruction)	

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

Instruction symbol	Processing details	Reference
LDPI	Turns on when the specified device is off, on, or at the falling edge (on to off). (Rising edge pulse NOT operation start instruction)	Page 165 LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI
LDFI	Turns on when the specified device is at the rising edge (off to on), off, or on. (Falling edge pulse NOT operation start instruction)	
ANDPI	Performs an AND operation with the previous operation result. (Rising edge pulse NOT series connection instruction)	
ANDFI	Performs an AND operation with the previous operation result. (Falling edge pulse NOT series connection instruction)	
ORPI	Performs an OR operation with the previous operation result. (Rising edge pulse NOT parallel connection instruction)	
ORFI	Performs an OR operation with the previous operation result. (Falling edge pulse NOT parallel connection instruction)	

Association instructions

■Ladder block series/parallel connection

Instruction symbol	Processing details	Reference
ANB	Performs AND operations between logical blocks (series connection between logical blocks)	Page 168 ANB, ORB
ORB	Performs OR operations between logical blocks (series connection between logical blocks)	

■Storing/reading/clearing the operation result

Instruction symbol	Processing details	Reference
MPS	Stores the operation result (on/off) immediately before the MPS instruction.	Page 169 MPS, MRD, MPP
MRD	Reads the operation result stored by using the MPS instruction.	
MPP	Clears the operation result stored by using the MPS instruction.	

■Inverting the operation result

Instruction symbol	Processing details	Reference
INV	Inverts the operation result up to just before the INV instruction.	Page 171 INV

■Converting the operation result into a pulse

Instruction symbol	Processing details	Reference
MEP	Turns on at the rising edge (off to on) of the operation result up to the MEP instruction.	Page 172 MEP, MEF
MEF	Turns on at the falling edge (on to off) of the operation result up to the MEF instruction.	

■Converting the edge relay operation result into a pulse

Instruction symbol	Processing details	Reference
EGP	Stores the operation result up to the EGP instruction in the edge relay (V). The instruction turns on at the rising edge (off to on) of the operation result.	Page 173 EGP, EGF
EGF	Stores the operation result up to the EGF instruction in the edge relay (V). The instruction turns on at the falling edge (on to off) of the operation result.	

Output instructions

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

Instruction symbol	Processing details	Reference
OUT	Outputs the operation result to the specified device.	Page 175 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 177 OUT T, OUTH T, OUT ST, OUTH ST
OUTH T		
OUT ST		
OUTH ST		
OUT LT		Page 181 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 184 OUT C
OUT LC		Page 186 OUT LC

■Annunciator

Instruction symbol	Processing details	Reference
OUT F	Outputs the operation result up to the OUT F instruction to the specified annunciator.	Page 188 OUT F

■Setting devices (excluding annunciator)

Instruction symbol	Processing details	Reference
SET	Turns on the specified bit.	Page 189 SET

■Resetting devices (excluding annunciator)

Instruction symbol	Processing details	Reference
RST	Turns off the specified device. For the timer and counter, the instruction clears the current value to 0 and turns off the contact or coil.	Page 191 RST

■Setting/resetting annunciator

Instruction symbol	Processing details	Reference
SET F	Turns on the specified annunciator.	Page 193 SET F
RST F	Turns off the specified annunciator.	Page 195 RST F

■Rising/falling edge output

Instruction symbol	Processing details	Reference
PLS	Turns on the specified device for one scan on the rising edge (off to on) of the execution command.	Page 197 PLS
PLF	Turns on the specified device for one scan on the falling edge (on to off) of the execution command.	Page 199 PLF

■Inverting the bit device output

Instruction symbol	Processing details	Reference
FF	Inverts the status of the specified device.	Page 201 FF

■Converting the direct access output into a pulse

Instruction symbol	Processing details	Reference
DELTA	Converts the specified direct access output (DY) into pulse output.	Page 203 DELTA(P)
DELTAP		

Shift instructions

■Shifting bit devices

Instruction symbol	Processing details	Reference
SFT	Shifts the on/off state of the device area just before the one specified to the specified device area, and turns off the shift source device.	Page 205 SFT(P)
SFTP		

Master control instructions

■Setting/resetting a master control

Instruction symbol	Processing details	Reference
MC	Starts a master control.	Page 207 MC, MCR
MCR	Ends a master control.	

Termination instructions

■Ending the main routine program

Instruction symbol	Processing details	Reference
FEND	Separates the main routine program from subroutine programs and interrupt programs in a program file.	Page 211 FEND

■Ending the sequence program

Instruction symbol	Processing details	Reference
END	Indicates the end of a program.	Page 213 END

Stop instruction

■Stopping the sequence program

Instruction symbol	Processing details	Reference
STOP	Stops the operation of the CPU module. (The operation of this instruction is the same as setting the switch of the CPU module to the STOP position.)	Page 215 STOP

No operation instruction

■No operation

Instruction symbol	Processing details	Reference
NOP	Inserts a space for debugging.	Page 216 NOP
NOPLF	This instruction is a no-operation instruction and has no impact on the previous operations.	Page 217 NOPLF

2.2 Basic Instructions

Comparison operation instructions

■Comparing 16-bit binary data

Instruction symbol	Processing details	Reference
LD=, AND=, OR=	Compares the two sets of 16-bit binary data specified. (Devices are used as normally open contacts.)	Page 218 LD□(_U), AND□(_U), OR□(_U)
LD=_U, AND=_U, OR=_U		
LD<>, AND<>, OR<>		
LD<>_U, AND<>_U, OR<>_U		
LD>, AND>, OR>		
LD>_U, AND>_U, OR>_U		
LD<=, AND<=, OR<=		
LD<=_U, AND<=_U, OR<=_U		
LD<, AND<, OR<		
LD<_U, AND<_U, OR<_U		
LD>=, AND>=, OR>=		
LD>=_U, AND>=_U, OR>=_U		

■Comparing 32-bit binary data

Instruction symbol	Processing details	Reference
LDD=, ANDD=, ORD=	Compares the two sets of 32-bit binary data specified. (Devices are used as normally open contacts.)	Page 220 LDD□(_U), ANDD□(_U), ORD□(_U)
LDD=_U, ANDD=_U, ORD=_U		
LDD<>, ANDD<>, ORD<>		
LDD<>_U, ANDD<>_U, ORD<>_U		
LDD>, ANDD>, ORD>		
LDD>_U, ANDD>_U, ORD>_U		
LDD<=, ANDD<=, ORD<=		
LDD<=_U, ANDD<=_U, ORD<=_U		
LDD<, ANDD<, ORD<		
LDD<_U, ANDD<_U, ORD<_U		
LDD>=, ANDD>=, ORD>=		
LDD>=_U, ANDD>=_U, ORD>=_U		

■Outputting a comparison result of 16-bit binary data

Instruction symbol	Processing details	Reference
CMP	Compares the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 222 CMP(P)(_U)
CMPP		
CMP_U		
CMPP_U		

■Outputting a comparison result of 32-bit binary data

Instruction symbol	Processing details	Reference
DCMP	Compares the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 224 DCMP(P)(_U)
DCMPP		
DCMP_U		
DCMPP_U		

■Outputting a band comparison result of 16-bit binary data

Instruction symbol	Processing details	Reference
ZCP	Compares the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 226 ZCP(P)(_U)
ZCPP		
ZCP_U		
ZCPP_U		

■Outputting a band comparison result of 32-bit binary data

Instruction symbol	Processing details	Reference
DZCP	Compares the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 228 DZCP(P)(_U)
DZCPP		
DZCP_U		
DZCPP_U		

■Comparing 16-bit binary block data

Instruction symbol	Processing details	Reference
BKCMP=, BKCMP<>, BKCMP>, BKCMP<=, BKCMP<, BKCMP>=	Compares the two sets of 16-bit binary block data specified.	Page 230 BKCMP□(P)(_U)
BKCMP=P, BKCMP<>P, BKCMP>P, BKCMP<=P, BKCMP<P, BKCMP>=P		
BKCMP=_U, BKCMP<>_U, BKCMP>_U, BKCMP<=_U, BKCMP<_U, BKCMP>=_U		
BKCMP=P_U, BKCMP<>P_U, BKCMP>P_U, BKCMP<=P_U, BKCMP<P_U, BKCMP>=_U		

■Comparing 32-bit binary block data

Instruction symbol	Processing details	Reference
DBKCMP=, DBKCMP<>, DBKCMP>, DBKCMP<=, DBKCMP<, DBKCMP>=	Compares the two sets of 32-bit binary block data specified.	Page 232 DBKCMP□(P)(_U)
DBKCMP=P, DBKCMP<>P, DBKCMP>P, DBKCMP<=P, DBKCMP<P, DBKCMP>=P		
DBKCMP=_U, DBKCMP<>_U, DBKCMP>_U, DBKCMP<=_U, DBKCMP<_U, DBKCMP>=_U		
DBKCMP=P_U, DBKCMP<>P_U, DBKCMP>P_U, DBKCMP<=P_U, DBKCMP<P_U, DBKCMP>=P_U		

Arithmetic operation instructions

■Adding/subtracting 16-bit binary data

Instruction symbol	Processing details	Reference
+	Adds the two sets of 16-bit binary data specified. (Two operands)	Page 235 +(P)(_U) [when two operands are set]
+P		
+_U		
+P_U		
+	Adds the two sets of 16-bit binary data specified. (Three operands)	Page 237 +(P)(_U) [when three operands are set]
+P		
+_U		
+P_U		
-	Performs subtraction between the two sets of 16-bit binary data specified. (Two operands)	Page 239 -(P)(_U) [when two operands are set]
-P		
-_U		
-P_U		
-	Performs subtraction between the two sets of 16-bit binary data specified. (Three operands)	Page 241 -(P)(_U) [when three operands are set]
-P		
-_U		
-P_U		

■Adding/subtracting 32-bit binary data

Instruction symbol	Processing details	Reference
D+	Adds the two sets of 32-bit binary data specified. (Two operands)	Page 243 D+(P)(_U) [when two operands are set]
D+P		
D+_U		
D+P_U		
D+	Adds the two sets of 32-bit binary data specified. (Three operands)	Page 245 D+(P)(_U) [when three operands are set]
D+P		
D+_U		
D+P_U		
D-	Performs subtraction between the two sets of 32-bit binary data specified. (Two operands)	Page 247 D-(P)(_U) [when two operands are set]
D-P		
D-_U		
D-P_U		
D-	Performs subtraction between the two sets of 32-bit binary data specified. (Three operands)	Page 249 D-(P)(_U) [when three operands are set]
D-P		
D-_U		
D-P_U		

■Multiplying/dividing 16-bit binary data

Instruction symbol	Processing details	Reference
*	Multiplies the two sets of 16-bit binary data specified.	Page 251 *(P)(_U)
*P		
*_U		
*P_U		
/	Performs division between the two sets of 16-bit binary data specified.	Page 253 /(P)(_U)
/P		
/_U		
/P_U		

■Multiplying/dividing 32-bit binary data

Instruction symbol	Processing details	Reference
D*	Multiplies the two sets of 32-bit binary data specified.	Page 255 D*(P)(_U)
D*P		
D*_U		
D*P_U		
D/	Performs division between the two sets of 32-bit binary data specified.	Page 257 D/(P)(_U)
D/P		
D/_U		
D/P_U		

■Adding/subtracting BCD 4-digit data

Instruction symbol	Processing details	Reference
B+	Adds the two sets of BCD 4-digit data specified. (Two operands)	Page 259 B+(P) [when two operands are set]
B+P		
B+	Adds the two sets of BCD 4-digit data specified. (Three operands)	Page 260 B+(P) [when three operands are set]
B+P		
B-	Performs subtraction between the two sets of BCD 4-digit data specified. (Two operands)	Page 262 B-(P) [when two operands are set]
B-P		
B-	Performs subtraction between the two sets of BCD 4-digit data specified. (Three operands)	Page 264 B-(P) [when three operands are set]
B-P		

■Adding/subtracting BCD 8-digit data

Instruction symbol	Processing details	Reference
DB+	Adds the two sets of BCD 8-digit data specified. (Two operands)	Page 266 DB+(P) [when two operands are set]
DB+P		
DB+	Adds the two sets of BCD 8-digit data specified. (Three operands)	Page 268 DB+(P) [when three operands are set]
DB+P		
DB-	Performs subtraction between the two sets of BCD 8-digit data specified. (Two operands)	Page 270 DB-(P) [when two operands are set]
DB-P		
DB-	Performs subtraction between the two sets of BCD 8-digit data specified. (Three operands)	Page 272 DB-(P) [when three operands are set]
DB-P		

■Multiplying/dividing BCD 4-digit data

Instruction symbol	Processing details	Reference
B*	Multiplies the two sets of BCD 4-digit data specified.	Page 274 B*(P)
B*P		
B/	Performs division between the two sets of BCD 4-digit data specified.	Page 276 B/(P)
B/P		

■Multiplying/dividing BCD 8-digit data

Instruction symbol	Processing details	Reference
DB*	Multiplies the two sets of BCD 8-digit data specified.	Page 278 DB*(P)
DB*P		
DB/	Performs division between the two sets of BCD 8-digit data specified.	Page 280 DB/(P)
DB/P		

■Adding/subtracting 16-bit binary block data

Instruction symbol	Processing details	Reference
BK+	Adds the two 16-bit binary data blocks specified.	Page 282 BK+(P)(_U)
BK+P		
BK+_U		
BK+P_U		
BK-	Performs subtraction between the two 16-bit binary data blocks specified.	Page 284 BK-(P)(_U)
BK-P		
BK-_U		
BK-P_U		

■Adding/subtracting 32-bit binary block data

Instruction symbol	Processing details	Reference
DBK+	Adds the two 32-bit binary data blocks specified.	Page 286 DBK+(P)(_U)
DBK+P		
DBK+_U		
DBK+P_U		
DBK-	Performs subtraction between the two 32-bit binary data blocks specified.	Page 289 DBK-(P)(_U)
DBK-P		
DBK-_U		
DBK-P_U		

■Incrementing/decrementing 16-bit binary data

Instruction symbol	Processing details	Reference
INC	Increments the specified 16-bit binary data by one.	Page 292 INC(P)(_U)
INCP		
INC_U		
INCP_U		
DEC	Decrements the specified 16-bit binary data by one.	Page 294 DEC(P)(_U)
DECP		
DEC_U		
DECP_U		

■Incrementing/decrementing 32-bit binary data

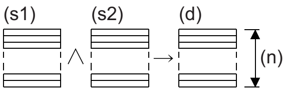
Instruction symbol	Processing details	Reference
DINC	Increments the specified 32-bit binary data by one.	Page 296 DINC(P)(_U)
DINCP		
DINC_U		
DINCP_U		
DDEC	Decrements the specified 32-bit binary data by one.	Page 298 DDEC(P)(_U)
DDECP		
DDEC_U		
DDECP_U		

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. (Two operands)	Page 300 WAND(P) [when two operands are set]
WANDP		
WAND	Performs an AND operation on the two sets of 16-bit binary data specified. (Three operands)	Page 302 WAND(P) [when three operands are set]
WANDP		
DAND	Performs an AND operation on the two sets of 32-bit binary data specified. (Two operands)	Page 304 DAND(P) [when two operands are set]
DANDP		
DAND	Performs an AND operation on the two sets of 32-bit binary data specified. (Three operands)	Page 306 DAND(P) [when three operands are set]
DANDP		

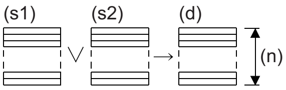
■Performing an AND operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKAND	Performs an AND operation on the two 16-bit binary data blocks specified.	Page 308 BKAND(P)
BKANDP		

■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. (Two operands)	Page 310 WOR(P) [when two operands are set]
WORP		
WOR	Performs an OR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 312 WOR(P) [when three operands are set]
WORP		
DOR	Performs an OR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 314 DOR(P) [when two operands are set]
DORP		
DOR	Performs an OR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 316 DOR(P) [when three operands are set]
DORP		

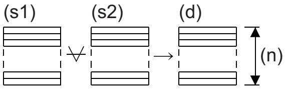
■Performing an OR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKOR	Performs an OR operation on the two 16-bit binary data blocks specified.	Page 318 BKOR(P)
BKORP		

■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. (Two operands)	Page 320 WXOR(P) [when two operands are set]
WXORP		
WXOR	Performs an XOR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 322 WXOR(P) [when three operands are set]
WXORP		
DXOR	Performs an XOR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 324 DXOR(P) [when two operands are set]
DXORP		
DXOR	Performs an XOR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 326 DXOR(P) [when three operands are set]
DXORP		

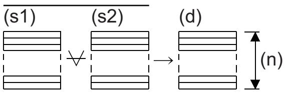
■Performing an XOR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKXOR	Performs an XOR operation on the two 16-bit binary data blocks specified. 	Page 328 BKXOR(P)
BKXORP		

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

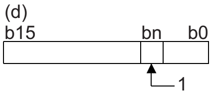
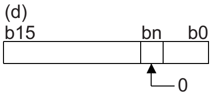
Instruction symbol	Processing details	Reference
WXNR	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Two operands)	Page 330 WXNR(P) [when two operands are set]
WXNRP		
WXNR	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 332 WXNR(P) [when three operands are set]
WXNRP		
DXNR	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 334 DXNR(P) [when two operands are set]
DXNRP		
DXNR	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 336 DXNR(P) [when three operands are set]
DXNRP		

■Performing an XNOR operation on 16-bit block data

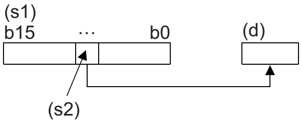
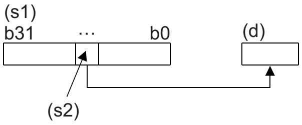
Instruction symbol	Processing details	Reference
BKXNR	Performs an XNOR operation on the two 16-bit binary data blocks specified. 	Page 338 BKXNR(P)
BKXNRP		

Bit processing instructions

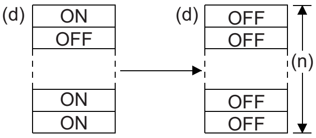
■Setting/resetting a bit in the word device

Instruction symbol	Processing details	Reference
BSET BSETP	Sets the 'n'th bit in the specified word device to 1. 	Page 340 BSET(P)
BRST BRSTP	Resets the 'n'th bit in the specified word device to 0. 	Page 342 BRST(P)

■Performing a bit test

Instruction symbol	Processing details	Reference
TEST TESTP	Extracts the 'n'th bit in the specified word device. 	Page 344 TEST(P)
DTEST DTESTP	Extracts the 'n'th bit in the specified double-word device. 	Page 346 DTEST(P)

■Batch-resetting bit devices

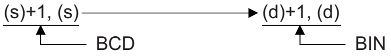
Instruction symbol	Processing details	Reference
BKRST BKRSTP	Resets the (n) points of bit devices starting from the bit device specified. 	Page 348 BKRST(P)

Data conversion instructions

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

Instruction symbol	Processing details	Reference
BCD BCDP	Converts the specified 16-bit binary data (0 to 9999) to BCD 4-digit data. 	Page 378 BCD(P)
DBCD DBCDP	Converts the specified 32-bit binary data (0 to 99999999) to BCD 8-digit data. 	Page 380 DBCD(P)

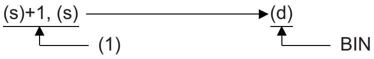
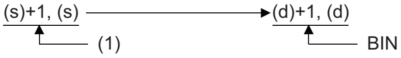
■Converting BCD 4-digit/8-digit data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
BIN BINP	Converts the specified BCD 4-digit data (0 to 9999) to 16-bit binary data. 	Page 382 BIN(P)
DBIN DBINP	Converts the specified BCD 8-digit data (0 to 99999999) to 32-bit binary data. 	Page 384 DBIN(P)

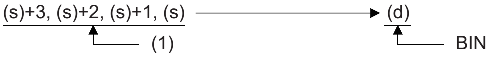
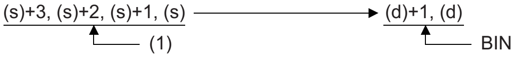
■Converting single-precision real number to 16-bit/32-bit signed binary data

Instruction symbol	Processing details	Reference
FLT2INT FLT2INTP	Converts the specified single-precision real number (-32768 to 32767) to 16-bit signed binary data.  (1) Real number	Page 386 FLT2INT(P)
FLT2DINT FLT2DINTP	Converts the specified single-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data.  (1) Real number	Page 390 FLT2DINT(P)

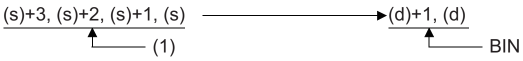
■Converting single-precision real number to 16-bit/32-bit unsigned binary data

Instruction symbol	Processing details	Reference
FLT2UINT FLT2UINTP	Converts the specified single-precision real number (0 to 65535) to 16-bit unsigned binary data.  (1) Real number	Page 388 FLT2UINT(P)
FLT2UDINT FLT2UDINTP	Converts the specified single-precision real number (0 to 4294967295) to 32-bit unsigned binary data.  (1) Real number	Page 392 FLT2UDINT(P)

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

Instruction symbol	Processing details	Reference
DBL2INT DBL2INTP	Converts the specified double-precision real number (-32768 to 32767) to 16-bit signed binary data.  (1) Real number	Page 394 DBL2INT(P)
DBL2DINT DBL2DINTP	Converts the specified double-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data.  (1) Real number	Page 398 DBL2DINT(P)

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

Instruction symbol	Processing details	Reference
DBL2UINT	Converts the specified double-precision real number (0 to 65535) to 16-bit unsigned binary data.  (1) Real number	Page 396 DBL2UINT(P)
DBL2UINTP		
DBL2UDINT	Converts the specified double-precision real number (0 to 4294967295) to 32-bit unsigned binary data.  (1) Real number	Page 400 DBL2UDINT(P)
DBL2UDINTP		

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

Instruction symbol	Processing details	Reference
INT2UINT	Converts the 16-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 402 INT2UINT(P)
INT2UINTP		
INT2UDINT	Converts the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 406 INT2UDINT(P)
INT2UDINTP		

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

Instruction symbol	Processing details	Reference
INT2DINT	Converts the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and stores the converted data in the device specified by (d).	Page 404 INT2DINT(P)
INT2DINTP		

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

Instruction symbol	Processing details	Reference
UINT2INT	Converts the 16-bit unsigned binary data in the device specified by (s) to 16-bit singed binary data, and stores the converted data in the device specified by (d).	Page 408 UINT2INT(P)
UINT2INTP		
UINT2DINT	Converts the 16-bit unsigned binary data in the device specified by (s) to 32-bit singed binary data, and stores the converted data in the device specified by (d).	Page 410 UINT2DINT(P)
UINT2DINTP		

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

Instruction symbol	Processing details	Reference
UINT2UDINT	Converts the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 412 UINT2UDINT(P)
UINT2UDINTP		

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

Instruction symbol	Processing details	Reference
DINT2INT	Converts the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and stores the converted data in the device specified by (d).	Page 414 DINT2INT(P)
DINT2INTP		

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

Instruction symbol	Processing details	Reference
DINT2UINT	Converts the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 416 DINT2UINT(P)
DINT2UINTP		
DINT2UDINT	Converts the 32-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 418 DINT2UDINT(P)
DINT2UDINTP		

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

Instruction symbol	Processing details	Reference
UDINT2INT	Converts the 32-bit unsigned binary data in the device specified by (s) to 16-bit singed binary data, and stores the converted data in the device specified by (d).	Page 420 UDINT2INT(P)
UDINT2INTP		
UDINT2DINT	Converts the 32-bit unsigned binary data in the device specified by (s) to 32-bit singed binary data, and stores the converted data in the device specified by (d).	Page 424 UDINT2DINT(P)
UDINT2DINTP		

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

Instruction symbol	Processing details	Reference
UDINT2UINT UDINT2UINTP	Converts the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 422 UDINT2UINT(P)

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

Instruction symbol	Processing details	Reference
GRY GRYP	Converts the specified 16-bit binary data (-32768 to 32767) to 16-bit binary Gray code data. $\underset{\text{BIN}}{\underline{(s)}} \longrightarrow \underset{\text{Gc}}{\underline{(d)}}$ Gc: Gray code	Page 426 GRY(P)(_U)
GRY_U GRYP_U	Converts the specified 16-bit binary data (0 to 65535) to 16-bit binary Gray code data. $\underset{\text{BIN}}{\underline{(s)}} \longrightarrow \underset{\text{Gc}}{\underline{(d)}}$ Gc: Gray code	
DGRY DGRYP	Converts the specified 32-bit binary data (-2147483648 to 2147483647) to 32-bit binary Gray code data. $\underset{\text{BIN}}{\underline{(s)+1, (s)}} \longrightarrow \underset{\text{Gc}}{\underline{(d)+1, (d)}}$ Gc: Gray code	Page 428 DGRY(P)(_U)
DGRY_U DGRYP_U	Converts the specified 32-bit binary data (0 to 4294967295) to 32-bit binary Gray code data. $\underset{\text{BIN}}{\underline{(s)+1, (s)}} \longrightarrow \underset{\text{Gc}}{\underline{(d)+1, (d)}}$ Gc: Gray code	

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

Instruction symbol	Processing details	Reference
GBIN GBINP	Converts the specified 16-bit binary Gray code data (-32768 to 32767) to 16-bit binary data. $\underset{\text{Gc}}{\underline{(s)}} \longrightarrow \underset{\text{BIN}}{\underline{(d)}}$ Gc: Gray code	Page 430 GBIN(P)(_U)
GBIN_U GBINP_U	Converts the specified 16-bit binary Gray code data (0 to 65535) to 16-bit binary data. $\underset{\text{Gc}}{\underline{(s)}} \longrightarrow \underset{\text{BIN}}{\underline{(d)}}$ Gc: Gray code	
DGBIN DGBINP	Converts the specified 32-bit binary Gray code data (-2147483648 to 2147483647) to 32-bit binary data. $\underset{\text{Gc}}{\underline{(s)+1, (s)}} \longrightarrow \underset{\text{BIN}}{\underline{(d)+1, (d)}}$ Gc: Gray code	Page 432 DGBIN(P)(_U)
DGBIN_U DGBINP_U	Converts the specified 32-bit binary Gray code data (0 to 4294967295) to 32-bit binary data. $\underset{\text{Gc}}{\underline{(s)+1, (s)}} \longrightarrow \underset{\text{BIN}}{\underline{(d)+1, (d)}}$ Gc: Gray code	

■Converting 16-bit binary data block to BCD 4-digit data block

Instruction symbol	Processing details	Reference
BKBCD BKBCDP	Batch-converts the (n) points of binary data in the device starting from the one specified by (s) to BCD data, and stores the converted data in the device specified by (d) and later.	Page 434 BKBCD(P)

■Converting BCD 4-digit data block to 16-bit binary data block

Instruction symbol	Processing details	Reference
BKBIN BKBINP	Batch-converts the (n) points of BCD data in the device starting from the one specified by (s) to binary data, and stores the converted data in the device specified by (d) and later.	Page 436 BKBIN(P)

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

Instruction symbol	Processing details	Reference
DABIN DABINP DABIN_U DABINP_U	Converts the decimal ASCII value in the device specified by (s) to 1-word binary data, and stores the converted data in the word device number specified by (d).	Page 438 DABIN(P)(_U)
DDABIN DDABINP DDABIN_U DDABINP_U	Converts the decimal ASCII value in the device specified by (s) to 2-word binary data, and stores the converted data in the word device number specified by (d).	Page 441 DDABIN(P)(_U)

■Converting hexadecimal ASCII data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
HABIN HABINP	Converts the hexadecimal ASCII value in the device specified by (s) to 1-word binary data, and stores the converted data in the word device number specified by (d).	Page 445 HABIN(P)
DHABIN DHABINP	Converts the hexadecimal ASCII value in the device specified by (s) to 2-word binary data, and stores the converted data in the word device number specified by (d).	Page 448 DHABIN(P)

■Converting decimal ASCII data to BCD 4-digit/8-digit data

Instruction symbol	Processing details	Reference
DABCD DABCDP	Converts the decimal ASCII value in the device specified by (s) to 1-word BCD data, and stores the converted data in the word device number specified by (d).	Page 451 DABCD(P)
DDABCD DDABCDP	Converts the decimal ASCII value in the device specified by (s) to 2-word BCD data, and stores the converted data in the word device number specified by (d).	Page 454 DDABCD(P)

■Converting decimal string data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
VAL VALP VAL_U VALP_U	Converts a character string including the decimal point in the device specified by (s) to 1-word binary data and the number of decimal positions, and stores the converted data in the device areas specified by (d1) and (d2).	Page 457 VAL(P)(_U)
DVAL DVALP DVAL_U DVALP_U	Converts a character string including the decimal point in the device specified by (s) to 2-word binary data and the number of decimal positions, and stores the converted data in the device areas specified by (d1) and (d2).	Page 460 DVAL(P)(_U)

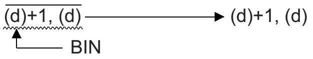
■Converting hexadecimal ASCII data to hexadecimal binary data

Instruction symbol	Processing details	Reference
ASC2INT ASC2INTP	Converts the hexadecimal ASCII data in the word device specified by (s) and later to binary data by the number of characters specified by (n), and stores the converted data in the device number specified by (d) and later.	Page 463 ASC2INT(P)

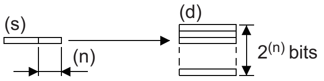
■Converting single-precision real number to BCD format data

Instruction symbol	Processing details	Reference
EMOD EMODP	Converts the 32-bit floating-point data in the device specified by (s1) to BCD of the number of decimal positions specified by (s2), and stores the converted data in the device specified by (d).	Page 465 EMOD(P)

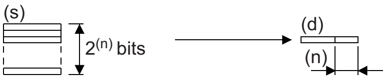
■Two's complement of 16-bit/32-bit binary data (sign inversion)

Instruction symbol	Processing details	Reference
NEG	Inverts the sign of 16-bit binary device. 	Page 467 NEG(P)
NEGP		
DNEG	Inverts the sign of 32-bit binary device. 	Page 469 DNEG(P)
DNEGP		

■Decoding 8-bit data to 256-bit data

Instruction symbol	Processing details	Reference
DECO	Decodes the lower (n) bits of the specified device. 	Page 471 DECO(P)
DECOP		

■Encoding 256-bit data to 8-bit data

Instruction symbol	Processing details	Reference
ENCO	Encodes the bit data of 'n'th power of 2. 	Page 473 ENCO(P)
ENCOP		

■Decoding data to seven-segment display data

Instruction symbol	Processing details	Reference
SEG	Decodes the data consisting of 0 to F specified by the lower 4 bits of the device to seven-segment display data. 	Page 475 SEG(P)
SEGP		

■Separating data in units of 4 bits

Instruction symbol	Processing details	Reference
DIS	Separates the 16-bit in the device specified by (s) in units of 4 bits, and stores the separated data in the (n) points of 4 low-order bits in the device starting from the one specified by (d). (n<4)	Page 478 DIS(P)
DISP		

■Combining data in units of 4 bits

Instruction symbol	Processing details	Reference
UNI	Adds the (n) points of 4 low-order bit data in the device starting from the one specified by (s), and stores the connected data in the device specified by (d). (n<4)	Page 480 UNI(P)
UNIP		

■Separating/combining data in units of bits

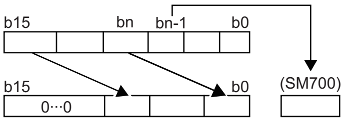
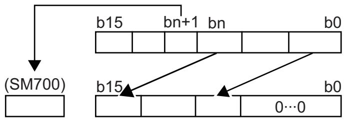
Instruction symbol	Processing details	Reference
NDIS	Separates the data in the device, specified by (s1) and later, to the bits in the device specified by (s2) and later, and stores the separated data in order in the device starting from the one specified by (d).	Page 482 NDIS(P)
NDISP		
NUNI	Connects the data in the device specified by (s1) and later, in units of bits in the device specified by (s2) and later, and stores the connected data in order in the device starting from the one specified by (d).	Page 484 NUNI(P)
NUNIP		

■Separating/combining data in units of bytes

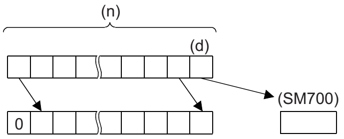
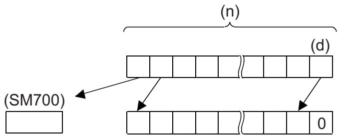
Instruction symbol	Processing details	Reference
WTOB	Converts the (n) points of 16 bit data in the device specified by (s) in units of 8 bits, and stores the converted data in order in the device starting from the one specified by (d).	Page 486 WTOB(P)
WTOBP		
BTOW	Connects 8 low-order bits of the (n) points of 16 bit data in the device specified by (s) to 16 bits, and stores the connected data in order in the device starting from the one specified by (d).	Page 488 BTOW(P)
BTOWP		

Shift instructions

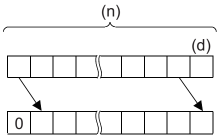
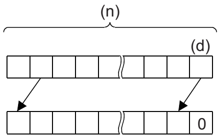
■Shifting 16-bit binary data to the right/left by n bit(s)

Instruction symbol	Processing details	Reference
SFR SFRP	Shifts the 16-bit binary data in the specified device to the right. 	Page 350 SFR(P)
SFL SFLP	Shifts the 16-bit binary data in the specified device to the left. 	Page 352 SFL(P)

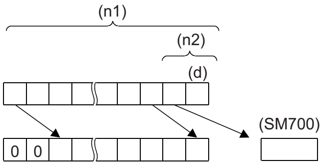
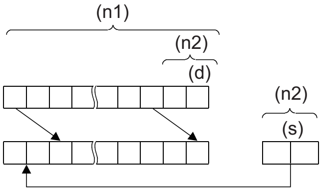
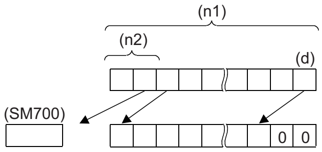
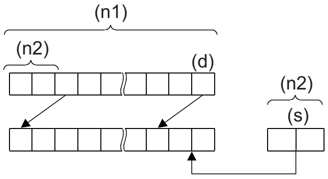
■Shifting n-bit data to the right/left by one bit

Instruction symbol	Processing details	Reference
BSFR BSFRP	Shifts the n points of data starting from the specified device to the right by one bit. 	Page 354 BSFR(P)
BSFL BSFLP	Shifts the n points of data starting from the specified device to the left by one bit. 	Page 356 BSFL(P)

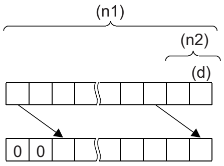
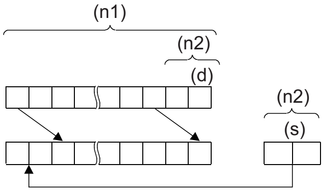
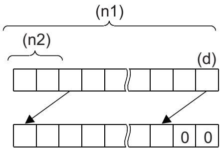
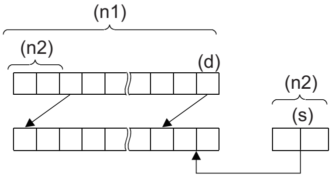
■Shifting n-word data to the right/left by one word

Instruction symbol	Processing details	Reference
DSFR DSFRP	Shifts the n points of data starting from the specified device to the right by one word. 	Page 358 DSFR(P)
DSFL DSFLP	Shifts the n points of data starting from the specified device to the left by one word. 	Page 360 DSFL(P)

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

Instruction symbol	Processing details	Reference
SFTBR	Shifts the n-bit data starting from the specified device to the right by n bit(s).  	Page 362 SFTBR(P)
SFTBRP		
SFTR	Shifts the (n2) bit(s) of area to the right within the (n1) bits of data area starting from the specified device.  	Page 364 SFTR(P)
SFTRP		
SFTBL	Shifts the n-bit data starting from the specified device to the left by n bit(s).  	Page 366 SFTBL(P)
SFTBLP		
SFTL	Shift the (n2) bit(s) of area to the left within the (n1) bits of data area starting from the specified device.  	Page 368 SFTL(P)
SFTLP		

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

Instruction symbol	Processing details	Reference
SFTWR	Shifts the n-word data starting from the specified device to the right by n word(s). 	Page 370 SFTWR(P)
SFTWRP		
WSFR	Shifts the (n2) word(s) of area to the right within the (n1) words of data area starting from the specified device. 	Page 372 WSFR(P)
WSFRP		
SFTWL	Shifts the n-word data starting from the specified device to the left by n word(s). 	Page 374 SFTWL(P)
SFTWLP		
WSFL	Shifts the (n2) word(s) of area to the left within the (n1) words of data area starting from the specified device. 	Page 376 WSFL(P)
WSFLP		

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) → (d)	Page 490 MOV(P)
MOVP		
DMOV	Transfers the 32-bit binary data in the device specified. (s)+1, (s) → (d)+1, (d)	Page 492 DMOV(P)
DMOVP		

■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. $\overline{(s)}$ → (d)	Page 494 CML(P)
CMLP		
DCML	Inverts the specified 32-bit binary data bit by bit, and transfers the inverted data. $\overline{(s)+1, (s)}$ → (d)+1, (d)	Page 496 DCML(P)
DCMLP		

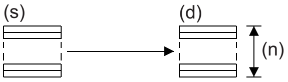
■Shifting data in units of 4 bits

Instruction symbol	Processing details	Reference
SMOV	Distributes and combines data in units of 4 bits.	Page 498 SMOV(P)
SMOVP		

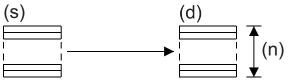
■Inverting and transferring 1-bit data

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 501 CMLB(P)
CMLBP		

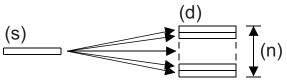
■Transferring 16-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
BMOV	Batch-transfers the (n) points of 16-bit binary data starting from the device specified.  (n) = 1 to 65535	Page 503 BMOV(P)
BMOVP		

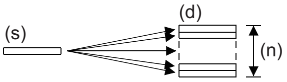
■Transferring 16-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
BMOVL	Batch-transfers the (n) points of 16-bit binary data starting from the device specified.  (n) = 1 to 4294967295	Page 505 BMOVL(P)
BMOVLP		

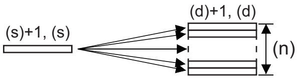
■Transferring the same 16-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
FMOV	Transfers 16-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 65535	Page 507 FMOV(P)
FMOVP		

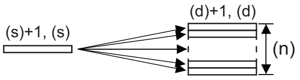
■Transferring the same 16-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
FMOVL	Transfers 16-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 4294967295	Page 509 FMOVL(P)
FMOVLP		

■Transferring the same 32-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
DFMOV	Transfers 32-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 65535	Page 511 DFMOV(P)
DFMOVP		

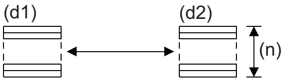
■Transferring the same 32-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
DFMOVL	Transfers 32-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 4294967295	Page 513 DFMOVL(P)
DFMOVLP		

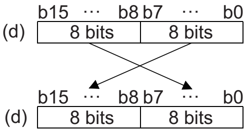
■Exchanging 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
XCH	Exchanges the 16-bit binary data specified. (d1) ←→ (d2)	Page 515 XCH(P)
XCHP		
DXCH	Exchanges the 32-bit binary data specified. (d1)+1, (d1) ←→ (d2)+1, (d2)	Page 517 DXCH(P)
DXCHP		

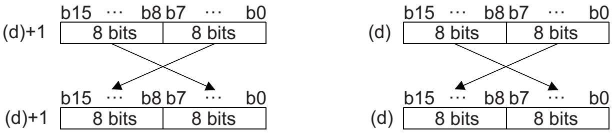
■Exchanging 16-bit binary block data

Instruction symbol	Processing details	Reference
BXCH	Exchanges the (n) points of 16-bit binary data starting from the devices specified. 	Page 519 BXCH(P)
BXCHP		

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

Instruction symbol	Processing details	Reference
SWAP	Exchanges upper and lower 8-bit data in the specified device. 	Page 521 SWAP(P)
SWAPP		

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

Instruction symbol	Processing details	Reference
DSWAP	Exchanges upper and lower 8-bit data in the specified device. 	Page 522 DSWAP(P)
DSWAPP		

■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 523 MOVB(P)
MOVBP		

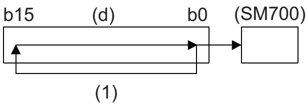
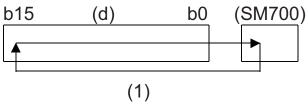
■Transferring n-bit data

Instruction symbol	Processing details	Reference
BLKMOVB	Batch-transfers the (n) points of bit data in the device starting from the one specified by (s) to the (n) points of bit data in the device starting from the one specified by (d).	Page 525 BLKMOVB(P)
BLKMOVBP		

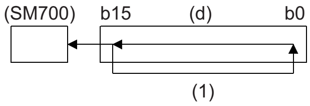
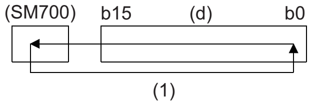
2.3 Application Instructions

Rotation instructions

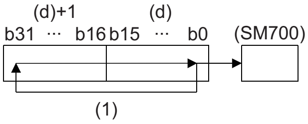
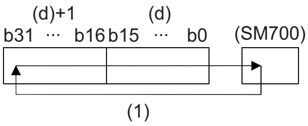
■Rotating 16-bit binary data to the right

Instruction symbol	Processing details	Reference
ROR RORP	Rotates the 16-bit binary data to the right by (n) bit(s) (not including the carry flag).  (1) Right rotation by (n) bits	Page 527 ROR(P), RCR(P)
RCR RCRP	Rotates the 16-bit binary data to the right by (n) bit(s) (including the carry flag).  (1) Right rotation by (n) bits	

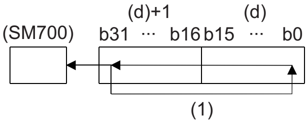
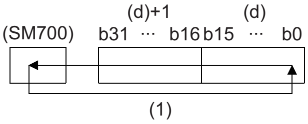
■Rotating 16-bit binary data to the left

Instruction symbol	Processing details	Reference
ROL ROLP	Rotates the 16-bit binary data to the left by (n) bit(s) (not including the carry flag).  (1) Left rotation by (n) bits	Page 530 ROL(P), RCL(P)
RCL RCLP	Rotates the 16-bit binary data to the left by (n) bit(s) (including the carry flag).  (1) Left rotation by (n) bits	

■Rotating 32-bit binary data to the right

Instruction symbol	Processing details	Reference
DROR DRORP	Rotates the 32-bit binary data to the right by (n) bit(s) (not including the carry flag).  (1) Right rotation by (n) bits	Page 533 DROR(P), DRCR(P)
DRCR DRCRP	Rotates the 32-bit binary data to the right by (n) bit(s) (including the carry flag).  (1) Right rotation by (n) bits	

■Rotating 32-bit binary data to the left

Instruction symbol	Processing details	Reference
DROL	Rotates the 32-bit binary data to the left by (n) bit(s) (not including the carry flag).  (1) Left rotation by (n) bits	Page 535 DROL(P), DRCL(P)
DROLP		
DRCL	Rotates the 32-bit binary data to the left by (n) bit(s) (including the carry flag).  (1) Left rotation by (n) bits	
DRCLP		

Program branch instructions

■Pointer branch

Instruction symbol	Processing details	Reference
CJ	Executes the program specified by the pointer number within the same program file.	Page 537 CJ, SCJ, JMP
SCJ	Executes the program specified by the pointer number within the same program file starting with the next scan.	
JMP	Unconditionally executes the program specified by the pointer number within the same program file.	

■Jumping to END

Instruction symbol	Processing details	Reference
GOEND	Invokes a jump to the FEND or END instruction within the same program file.	Page 540 GOEND

Program execution control instructions

■Disabling/enabling interrupt programs

Instruction symbol	Processing details	Reference
DI	Disables the execution of interrupt programs.	Page 541 DI, EI
EI	Clears the interrupt disabled state.	

■Disabling the interrupt program with specified priority or lower

Instruction symbol	Processing details	Reference
DI	Disables the execution of the interrupt program with a priority specified by (s) or lower until the EI instruction is executed.	Page 544 DI

■Interrupt program mask

Instruction symbol	Processing details	Reference
IMASK	Enables or disables the execution of the interrupt program with the specified interrupt pointer number.	Page 549 IMASK

■Disabling/enabling the specified interrupt pointer

Instruction symbol	Processing details	Reference
SIMASK	Enables or disables the execution of the interrupt program with the specified interrupt pointer number.	Page 551 SIMASK

■Returning from the interrupt program

Instruction symbol	Processing details	Reference
IRET	Indicates the end of the processing of an interrupt program.	Page 553 IRET

■Resetting the watchdog timer

Instruction symbol	Processing details	Reference
WDT	Resets the watchdog timer.	Page 554 WDT(P)
WDTP		

Structure creation instructions

■Performing the FOR to NEXT instruction loop

Instruction symbol	Processing details	Reference
FOR	Executes the processing between FOR to NEXT (n) times.	Page 555 FOR, NEXT
NEXT		

■Forcibly terminating the FOR to NEXT instruction loop

Instruction symbol	Processing details	Reference
BREAK	Forcibly terminates the loop processing between the FOR and NEXT instructions, and passes the control to the specified pointer.	Page 557 BREAK(P)
BREAKP		

■Calling a subroutine program

Instruction symbol	Processing details	Reference
CALL	Executes a subroutine program specified by (P) when the input condition is met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 559 CALL(P)
CALLP		

■Returning from the subroutine program called

Instruction symbol	Processing details	Reference
RET	Indicates the end of a subroutine program.	Page 563 RET

■Calling a subroutine program and turning the output off

Instruction symbol	Processing details	Reference
FCALL	Performs non-execution processing of the subroutine program specified by (P) when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 564 FCALL(P)
FCALLP		

■Calling a subroutine program in the specified program file

Instruction symbol	Processing details	Reference
ECALL	Executes the subroutine program specified by (P) of the specified program when the input conditions are met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 568 ECALL(P)
ECALLP		

■Calling a subroutine program in the specified program file and turning the output off

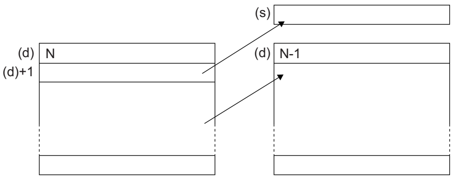
Instruction symbol	Processing details	Reference
EFCALL	Performs non-execution processing of the subroutine program specified by (P) of the specified program when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 573 EFCALL(P)
EFCALLP		

■Calling a subroutine program

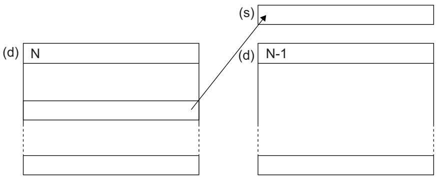
Instruction symbol	Processing details	Reference
XCALL	Executes a subroutine program specified by (P) when the input condition is met. Perform non-execution processing of the subroutine program specified by (P) when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 578 XCALL

Data table operation instructions

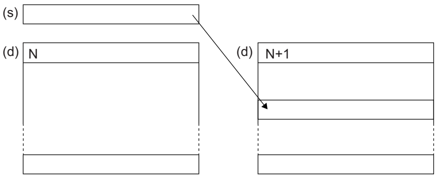
■Reading the oldest data from the data table

Instruction symbol	Processing details	Reference
FIFR FIFRP	Stores the data first stored in the table in the specified device.  N: Number of data	Page 583 FIFR(P)

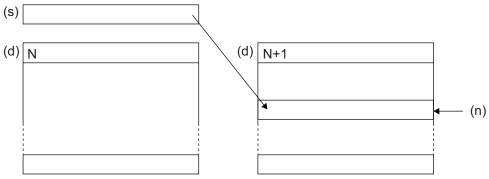
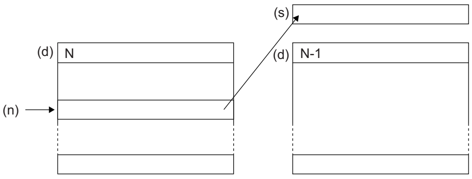
■Reading the newest data from the data table

Instruction symbol	Processing details	Reference
FPOP FPOPP	Stores the data last stored in the table in the specified device.  N: Number of data	Page 585 FPOP(P)

■Writing data to the data table

Instruction symbol	Processing details	Reference
FIFW FIFWP	Stores 16-bit binary data to the specified data table.  N: Number of data	Page 587 FIFW(P)

■Inserting/deleting data to/from the data table

Instruction symbol	Processing details	Reference
FINS FINSP	Inserts 16-bit binary data to the (n)th position in the specified data table.  N: Number of data	Page 589 FINS(P)
FDEL FDELP	Deletes the data at the (n)th position in the data table.  N: Number of data	Page 591 FDEL(P)

Reading/writing data instructions

■Reading data from the data memory

Instruction symbol	Processing details	Reference
S.DEVLD	Reads data from the device data storage file in data memory.	Page 594 S(P).DEVLD
SP.DEVLD		

■Writing data to the data memory

Instruction symbol	Processing details	Reference
SP.DEVST	Writes the specified number of points of data to the device data storage file in data memory.	Page 596 SP.DEVST

■Reading data from the specified file

Instruction symbol	Processing details	Reference
SP.FREAD	Reads data from the specified file.	Page 599 SP.FREAD

■Writing data to the specified file

Instruction symbol	Processing details	Reference
SP.FWRITE	Writes data to the specified file.	Page 616 SP.FWRITE

Debugging and failure diagnostic instructions

■Resetting the error display and the annunciator display

Instruction symbol	Processing details	Reference
LEDR	This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the CPU module.	Page 628 LEDR

■Generating a continuation error

Instruction symbol	Processing details	Reference
PALERT	Generates a continuation error in the CPU module.	Page 629 PALERT(P)
PALERTP		

■Generating a stop error

Instruction symbol	Processing details	Reference
PABORT	Stops program execution and generates a stop error in the CPU module.	Page 630 PABORT

String processing instructions

■Comparing string data

Instruction symbol	Processing details	Reference
LD$=, AND$=, OR$=	Compares the character string specified by (s1) with the character string specified by (s2) (character by character).	Page 631 LD$□, AND$□, OR$□
LD$<>, AND$<>, OR$<>		
LD$>, AND$>, OR$>		
LD$<=, AND$<=, OR$<=		
LD$<, AND$<, OR$<		
LD$>=, AND$>=, OR$>=		

■Concatenating string data

Instruction symbol	Processing details	Reference
$+	Connects the character strings in the device specified by (s) to those in the device specified by (d), and stores the connected data in the device specified by (d) and later.	Page 634 $+(P) [when two operands are set]
$+P		
$+	Connects the character strings in the device specified by (s2) to those in the device specified by (s1), and stores the connected data in the device specified by (d) and later.	Page 636 $+(P) [when three operands are set]
$+P		

■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 638 $MOV(P)
$MOVP		
$MOV_WS	Transfers the Unicode character strings in the device specified by (s) to the device specified by (d) and later.	Page 640 $MOV(P)_WS
$MOVP_WS		

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

Instruction symbol	Processing details	Reference
BINDA	Converts the 1-word binary data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 642 BINDA(P)(_U)
BINDAP		
BINDA_U		
BINDAP_U		
DBINDA	Converts the 2-word binary data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 646 DBINDA(P)(_U)
DBINDAP		
DBINDA_U		
DBINDAP_U		

■Converting 16-bit/32-bit binary data to hexadecimal ASCII

Instruction symbol	Processing details	Reference
BINHA	Converts the 1-word binary data in the device specified by (s) to hexadecimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 651 BINHA(P)
BINHAP		
DBINHA	Converts the 2-word binary data in the device specified by (s) to hexadecimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 655 DBINHA(P)
DBINHAP		

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

Instruction symbol	Processing details	Reference
STR	Converts the 1-word binary data in the device specified by (s2) to a decimal character string consisting of the total number digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the word device specified by (d).	Page 659 STR(P)(_U)
STRP		
STR_U		
STRP_U		
DSTR	Converts the 2-word binary data in the device specified by (s2) to a decimal character string consisting of the total number digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the word device specified by (d).	Page 662 DSTR(P)(_U)
DSTRP		
DSTR_U		
DSTRP_U		

■Converting BCD 4-digit/8-digit data to decimal ASCII code

Instruction symbol	Processing details	Reference
BCDDA	Converts the 1-word BCD data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 665 BCDDA(P)
BCDDAP		
DBCDDA	Converts the 2-word BCD data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 669 DBCDDA(P)
DBCDDAP		

■Converting single-precision real number to string data

Instruction symbol	Processing details	Reference
ESTR	Converts the single-precision real number in the device specified by (s1) to a character string, and stores the converted data in the word device specified by (d).	Page 674 ESTR(P)
ESTRP		

■Converting hexadecimal binary data to hexadecimal ASCII code

Instruction symbol	Processing details	Reference
INT2ASC	Converts the 1-word binary data in the device number specified by (s) and later to hexadecimal ASCII, and stores the converted data by the number of characters in the device specified by (n) in the word device number specified by (d) and later.	Page 679 INT2ASC(P)
INT2ASCP		

■Converting Unicode character string to Shift JIS character string

Instruction symbol	Processing details	Reference
WS2SJIS WS2SJISP	Converts the Unicode character string in the device specified by (s) to the shift JIS character string, and stores the converted data in the device specified by (d).	Page 681 WS2SJIS(P)

■Converting shift JIS character string to Unicode character string (without byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WS SJIS2WSP	Converts the shift JIS character string in the device specified by (s) to a Unicode character string, and stores the converted data in the device specified by (d).	Page 683 SJIS2WS(P)

■Converting shift JIS to Unicode (with byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WSB SJIS2WSBP	Converts the shift JIS character string in the device specified by (s) to the Unicode character string, add a byte order mark to the head of the converted data, and stores it in the device specified by (d).	Page 685 SJIS2WSB(P)

■Detecting a string length

Instruction symbol	Processing details	Reference
LEN LENP	Stores the length (the number of characters) of the character string data, which is stored in the device specified by (s), in the device specified by (d).	Page 687 LEN(P)

■Extracting string data from the right/left

Instruction symbol	Processing details	Reference
RIGHT RIGHTP	Stores the (n) characters from the last character of the character string, which is stored in the device specified by (s), in the device specified by (d).	Page 689 RIGHT(P)
LEFT LEFTP	Stores the (n) characters from the first character of the character string, which is stored in the device specified by (s), in the device specified by (d).	Page 691 LEFT(P)

■Extracting/replacing the specified string data

Instruction symbol	Processing details	Reference
MIDR MIDRP	Retrieves the character string in the device specified by (s1) by the number of specified characters from the location in the device specified by (s2), and stores the retrieved data in the device specified by (d).	Page 693 MIDR(P)
MIDW MIDWP	Retrieves the specified number of characters from the character string in the device specified by (s1), and stores the retrieved data at the location specified by (s2) in the character string stored in the device specified by (d).	Page 695 MIDW(P)

■Searching string data

Instruction symbol	Processing details	Reference
INSTR INSTRP	Searches the character string in the device specified by (s2), starting from the (s3)th character, for the character string in the device specified by (s1), and stores the matching location in the device specified by (d).	Page 698 INSTR(P)

■Inserting string data

Instruction symbol	Processing details	Reference
STRINS STRINSP	Inserts the character string data in the device specified by (s1) to the (s2)th character (insertion position) from the head of the character string data in the device specified by (d).	Page 700 STRINS(P)

■Deleting string data

Instruction symbol	Processing details	Reference
STRDEL STRDELP	Deletes the (n) characters starting from the position (deletion start position) specified by the (s)th character from the head of the character string data in the device specified by (d).	Page 702 STRDEL(P)

Real number instructions

■Comparing single-precision real numbers

Instruction symbol	Processing details	Reference
LDE=, ANDE=, ORE=	Performs a comparison operation of a single-precision real number. (Devices are used as a normally open contact.)	Page 704 LDE□, ANDE□, ORE□
LDE<>, ANDE<>, ORE<>		
LDE>, ANDE>, ORE>		
LDE<=, ANDE<=, ORE<=		
LDE<, ANDE<, ORE<		
LDE>=, ANDE>=, ORE>=		

■Comparing double-precision real numbers

Instruction symbol	Processing details	Reference
LDED=, ANDED=, ORED=	Performs a comparison operation of a double-precision real number. (Devices are used as a normally open contact.)	Page 706 LDED□, ANDED□, ORED□
LDED<>, ANDED<>, ORED<>		
LDED>, ANDED>, ORED>		
LDED<=, ANDED<=, ORED<=		
LDED<, ANDED<, ORED<		
LDED>=, ANDED>=, ORED>=		

■Outputting a comparison result of single-precision real numbers

Instruction symbol	Processing details	Reference
ECMP	Compares the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 709 ECMP(P)
ECMPP		

■Outputting a comparison result of double-precision real numbers

Instruction symbol	Processing details	Reference
EDCMP	Compares the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 711 EDCMP(P)
EDCMPP		

■Outputting a band comparison result of single-precision real number

Instruction symbol	Processing details	Reference
EZCP	Compares the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 713 EZCP(P)
EZCPP		

■Outputting a band comparison result of double-precision real number

Instruction symbol	Processing details	Reference
EDZCP	Compares the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 715 EDZCP(P)
EDZCPP		

■Adding/subtracting single-precision real numbers

Instruction symbol	Processing details	Reference
E+	Adds single-precision real numbers. (Using two operands)	Page 717 E+(P) [when two operands are set]
E+P		
E+	Adds single-precision real numbers. (Using three operands)	Page 719 E+(P) [when three operands are set]
E+P		
E-	Performs subtraction between single-precision real numbers. (Using two operands)	Page 721 E-(P) [when two operands are set]
E-P		
E-	Performs subtraction between single-precision real numbers. (Using three operands)	Page 723 E-(P) [when three operands are set]
E-P		

■Adding/subtracting double-precision real numbers

Instruction symbol	Processing details	Reference
ED+ ED+P	Adds double-precision real numbers. (Using two operands)	Page 725 ED+(P) [when two operands are set]
ED+ ED+P	Adds double-precision real numbers. (Using three operands)	Page 727 ED+(P) [when three operands are set]
ED- ED-P	Performs subtraction between double-precision real numbers. (Using two operands)	Page 729 ED-(P) [when two operands are set]
ED- ED-P	Performs subtraction between double-precision real numbers. (Using three operands)	Page 731 ED-(P) [when three operands are set]

■Multiplying/dividing single-precision real numbers

Instruction symbol	Processing details	Reference
E* E*P	Multiplies single-precision real numbers.	Page 733 E*(P)
E/ E/P	Performs division between single-precision real numbers.	Page 735 E/(P)

■Multiplying/dividing double-precision real numbers

Instruction symbol	Processing details	Reference
ED* ED*P	Multiplies double-precision real numbers.	Page 737 ED*(P)
ED/ ED/P	Performs division between double-precision real numbers.	Page 739 ED/(P)

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

Instruction symbol	Processing details	Reference
INT2FLT INT2FLTP	Converts the 16-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the converted data in the device specified by (d).	Page 741 INT2FLT(P)
DINT2FLT DINT2FLTP	Converts the 32-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 745 DINT2FLT(P)

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

Instruction symbol	Processing details	Reference
UINT2FLT UINT2FLTP	Converts the 16-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 743 UINT2FLT(P)
UDINT2FLT UDINT2FLTP	Converts the 32-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 747 UDINT2FLT(P)

■Converting double-precision real number to single-precision real number

Instruction symbol	Processing details	Reference
DBL2FLT DBL2FLTP	Converts the double-precision real number in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 749 DBL2FLT(P)

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

Instruction symbol	Processing details	Reference
INT2DBL	Converts the 16-bit signed binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 751 INT2DBL(P)
INT2DBLP		
DINT2DBL	Converts the 32-bit signed binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 755 DINT2DBL(P)
DINT2DBLP		

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

Instruction symbol	Processing details	Reference
UINT2DBL	Converts the 16-bit unsigned binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 753 UINT2DBL(P)
UINT2DBLP		
UDINT2DBL	Converts the 32-bit unsigned binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 757 UDINT2DBL(P)
UDINT2DBLP		

■Converting single-precision real number to double-precision real number

Instruction symbol	Processing details	Reference
FLT2DBL	Converts the single-precision real number in the device specified by (s) to a double-precision real number, and stores the double-precision real number in the device specified by (d).	Page 759 FLT2DBL(P)
FLT2DBLP		

■Converting string data to single-precision real number

Instruction symbol	Processing details	Reference
EVAL	Converts the character string in the device specified by (s) to a single-precision real number, and stores the converted data in the device specified by (d).	Page 761 EVAL(P)
EVALP		

■Converting BCD format data to single-precision real number

Instruction symbol	Processing details	Reference
EREXP	Converts the BCD data in the device specified by (s1) to a single-precision real number with the number of decimal positions specified by (s2), and stores the converted data in the device specified by (d).	Page 765 EREXP(P)
EREXPP		

■Inverting the sign of single-precision real number

Instruction symbol	Processing details	Reference
ENEG	Inverts the sign of single-precision real number data. $\underbrace{\overline{(d)+1, (d)}}_{(1)} \longrightarrow (d)+1, (d)$ (1) Real number	Page 767 ENEG(P)
ENEGP		

■Inverting the sign of double-precision real number

Instruction symbol	Processing details	Reference
EDNEG	Inverts the sign of double-precision real number data. $\underbrace{\overline{(d)+3, (d)+2, (d)+1, (d)}}_{(1)} \longrightarrow (d)+3, (d)+2, (d)+1, (d)$ (1) Real number	Page 768 EDNEG(P)
EDNEGP		

■Transferring single-precision real number

Instruction symbol	Processing details	Reference
EMOV	Transfers single-precision real number data to the specified device. $\underbrace{(s)+1, (s)}_{(1)} \longrightarrow (d)+1, (d)$ (1) Real number	Page 769 EMOV(P)
EMOVP		

■Transferring double-precision real number

Instruction symbol	Processing details	Reference
EDMOV	Transfers double-precision real number data to the specified device. (s)+3, (s)+2, (s)+1, (s) → (d)+3, (d)+2, (d)+1, (d) (1) (1) Real number	Page 770 EDMOV(P)
EDMOVP		

■Calculating the sine of single-precision real number

Instruction symbol	Processing details	Reference
SIN	Calculates the sine of the angle specified by a single-precision real number.	Page 771 SIN(P)
SINP		

■Calculating the cosine of single-precision real number

Instruction symbol	Processing details	Reference
COS	Calculates the cosine of the angle specified by a single-precision real number.	Page 773 COS(P)
COSP		

■Calculating the tangent of single-precision real number

Instruction symbol	Processing details	Reference
TAN	Calculates the tangent of the angle specified by a single-precision real number.	Page 775 TAN(P)
TANP		

■Calculating the arc sine of single-precision real number

Instruction symbol	Processing details	Reference
ASIN	Calculates the angle from the sine specified by a single-precision real number.	Page 777 ASIN(P)
ASINP		

■Calculating the arc cosine of single-precision real number

Instruction symbol	Processing details	Reference
ACOS	Calculates the angle from the cosine specified by a single-precision real number.	Page 779 ACOS(P)
ACOSP		

■Calculating the arc tangent of single-precision real number

Instruction symbol	Processing details	Reference
ATAN	Calculates the angle from the tangent specified by a single-precision real number.	Page 781 ATAN(P)
ATANP		

■Calculating the sine of double-precision real number

Instruction symbol	Processing details	Reference
SIND	Calculates the sine of the angle specified by a double-precision real number.	Page 783 SIND(P)
SINDP		

■Calculating the cosine of double-precision real number

Instruction symbol	Processing details	Reference
COSD	Calculates the cosine of the angle specified by a double-precision real number.	Page 785 COSD(P)
COSDP		

■Calculating the tangent of double-precision real number

Instruction symbol	Processing details	Reference
TAND	Calculates the tangent of the angle specified by a double-precision real number.	Page 787 TAND(P)
TANDP		

■Calculating the arc sine of double-precision real number

Instruction symbol	Processing details	Reference
ASIND	Calculates the angle from the sine specified by a double-precision real number.	Page 789 ASIND(P)
ASINDP		

■Calculating the arc cosine of double-precision real number

Instruction symbol	Processing details	Reference
ACOSD	Calculates the angle from the cosine specified by a double-precision real number.	Page 791 ACOSD(P)
ACOSDP		

■Calculating the arc tangent of double-precision real number

Instruction symbol	Processing details	Reference
ATAND	Calculates the angle from the tangent specified by a double-precision real number.	Page 793 ATAND(P)
ATANDP		

■Calculating the sine of BCD data

Instruction symbol	Processing details	Reference
BSIN	Calculates the sine of the angle specified by a BCD value. SIN (s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 795 BSIN(P)
BSINP		

■Calculating the cosine of BCD data

Instruction symbol	Processing details	Reference
BCOS	Calculates the cosine of the angle specified by a BCD value. COS (s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 797 BCOS(P)
BCOSP		

■Calculating the tangent of BCD data

Instruction symbol	Processing details	Reference
BTAN	Calculates the tangent of the angle specified by a BCD value. TAN(s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 799 BTAN(P)
BTANP		

■Calculating the arc sine of BCD data

Instruction symbol	Processing details	Reference
BASIN	Calculates the arc sine of the angle specified by a BCD value. SIN^{-1} (s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 801 BASIN(P)
BASINP		

■Calculating the arc cosine of BCD data

Instruction symbol	Processing details	Reference
BACOS	Calculates the arc cosine of the angle specified by a BCD value. COS^{-1} (s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 803 BACOS(P)
BACOSP		

■Calculating the arc tangent of BCD data

Instruction symbol	Processing details	Reference
BATAN	Calculates the arc tangent of the angle specified by a BCD value. TAN^{-1} (s) → (d) (d)+1 (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 805 BATAN(P)
BATANP		

■Converting single-precision real number angle to radian

Instruction symbol	Processing details	Reference
RAD	Converts the unit of the measure of angle from the degree specified by a single-precision real number to radian. Converting degree to radian (s)+1, (s) → (d)+1, (d)	Page 807 RAD(P)
RADP		

■Converting single-precision real number radian to angle

Instruction symbol	Processing details	Reference
DEG	Converts the unit of the measure of angle from the radian specified by a single-precision real number to the degree. Converting radian to degree (s)+1, (s) → (d)+1, (d)	Page 809 DEG(P)
DEGP		

■Converting double-precision real number angle to radian

Instruction symbol	Processing details	Reference
RADD	Converts the unit of the measure of angle from the degree specified by a single-precision real number to radian. Converting degree to radian (s)+3, (s)+2, (s)+1, (s) → (d)+3, (d)+2, (d)+1, (d)	Page 811 RADD(P)
RADDP		

■Converting double-precision real number radian to angle

Instruction symbol	Processing details	Reference
DEGD	Converts the unit of the measure of angle from the radian specified by a double-precision real number to the degree. Converting radian to degree (s)+3, (s)+2, (s)+1, (s) → (d)+3, (d)+2, (d)+1, (d)	Page 813 DEGD(P)
DEGDP		

■Calculating the square root of single-precision real number

Instruction symbol	Processing details	Reference
ESQRT	Calculates the square root of the value specified by a single-precision real number. $\sqrt{(s)+1, (s)} \longrightarrow (d)+1, (d)$	Page 815 ESQRT(P)
ESQRTP		

■Calculating the square root of double-precision real number

Instruction symbol	Processing details	Reference
EDSQRT	Calculates the square root of the value specified by a double-precision real number. $\sqrt{(s)+3, (s)+2, (s)+1, (s)} \longrightarrow (d)+3, (d)+2, (d)+1, (d)$	Page 817 EDSQRT(P)
EDSQRTP		

■Calculating the exponent of single-precision real number

Instruction symbol	Processing details	Reference
EXP	Calculates the exponent of the value specified by a single-precision real number.	Page 819 EXP(P)
EXPP		

■Calculating the exponent of double-precision real number

Instruction symbol	Processing details	Reference
EXPD	Calculates the exponent of the value specified by a double-precision real number.	Page 821 EXPD(P)
EXPDP		

■Calculating the natural logarithm of single-precision real number

Instruction symbol	Processing details	Reference
LOG	Calculates the logarithm using the natural logarithm (e) of the value specified by a single-precision real number as the base.	Page 823 LOG(P)
LOGP		

■Calculating the natural logarithm of double-precision real number

Instruction symbol	Processing details	Reference
LOGD	Calculates the logarithm using the natural logarithm (e) of the value specified by a double-precision real number as the base.	Page 825 LOGD(P)
LOGDP		

■Calculating the square root of BCD 4-digit/8-digit data

Instruction symbol	Processing details	Reference
BSQRT	Calculates the square root of the value specified by a BCD 4-digit data. $\sqrt{(s)} \longrightarrow \begin{array}{\|c\|}\hline (d) \\ \hline (d)+1 \\ \hline\end{array}$ (d): Integral part (d)+1: Decimal part	Page 827 BSQRT(P)
BSQRTP		
BDSQRT	Calculates the square root of the value specified by a BCD 8-digit data. $\sqrt{(s)+1, (s)} \longrightarrow \begin{array}{\|c\|}\hline (d) \\ \hline (d)+1 \\ \hline\end{array}$ (d): Integral part (d)+1: Decimal part	Page 829 BDSQRT(P)
BDSQRTP		

■Calculating the exponentiation of single-precision real number

Instruction symbol	Processing details	Reference
POW	Calculates the exponentiation of a single-precision real number.	Page 831 POW(P)
POWP		

■Calculating the exponentiation of double-precision real number

Instruction symbol	Processing details	Reference
POWD	Calculates the exponentiation of a double-precision real number.	Page 833 POWD(P)
POWDP		

■Calculating the common logarithm of single-precision real number

Instruction symbol	Processing details	Reference
LOG10 LOG10P	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a single-precision real number.	Page 835 LOG10(P)

■Calculating the common logarithm of double-precision real number

Instruction symbol	Processing details	Reference
LOG10D LOG10DP	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a double-precision real number.	Page 837 LOG10D(P)

■Searching the maximum value of single-precision real number

Instruction symbol	Processing details	Reference
EMAX EMAXP	Searches for the maximum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (maximum value) in the device specified by (d).	Page 839 EMAX(P)

■Searching the maximum value of double-precision real number

Instruction symbol	Processing details	Reference
EDMAX EDMAXP	Searches for the maximum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (maximum value) in the device specified by (d).	Page 841 EDMAX(P)

■Searching the minimum value of single-precision real number

Instruction symbol	Processing details	Reference
EMIN EMINP	Searches for the minimum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (minimum value) in the device specified by (d).	Page 843 EMIN(P)

■Searching the minimum value of double-precision real number

Instruction symbol	Processing details	Reference
EDMIN EDMINP	Searches for the minimum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (minimum value) in the device specified by (d).	Page 845 EDMIN(P)

Random number instructions

■Generating random number, changing random sequence

Instruction symbol	Processing details	Reference
RND	Generates a random number between 0 and less than 32767, and stores the random number in the device specified by (d).	Page 847 RND(P)
RNDP		
SRND	Changes the random number sequence according to the content of the 16-bit binary data stored in the device specified by (s).	Page 848 SRND(P)
SRNDP		

Index register instructions

■Saving/returning all data of the index register

Instruction symbol	Processing details	Reference
ZPUSH	Saves data of the index register to the area specified by (d).	Page 849 ZPUSH(P)
ZPUSHP		
ZPOP	Reads the data, which has been saved to the area specified by (d)and later, into the index register.	Page 851 ZPOP(P)
ZPOPP		

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

Instruction symbol	Processing details	Reference
ZPUSH	Saves the contents of the index register and long index register specified by (s) to the area specified by (d).	Page 852 ZPUSH(P)
ZPUSHP		
ZPOP	Reads the data, which has been saved to the area specified by (d), into the index register and long index register.	Page 855 ZPOP(P)
ZPOPP		

Data control instructions

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

Instruction symbol	Processing details	Reference
LIMIT	Controls the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits.	Page 857 LIMIT(P)(_U)
LIMITP		
LIMIT_U		
LIMITP_U		
DLIMIT	Controls the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits.	Page 859 DLIMIT(P)(_U)
DLIMITP		
DLIMIT_U		
DLIMITP_U		

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

Instruction symbol	Processing details	Reference
BAND	Controls the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits of the dead band.	Page 861 BAND(P)(_U)
BANDP		
BAND_U		
BANDP_U		
DBAND	Controls the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits of the dead band.	Page 863 DBAND(P)(_U)
DBANDP		
DBAND_U		
DBANDP_U		

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

Instruction symbol	Processing details	Reference
ZONE	Adds a bias value to the specified input value (16-bit binary).	Page 865 ZONE(P)(_U)
ZONEP		
ZONE_U		
ZONEP_U		
DZONE	Adds a bias value to the specified input value (32-bit binary).	Page 867 DZONE(P)(_U)
DZONEP		
DZONE_U		
DZONEP_U		

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

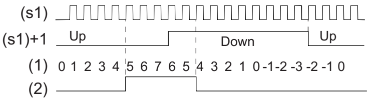
Instruction symbol	Processing details	Reference
SCL	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 869 SCL(P)(_U)
SCLP		
SCL_U		
SCLP_U		
DSCL	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 872 DSCL(P)(_U)
DSCLP		
DSCL_U		
DSCLP_U		

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

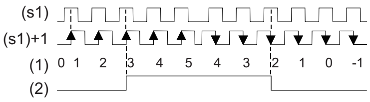
Instruction symbol	Processing details	Reference
SCL2	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 875 SCL2(P)(_U)
SCL2P		
SCL2_U		
SCL2P_U		
DSCL2	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 877 DSCL2(P)(_U)
DSCL2P		
DSCL2_U		
DSCL2P_U		

Special counter instructions

■Counting up or down the current value (1-phase input)

Instruction symbol	Processing details	Reference
UDCNT1	Updates the current value of the specified counter.  (1) Current value of Cn (2) Contact of Cn	Page 879 UDCNT1

■Counting up or down the current value (2-phase input)

Instruction symbol	Processing details	Reference
UDCNT2	Updates the current value of the counter depending on the status of phases A and B pulses.  (1) Current value of Cn (2) Contact of Cn	Page 881 UDCNT2

Special timer instructions

■Teaching timer

Instruction symbol	Processing details	Reference
TTMR	Measures the on time of the measurement command in seconds, multiplies it by a multiplier, and stores the operation result. $\begin{tikzcd} T_{ON}\times(s) \arrow[r] & (d) \\ (s)=0:1,\ (s)=1:10,\ (s)=2:100 \arrow[u] & \end{tikzcd}$ T_{ON}: On time of TTMR	Page 883 TTMR

■Special function timer

Instruction symbol	Processing details	Reference
STMR	Performs the following operations at the four points from the bit device specified by (d) according to on/off of the input condition of the instruction. • (d)+0: Off delay timer output • (d)+1: After-off one-shot timer output • (d)+2: After-on one-shot timer output • (d)+3: On delay + off delay timer	Page 885 STMR

Shortcut control instruction

■Rotary table shortest direction control

Instruction symbol	Processing details	Reference
ROTC	Controls the rotary table divided by (n1) so that it makes s shortcut rotation from the stop position to the position specified by (s)+1.	Page 888 ROTC

Ramp signal instruction

■Ramp signal

Instruction symbol	Processing details	Reference
RAMPQ	Changes the value specified by (s1) to the value specified by (s2) by the number of times specified by (n). The current value is stored in the device specified by (d1)+0.	Page 891 RAMPQ

Pulse related instructions

■Measuring the density of pulses

Instruction symbol	Processing details	Reference
SPD	Counts the input pulses in the device specified by (s1) for the period specified by (s2), and stores the result data in the device specified by (d).	Page 894 SPD

■Outputting pulses at regular intervals

Instruction symbol	Processing details	Reference
PLSY	Outputs the pulses at the frequency specified by (s), by the number of times specified by (n), to the output number (Y) in the device specified by (d).	Page 896 PLSY

■Performing the pulse width modulation

Instruction symbol	Processing details	Reference
PWM	Outputs the on time specified by (s1) and the pulses in the period specified by (s2) to the output number (Y) in the device specified by (d).	Page 898 PWM

Matrix input instruction

■Matrix input

Instruction symbol	Processing details	Reference
MTR	Reads 16 points by (n) columns of data from the device specified by (s), and stores it in the device specified by (d2) and later.	Page 900 MTR

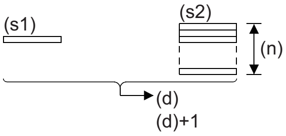
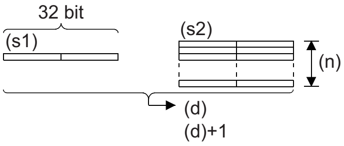
Check code instructions

■Check code

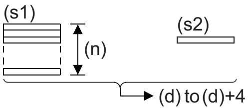
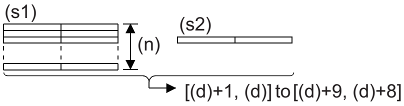
Instruction symbol	Processing details	Reference
CCD	Performs addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1.	Page 903 CCD(P)
CCDP		

Data processing instructions

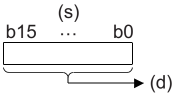
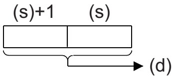
■Searching 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SERDATA	Searches the (n) points in the 16-bit binary data specified by (s2) for the 16-bit binary data specified by (s1).  (d): Location (d)+1: Number of matches	Page 906 SERDATA(P)
SERDATAP		
DSERDATA	Searches the (n) points in the 32-bit binary data specified by (s2) for the 32-bit binary data specified by (s1).  (d): Location (d)+1: Number of matches	Page 908 DSERDATA(P)
DSERDATAP		

■Searching 16-bit/32-bit binary data (minimum, match, maximum)

Instruction symbol	Processing details	Reference
SERMM	Searches the (n) points in the 16-bit binary data specified by (s1) for the same data as the 16-bit binary data specified by (s2), the minimum value, and the maximum value. 	Page 910 SERMM(P)
SERMMP		
DSERMM	Searches the (n) points in the 32-bit binary data specified by (s1) for the same data as the 32-bit binary data specified by (s2), the minimum value, and the maximum value. 	Page 912 DSERMM(P)
DSERMMP		

■Checking 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SUM	Stores the total number of "1" bits in the 16-bit binary data stored in the specified device.  (d): Number of 1s	Page 914 SUM(P)
SUMP		
DSUM	Stores the total number of "1" bits in the 32-bit binary data stored in the specified device.  (d): Number of 1s	Page 916 DSUM(P)
DSUMP		

■Checking the bit status in 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
BON	Checks whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).	Page 918 BON(P)
BONP		
DBON	Checks whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).	Page 920 DBON(P)
DBONP		

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

Instruction symbol	Processing details	Reference
MAX	Searches the (n) points of data in the device specified by (s) in units of 16 bits, and stores the maximum value in the device specified by (d).	Page 922 MAX(P)(_U)
MAXP		
MAX_U		
MAXP_U		
DMAX	Searches the (n) points of data in the device specified by (s) in units of 32 bits, and stores the maximum value in the device specified by (d).	Page 924 DMAX(P)(_U)
DMAXP		
DMAX_U		
DMAXP_U		

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

Instruction symbol	Processing details	Reference
MIN	Searches the (n) points of data in the device specified by (s) in units of 16 bits, and stores the minimum value in the device specified by (d).	Page 926 MIN(P)(_U)
MINP		
MIN_U		
MINP_U		
DMIN	Searches the (n) points of data in the device specified by (s) in units of 32 bits, and stores the minimum value in the device specified by (d).	Page 928 DMIN(P)(_U)
DMINP		
DMIN_U		
DMINP_U		

■Sorting 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SORTD	Sorts the (n) points of data in the device specified by (s) in units of 16 bits. ("(n)×((n)-1)÷2" scanning required)	Page 930 SORTD(_U)
SORTD_U		
DSORTD	Sorts the (n) points of data in the device specified by (s) in units of 32 bits. ("(n)×((n)-1)÷2" scanning required)	Page 932 DSORTD(_U)
DSORTD_U		

■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 934 WSUM(P)(_U)
WSUM_U		
WSUMP		
WSUMP_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 936 DWSUM(P)(_U)
DWSUM_U		
DWSUMP		
DWSUMP_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 938 MEAN(P)(_U)
MEANP		
MEAN_U		
MEANP_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 940 DMEAN(P)(_U)
DMEANP		
DMEAN_U		
DMEANP_U		

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

Instruction symbol	Processing details	Reference
SQRT	Performs a square root operation of the specified 16-bit binary data. $\sqrt{(s)}\rightarrow(d)$	Page 942 SQRT(P)
SQRTP		
DSQRT	Performs a square root operation of the specified 32-bit binary data. $\sqrt{(s)+1, (s)}\rightarrow(d)$	Page 943 DSQRT(P)
DSQRTP		

■CRC operation

Instruction symbol	Processing details	Reference
CRC	Generates the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d).	Page 944 CRC(P)
CRCP		

Database access instructions

■Opening the database

Instruction symbol	Processing details	Reference
DBOPEN	Connects to the database specified by (s), and makes it available.	Page 952 DBOPEN(P)
DBOPENP		

■Closing the database

Instruction symbol	Processing details	Reference
DBCLOSE	Releases the identification number specified by (s) and the allocation of the database.	Page 954 DBCLOSE(P)
DBCLOSEP		

■Adding a record to the database

Instruction symbol	Processing details	Reference
DBINSERT	Adds a record to the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 956 DBINSERT(P)
DBINSERTP		

■Updating the record in the database

Instruction symbol	Processing details	Reference
DBUPDATE	Updates all record that meets the condition specified by (s5) in the table specified by (s2) in the database specified by the identification number specified by (s1).	Page 962 DBUPDATE(P)
DBUPDATEP		

■Searching the record in the database

Instruction symbol	Processing details	Reference
DBSELECT	Searches the records in the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 968 DBSELECT(P)
DBSELECTP		

■Deleting the record in the database

Instruction symbol	Processing details	Reference
DBDELETE	Deletes the record that meets the condition specified by (s3) in the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 976 DBDELETE(P)
DBDELETEP		

■Importing data to the database

Instruction symbol	Processing details	Reference
DBIMPORT	Imports the data set in the Unicode text file stored in the path specified by (s) to construct a database.	Page 946 DBIMPORT(P)
DBIMPORTP		

■Exporting data from the database

Instruction symbol	Processing details	Reference
DBEXPORT	Exports the data stored in the database to the Unicode text file stored in the path specified by (s).	Page 949 DBEXPORT(P)
DBEXPORTP		

■Starting a transaction

Instruction symbol	Processing details	Reference
DBTRANS	Declares the start of a transaction in relation to the database corresponding to the identification number specified by (s).	Page 980 DBTRANS(P)
DBTRANSP		

■Committing a transaction

Instruction symbol	Processing details	Reference
DBCOMMIT	Commits the transaction in relation to the database corresponding to the identification number specified by (s).	Page 982 DBCOMMIT(P)
DBCOMMITP		

■Performing a rollback

Instruction symbol	Processing details	Reference
DBROLBAK	Executes the rollback in relation to the database corresponding to the identification number specified by (s).	Page 984 DBROLBAK(P)
DBROLBAKP		

File register operation instructions

■Switching the file register block number

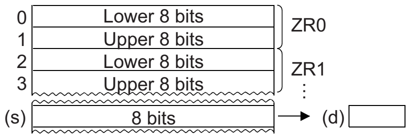
Instruction symbol	Processing details	Reference
RSET	Switches the block number of the file register used in the program to that stored in the device specified by (s).	Page 991 RSET(P)
RSETP		

■Changing the file register file name

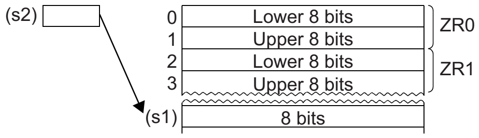
Instruction symbol	Processing details	Reference
QDRSET	Changes the file name of the file register used in the program.	Page 993 QDRSET(P)
QDRSETP		

File register read/write instructions

■Reading 1-byte data from the file register

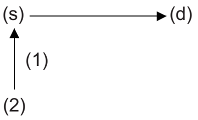
Instruction symbol	Processing details	Reference
ZRRDB	Reads the data from the file register with the specified serial byte number. 	Page 995 ZRRDB(P)
ZRRDBP		

■Writing 1-byte data to the file register

Instruction symbol	Processing details	Reference
ZRWRB	Writes the data in the lower bits of the specified device to the file register with the specified serial byte number. 	Page 997 ZRWRB(P)
ZRWRBP		

Indirect address read instructions

■Reading the indirect address

Instruction symbol	Processing details	Reference
ADRSET ADRSETP	Reads the indirect address of the specified device.  (1) Indirect address of specified device (2) Device name	Page 999 ADRSET(P)

Clock instructions

■Reading clock data

Instruction symbol	Processing details	Reference
DATERD DATERDP	Reads "year, month, day, hour, minute, second, and day of week" from the clock element of the CPU module. (d) (d)+1 (d)+2 (d)+3 (d)+4 (d)+5 (d)+6 (d): Year (d)+1: Month (d)+2: Day (d)+3: Hour (d)+4: Minute (d)+5: Second (d)+6: Day of week	Page 1001 DATERD(P)

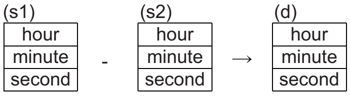
■Writing clock data

Instruction symbol	Processing details	Reference
DATEWR DATEWRP	Writes the clock data stored in the specified device and later to the clock element of the CPU module. (s) (s)+1 (s)+2 (s)+3 (s)+4 (s)+5 (s)+6 (s): Year (s)+1: Month (s)+2: Day (s)+3: Hour (s)+4: Minute (s)+5: Second (s)+6: Day of week	Page 1003 DATEWR(P)

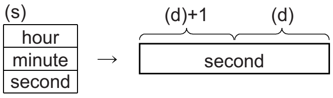
■Adding clock data

Instruction symbol	Processing details	Reference
DATE+ DATE+P	Adds time data. (s1) hour, minute, second + (s2) hour, minute, second → (d) hour, minute, second	Page 1005 DATE+(P)

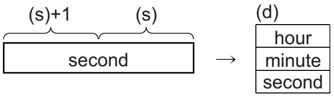
■Subtracting clock data

Instruction symbol	Processing details	Reference
DATE-	Subtracts time data.	Page 1007 DATE-(P)
DATE-P		

■Converting time data from hour/minute/second to second

Instruction symbol	Processing details	Reference
TIME2SEC	Converts time data from hour/minute/second to second.	Page 1009 TIME2SEC(P)
TIME2SECP		

■Converting time data from second to hour/minute/second

Instruction symbol	Processing details	Reference
SEC2TIME	Converts time data from second to hour/minute/second.	Page 1011 SEC2TIME(P)
SEC2TIMEP		

■Comparing date data

Instruction symbol	Processing details	Reference
LDDT=, ANDDT=, ORDT=	Compares the specified date data, or compares the date data with the current date.	Page 1013 LDDT☐, ANDDT☐, ORDT☐
LDDT<>, ANDDT<>, ORDT<>		
LDDT>, ANDDT>, ORDT>		
LDDT<=, ANDDT<=, ORDT<=		
LDDT<, ANDDT<, ORDT<		
LDDT>=, ANDDT>=, ORDT>=		

■Comparing time data

Instruction symbol	Processing details	Reference
LDTM=, ANDTM=, ORTM=	Compares the specified time data, or compares the specified time data with the current time.	Page 1017 LDTM☐, ANDTM☐, ORTM☐
LDTM<>, ANDTM<>, ORTM<>		
LDTM>, ANDTM>, ORTM>		
LDTM<=, ANDTM<=, ORTM<=		
LDTM<, ANDTM<, ORTM<		
LDTM>=, ANDTM>=, ORTM>=		

■Outputting a comparison result of time data

Instruction symbol	Processing details	Reference
TCMP	Compares the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.	Page 1021 TCMP(P)
TCMPP		

■Outputting a band comparison result of time data

Instruction symbol	Processing details	Reference
TZCP	Compares the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 1023 TZCP(P)
TZCPP		

■Reading expansion clock data

Instruction symbol	Processing details	Reference
S.DATERD SP.DATERD	Reads clock data including millisecond from the clock elements in the CPU module. (d) (d)+1 (d)+2 (d)+3 (d)+4 (d)+5 (d)+6 (d)+7 (d): Year (d)+1: Month (d)+2: Day (d)+3: Hour (d)+4: Minute (d)+5: Second (d)+6: Day of week (d)+7: Millisecond	Page 1025 S(P).DATERD

■Adding expansion clock data

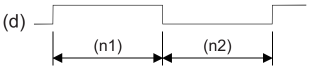
Instruction symbol	Processing details	Reference
S.DATE+ SP.DATE+	Adds time data. (s1): hour, minute, second, —, 1/1000 second + (s2): hour, minute, second, —, 1/1000 second → (d): hour, minute, second, —, 1/1000 second	Page 1027 S(P).DATE+

■Subtracting expansion clock data

Instruction symbol	Processing details	Reference
S.DATE- SP.DATE-	Subtracts time data. (s1): hour, minute, second, —, 1/1000 second - (s2): hour, minute, second, —, 1/1000 second → (d): hour, minute, second, —, 1/1000 second	Page 1029 S(P).DATE-

Timing check instructions

■Generating timing pulses

Instruction symbol	Processing details	Reference
DUTY	Turns on the user timing clock for the specified number of scans and off for the specified number of scans.  (n1): (n1) scans (n2): (n2) scans (d): SM420 to SM424	Page 1031 DUTY

■Measuring time of the specified data

Instruction symbol	Processing details	Reference
TIMCHK	Measures the on time of the input condition and, if the on time has continued as specified or longer, turns on the device specified by (d).	Page 1033 TIMCHK

■Hour meter

Instruction symbol	Processing details	Reference
HOURM	Measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).	Page 1035 HOURM
DHOURM	Measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).	Page 1037 DHOURM

Module access instructions

■Performing I/O refresh

Instruction symbol	Processing details	Reference
RFS	Performs partial refresh of the relevant input/output during one scan.	Page 1039 RFS(P)
RFSP		

■Selecting refresh to be performed

Instruction symbol	Processing details	Reference
COM	Performs refresh and service processing for various modules when the input condition is met.	Page 1041 COM(P)
COMP		

■Performing module refresh

Instruction symbol	Processing details	Reference
S.ZCOM	Performs refresh processing for the specified modules.	Page 1043 S(P).ZCOM
SP.ZCOM		

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

Instruction symbol	Processing details	Reference
FROM	Reads (n) words of data in units of 16 bits from the buffer memory areas of the intelligent function module and other CPU modules.	Page 1045 FROM(P), DFROM(P)
FROMP		
DFROM	Reads (n)×2 words of data in units of 16 bits from the buffer memory areas of the intelligent function module and other CPU modules.	
DFROMP		

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

Instruction symbol	Processing details	Reference
TO	Writes (n) words of data in units of 16 bits to the buffer memory areas of the intelligent function module and own CPU module.	Page 1049 TO(P), DTO(P)
TOP		
DTO	Writes (n)×2 words of data in units of 16 bits to the buffer memory areas of the intelligent function module and own CPU module.	
DTOP		

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

Instruction symbol	Processing details	Reference
FROMD	Reads (n) words of data in units of 32 bits from the buffer memory areas of the intelligent function module and other CPU modules.	Page 1054 FROMD(P), DFROMD(P)
FROMDP		
DFROMD	Reads (n)×2 words of data in units of 32 bits from the buffer memory areas of the intelligent function module and other CPU modules.	
DFROMDP		

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

Instruction symbol	Processing details	Reference
TOD	Writes (n) words of data in units of 32 bits to the buffer memory areas of the intelligent function module and own CPU module.	Page 1058 TOD(P), DTOD(P)
TODP		
DTOD	Writes (n)×2 words of data in units of 32 bits to the buffer memory areas of the intelligent function module and own CPU module.	
DTODP		

■Reading the module model name

Instruction symbol	Processing details	Reference
TYPERD TYPERDP	Reads the module model name mounted on the slot specified by (H), and stores the model name in the device areas specified by (d) and later.	Page 1063 TYPERD(P)

■Reading module specific information

Instruction symbol	Processing details	Reference
UNIINFRD UNIINFRDP	Reads the module information by the number of points specified by (n) from the module specified by (H), and stores the information in the device areas specified by (d) and later.	Page 1067 UNIINFRD(P)

Routing information instructions

■Reading routing information

Instruction symbol	Processing details	Reference
S.RTREAD SP.RTREAD	Reads the data set by routing parameters.	Page 1072 S(P).RTREAD

■Registering routing information

Instruction symbol	Processing details	Reference
S.RTWRITE SP.RTWRITE	Registers the routing information to the area specified by a routing parameter.	Page 1074 S(P).RTWRITE

Logging instructions

■Setting/resetting trigger logging

Instruction symbol	Processing details	Reference
LOGTRG	Generates a trigger for trigger logging. Data sampled for the number of records (specified in the trigger logging setting parameter using the engineering tool) are stored in the data logging file.	Page 1076 LOGTRG
LOGTRGR	Resets the trigger condition.	Page 1078 LOGTRGR

Program control instructions

■Changing the program execution type to standby type

Instruction symbol	Processing details	Reference
PSTOP PSTOPP	Changes the type of the specified program to standby type.	Page 1079 PSTOP(P)

■Changing the program execution type to standby type (output off)

Instruction symbol	Processing details	Reference
POFF POFFP	Turns off the coil of the OUT instruction used in the specified program and changes the type of the specified program to standby type.	Page 1081 POFF(P)

■Changing the program execution type to scan execution type

Instruction symbol	Processing details	Reference
PSCAN PSCANP	Changes the type of the specified program to scan execution type.	Page 1083 PSCAN(P)

2.4 Built-in Ethernet Function Instructions

Open/close processing instructions

■Opening a connection

Instruction symbol	Processing details	Reference
SP.SOCOPEN	Opens the connection specified by (s1).	Page 1085 SP.SOCOPEN

■Closing a connection

Instruction symbol	Processing details	Reference
SP.SOCCLOSE	Closes the connection specified by (s1). (Closing a connection)	Page 1088 SP.SOCCLOSE

Socket communications instructions

■Reading receive data during the END processing

Instruction symbol	Processing details	Reference
SP.SOCRCV	Reads the receive data of the connection specified by (s1) during END processing from the socket communication receive data area.	Page 1090 SP.SOCRCV

■Reading receive data when the instruction is executed

Instruction symbol	Processing details	Reference
S.SOCRCVS	Reads the receive data of the connection specified by (s) during instruction execution from the socket communication receive data area.	Page 1093 S.SOCRCVS

■Sending data

Instruction symbol	Processing details	Reference
SP.SOCSND	Sends the data in the device specified by (s3) to the external device of the connection specified by (s1).	Page 1096 SP.SOCSND

■Reading connection information

Instruction symbol	Processing details	Reference
SP.SOCCINF	Reads the connection information of the connection specified by (s1).	Page 1099 SP.SOCCINF

■Changing the communication target (UDP/IP)

Instruction symbol	Processing details	Reference
SP.SOCCSET	Changes the communication target IP address and port number of the connection specified by (s1). (UDP/IP communications only)	Page 1101 SP.SOCCSET

■Changing the receive mode

Instruction symbol	Processing details	Reference
SP.SOCRMODE	Changes the TCP receive mode and receive data size for the connection specified by (s1).	Page 1103 SP.SOCRMODE

■Reading socket communications receive data

Instruction symbol	Processing details	Reference
S.SOCRDATA	Reads data by the number of words specified by (n) from the socket communication receive data area of the connection specified by (s1), and stores them in the device specified by (d) and later.	Page 1107 S(P).SOCRDATA
SP.SOCRDATA		

Predefined protocol support function instruction

■Executing the registered protocols

Instruction symbol	Processing details	Reference
SP.ECPRTCL	Executes the protocol specified by the communication protocol support tool of the engineering tool.	Page 1109 SP.ECPRTCL

SLMP frame send instruction

■Sending an SLMP frame

Instruction symbol	Processing details	Reference
SP.SLMPSND	Sends SLMP messages to the SLMP-compatible device.	Page 1117 SP.SLMPSND

File transfer function instructions

■Sending FTP client files

Instruction symbol	Processing details	Reference
SP.FTPPUT	Sends files in the CPU module, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).	Page 1123 SP.FTPPUT

■Retrieving FTP client files

Instruction symbol	Processing details	Reference
SP.FTPGET	Retrieves files on the FTP server, which are specified by (s2), to the folder path of the CPU module, which is specified by (s3).	Page 1128 SP.FTPGET

2.5 PID Operation Instruction

Performing PID operation

Instruction symbol	Processing details	Reference
PID	Performs PID operation using the values set in (s1) to (s3), and stores the operation result in (d) at each cycle of sampling time.	Page 1145 PID

2.6 PID Control Instructions

PID control instructions (inexact differential)

■Registering the PID control data to the CPU module

Instruction symbol	Processing details	Reference
S.PIDINIT	Stores the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the CPU module to enable PID control.	Page 1160 S(P).PIDINIT
SP.PIDINIT		

■Performing PID operation

Instruction symbol	Processing details	Reference
S.PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1163 S(P).PIDCONT
SP.PIDCONT		

■Stopping/starting the operation of specified loop number

Instruction symbol	Processing details	Reference
S.PIDSTOP	Stops the PID operation of the loop number in the device specified by (s).	Page 1166 S(P).PIDSTOP
SP.PIDSTOP		
S.PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1167 S(P).PIDRUN
SP.PIDRUN		

■Changing the parameters of specified loop number

Instruction symbol	Processing details	Reference
S.PIDPRMW	Changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.	Page 1168 S(P).PIDPRMW
SP.PIDPRMW		

PID control instructions (exact differential)

■Registering the PID control data to the CPU module

Instruction symbol	Processing details	Reference
PIDINIT	Stores the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the CPU module to enable PID control.	Page 1172 PIDINIT(P)
PIDINITP		

■Performing PID operation

Instruction symbol	Processing details	Reference
PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1174 PIDCONT(P)
PIDCONTP		

■Stopping/starting the operation of specified loop number

Instruction symbol	Processing details	Reference
PIDSTOP	Stops the PID operation of the loop number in the device specified by (s).	Page 1177 PIDSTOP(P)
PIDSTOPP		
PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1178 PIDRUN(P)
PIDRUNP		

■Changing the parameters of specified loop number

Instruction symbol	Processing details	Reference
PIDPRMW	Changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.	Page 1179 PIDPRMW(P)
PIDPRMWP		

2.7 Process Control Instructions

Point

When a process control program is created, using process control function blocks is recommended.

Process control function blocks have features as follows.

- A process control program can be easily created by placing and connecting FB elements.
- Since the initial value of the function block can be set in the "FB Property" window of the engineering tool, the program for the initial value setting is not required.
- An operation constant can be input to a label indicating a tag name without being conscious of address of a device.
- The operating status of a tag FB can be checked and controlled by accessing the tag data from the faceplate of an engineering tool.

For details on the process control function blocks, refer to the following.

📖 MELSEC iQ-R Programming Manual (Process Control Function Blocks)

I/O control instructions

■Analog input processing

Instruction symbol	Processing details	Reference
S.IN	Performs following processing to the input data (PV): range check, input limiter, engineering value transformation, and digital filter.	Page 1197 S.IN

■Output processing 1 with mode switching

Instruction symbol	Processing details	Reference
S.OUT1	Calculates MV (0 to 100%) from the input data (△MV), and performs the variation rate & upper/lower limiter processing and output conversion processing.	Page 1203 S.OUT1

■Output processing 2 with mode switching

Instruction symbol	Processing details	Reference
S.OUT2	Performs variation rate & upper/lower limiter processing and output conversion processing to the input data (MV).	Page 1209 S.OUT2

■Manual output

Instruction symbol	Processing details	Reference
S.MOUT	Reads the manipulated value (MV) from the loop tag memory and performs output conversion.	Page 1214 S.MOUT

■Time proportioning

Instruction symbol	Processing details	Reference
S.DUTY	Outputs ON and OFF by changing the ON/OFF ratio in a given cycle in proportion to the input data (0 to 100%).	Page 1217 S.DUTY

■Batch counter

Instruction symbol	Processing details	Reference
S.BC	Compares the input data with the set value, and outputs bit data when it reaches the set value.	Page 1223 S.BC

■Pulse integration

Instruction symbol	Processing details	Reference
S.PSUM	Integrates the number of input pulses, and outputs the operation result.	Page 1227 S.PSUM

Control operation instructions

■Basic PID control

Instruction symbol	Processing details	Reference
S.PID	Performs process value derivative type (inexact differential) PID operation. The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, PID operation, and deviation check.	Page 1232 S.PID

■Two-degree-of-freedom PID control

Instruction symbol	Processing details	Reference
S.2PID	Performs two-degree-of-freedom PID control operation (inexact differential). The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, two-degree-of-freedom PID control operation, and deviation check.	Page 1239 S.2PID

■Position type PID control

Instruction symbol	Processing details	Reference
S.PIDP	Performs position type PID operation. The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, PID operation, deviation check, and control mode determination. Depending on the operation result up to the mode determination processing, the instruction performs either variation rate & upper/lower limiter and output conversion, or alarm clear and output conversion processing.	Page 1246 S.PIDP

■Sample PI control

Instruction symbol	Processing details	Reference
S.SPI	Monitors the operating time (operating time (ST) or hold time (HT)), and performs the following processing steps: SV setting, tracking, gain (Kp) operation, SPI operation, and deviation check if the operating time is in ST.	Page 1254 S.SPI

■I-PD control

Instruction symbol	Processing details	Reference
S.IPD	Performs I-PD operation. The instruction performs the following processing steps: SV setting, tracking, gain Kp operation, IPD operation, and deviation check.	Page 1262 S.IPD

■Blend PI control

Instruction symbol	Processing details	Reference
S.BPI	Performs blend PI operation. The instruction performs the following processing steps: SV setting, tracking, gain Kp operation, BPI operation, and deviation check.	Page 1269 S.BPI

■Ratio calculation

Instruction symbol	Processing details	Reference
S.R	Performs the following processing steps to the input data: engineering value transformation, tracking, variation rate limiter, and ratio calculation.	Page 1276 S.R

■Upper/lower limit alarm

Instruction symbol	Processing details	Reference
S.PHPL	Checks the range of the PV that has been output by the S.IN instruction.	Page 1281 S.PHPL

■Lead-lag compensation

Instruction symbol	Processing details	Reference
S.LLAG	Performs lead-lag compensation to the input data, and outputs the operation result.	Page 1287 S.LLAG

■Integral control

Instruction symbol	Processing details	Reference
S.I	Performs integration operation to the input data, and outputs the operation result.	Page 1290 S.I

■Derivative control

Instruction symbol	Processing details	Reference
S.D	Performs differential operation to the input data, and outputs the operation result.	Page 1292 S.D

■Dead time

Instruction symbol	Processing details	Reference
S.DED	Delays the output of the input data by the specified dead time.	Page 1295 S.DED

■High selector

Instruction symbol	Processing details	Reference
S.HS	Outputs the maximum value of the input data.	Page 1299 S.HS

■Low selector

Instruction symbol	Processing details	Reference
S.LS	Outputs the minimum value of the input data.	Page 1301 S.LS

■Middle value selector

Instruction symbol	Processing details	Reference
S.MID	Outputs the intermediate value (the value between the maximum and minimum values) of the input data.	Page 1303 S.MID

■Average value calculation

Instruction symbol	Processing details	Reference
S.AVE	Calculates the mean (average) value of the input data, and outputs the operation result.	Page 1306 S.AVE

■Upper/lower limiter

Instruction symbol	Processing details	Reference
S.LIMT	Applies a limiter with hysteresis to the output value.	Page 1308 S.LIMT

■Variation rate limiter 1

Instruction symbol	Processing details	Reference
S.VLMT1	Limits the variation speed when the variation rate of the input (E1) exceeds the limit.	Page 1311 S.VLMT1

■Variation rate limiter 2

Instruction symbol	Processing details	Reference
S.VLMT2	Holds the last value when the variation rate of the input (E1) exceeds the limit.	Page 1314 S.VLMT2

■Two-position (on/off) control

Instruction symbol	Processing details	Reference
S.ONF2	Performs two-position (on/off) control. The instruction performs the following processing steps: SV setting, tracking, MV correction, MV output, and two-position (on/off) control.	Page 1317 S.ONF2

■Three-position (on/off) control

Instruction symbol	Processing details	Reference
S.ONF3	Performs three-position (on/off) control. The instruction performs the following processing steps: SV setting, tracking, MV correction, MV output, and three-position (on/off) control.	Page 1323 S.ONF3

■Dead band

Instruction symbol	Processing details	Reference
S.DBND	Provides a dead band and performs output processing.	Page 1330 S.DBND

■Program setter

Instruction symbol	Processing details	Reference
S.PGS	Provides control output according to the SV and MV patterns.	Page 1332 S.PGS

■Loop selector

Instruction symbol	Processing details	Reference
S.SEL	Outputs the value selected by the selection signal from the input data in automatic mode, and outputs the manipulated value (MV) in the loop tag memory in manual mode.	Page 1338 S.SEL

■Bumpless transfer

Instruction symbol	Processing details	Reference
S.BUMP	Gradually brings the output value (BW) closer to the output set value (E1) from the output control value (E2) when the mode switching signal (e1) changes from manual to automatic.	Page 1344 S.BUMP

■Analog memory

Instruction symbol	Processing details	Reference
S.AMR	Increases or decreases the output value (BW) at a fixed rate.	Page 1347 S.AMR

Correction operation instructions

■Function generator

Instruction symbol	Processing details	Reference
S.FG	Outputs the input data values according to the specified function generator pattern.	Page 1350 S.FG

■Inverse function generator

Instruction symbol	Processing details	Reference
S.IFG	Outputs the input data values according to the specified inverse function generator pattern.	Page 1353 S.IFG

■Standard filter

Instruction symbol	Processing details	Reference
S.FLT	Outputs the mean (average) value of the 'n' pieces of data collected at the specified data collection intervals (ST).	Page 1356 S.FLT

■Integration

Instruction symbol	Processing details	Reference
S.SUM	Integrates the input data, and outputs the operation result.	Page 1359 S.SUM

■Temperature/pressure correction

Instruction symbol	Processing details	Reference
S.TPC	Performs temperature/pressure correction processing to the input data.	Page 1361 S.TPC

■Engineering value transformation

Instruction symbol	Processing details	Reference
S.ENG	Performs engineering value transformation processing to the input data.	Page 1364 S.ENG

■Engineering value inverse transformation

Instruction symbol	Processing details	Reference
S.IENG	Performs engineering value inverse transformation processing to the input data.	Page 1366 S.IENG

Arithmetic operation instructions

■Addition

Instruction symbol	Processing details	Reference
S.ADD	Adds input data with a coefficient.	Page 1368 S.ADD

■Subtraction

Instruction symbol	Processing details	Reference
S.SUB	Subtracts input data with a coefficient.	Page 1370 S.SUB

■Multiplication

Instruction symbol	Processing details	Reference
S.MUL	Multiplies input data with a coefficient.	Page 1372 S.MUL

■Division

Instruction symbol	Processing details	Reference
S.DIV	Divides input data with a coefficient.	Page 1374 S.DIV

■Square root

Instruction symbol	Processing details	Reference
S.SQR	Outputs the square root (√) of input data.	Page 1376 S.SQR

■Absolute value

Instruction symbol	Processing details	Reference
S.ABS	Outputs the absolute value of input data.	Page 1378 S.ABS

Comparison operation instructions

■Comparing data

Instruction symbol	Processing details	Reference
S.>	Compares input data, and outputs the comparison result.	Page 1380 S.>
S.<		Page 1382 S.<
S.=		Page 1384 S.=
S.>=		Page 1386 S.>=
S.<=		Page 1388 S.<=

Auto tuning

Instruction symbol	Processing details	Reference
S.AT1	Performs auto tuning to make initial setting of PID constants.	Page 1393 S.AT1

2.8 Multiple CPU Dedicated Instructions

Reading device data from another CPU module

Instruction symbol	Processing details	Reference
D.DDRD	Reads the data in the device of another CPU module specified by (n), and stores the data to the read-source CPU module in a multiple CPU system.	Page 1405 D(P).DDRD, M(P).DDRD
DP.DDRD		
M.DDRD		
MP.DDRD		

Writing device data to another CPU module

Instruction symbol	Processing details	Reference
D.DDWR	Writes the data specified in the own CPU module to another CPU module specified by (n) in a multiple CPU system.	Page 1408 D(P).DDWR, M(P).DDWR
DP.DDWR		
M.DDWR		
MP.DDWR		

Motion CPU dedicated instructions

For available Motion CPU dedicated instructions, refer to the following.

📖 MELSEC iQ-R Motion Controller Programming Manual (Program Design)

2.9 SFC Program Instructions

SFC control instructions

■Checking the status of a step

Instruction symbol	Processing details	Reference
LD [S□/BL□\S□]	Outputs the status (active or inactive) of the specified step as the operation result. (Normally open contact instruction)	Page 1411 LD, LDI, AND, ANI, OR, ORI [S□/BL□\S□]
LDI [S□/BL□\S□]	Outputs the status (active or inactive) of the specified step as the operation result. (Normally closed contact instruction)	
AND [S□/BL□\S□]	Performs an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Normally open contact series connection instruction)	
ANI [S□/BL□\S□]	Performs an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Normally closed contact series connection instruction)	
OR [S□/BL□\S□]	Performs an OR operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Single normally open contact parallel connection instruction)	
ORI [S□/BL□\S□]	Performs an OR operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Single normally closed contact parallel connection instruction)	

■Checking the status of a block

Instruction symbol	Processing details	Reference
LD [BL□]	Outputs the status (active or inactive) of the specified block as the operation result. (Normally open contact instruction)	Page 1414 LD, LDI, AND, ANI, OR, ORI [BL□]
LDI [BL□]	Outputs the status (active or inactive) of the specified block as the operation result. (Normally closed contact instruction)	
AND [BL□]	Performs an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Normally open contact series connection instruction)	
ANI [BL□]	Performs an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Normally closed contact series connection instruction)	
OR [BL□]	Performs an OR operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Single normally open contact parallel connection instruction)	
ORI [BL□]	Performs an OR operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Single normally closed contact parallel connection instruction)	

■Batch-reading the status of steps

Instruction symbol	Processing details	Reference
MOV [K4S□/BL□\K4S□]	Batch-reads (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block, and stores the read data in the specified device.	Page 1416 MOV(P) [K4S□/BL□\K4S□]
MOVP [K4S□/BL□\K4S□]		
DMOV [K8S□/BL□\K8S□]	Batch-reads (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block, and stores the read data in the specified device.	Page 1419 DMOV(P) [K8S□/BL□\K8S□]
DMOVP [K8S□/BL□\K8S□]		
BMOV [K4S□/BL□\K4S□]	Batch-reads (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.	Page 1422 BMOV(P) [K4S□/BL□\K4S□]
BMOVP [K4S□/BL□\K4S□]		

■Starting a block

Instruction symbol	Processing details	Reference
SET [BL□]	Activates the specified block, and executes a step sequence starting from an initial step.	Page 1425 SET [BL□]

■Ending a block

Instruction symbol	Processing details	Reference
RST [BL□]	Deactivates the specified block.	Page 1427 RST [BL□]

■Pausing a block

Instruction symbol	Processing details	Reference
PAUSE [BL□]	Temporarily stops the step sequence in the specified block.	Page 1429 PAUSE [BL□]

■Restarting a block

Instruction symbol	Processing details	Reference
RSTART [BL□]	Releases the temporary stop, and restarts the sequence from the step where the sequence was stopped in the specified block.	Page 1431 RSTART [BL□]

■Activating a step

Instruction symbol	Processing details	Reference
SET [S□/BL□\S□]	Activates the specified step.	Page 1433 SET [S□/BL□\S□]

■Deactivating a step

Instruction symbol	Processing details	Reference
RST [S□/BL□\S□]	Deactivates the specified step.	Page 1435 RST [S□/BL□\S□]

■Switching a target block

Instruction symbol	Processing details	Reference
BRSET	Specifies an SFC control instruction target block No.	Page 1437 BRSET

SFC dedicated instruction

■Creating a dummy transition condition

Instruction symbol	Processing details	Reference
TRAN	A dummy output which satisfies a transition condition	Page 1439 TRAN

2.10 Redundant System Instructions

System switching

Instruction symbol	Processing details	Reference
SP.CONTSW	Switches the systems (control system and standby system) during END processing of the scan where the instruction is executed.	Page 1440 SP.CONTSW

Disabling/enabling system switching

Instruction symbol	Processing details	Reference
DCONTSW	Disables manual system switching.	Page 1444 DCONTSW, ECONTSW
ECONTSW	Enables manual system switching.	

3 MODULE DEDICATED INSTRUCTIONS

How to read the list is shown below.

Item	Description
Instruction symbol	Indicates the instruction name.
Processing details	Indicates the overview of an instruction.
Reference	Indicates the reference of detailed information.

3.1 Network Common Instructions

Link dedicated instructions

■Reading data from the programmable controller on another station

Instruction symbol	Processing details	Reference
JP.READ GP.READ	Reads data in units of words from a device in the programmable controller of another station.	Page 1454 JP.READ, GP.READ

■Reading data from the programmable controller on another station (with notification)

Instruction symbol	Processing details	Reference
JP.SREAD GP.SREAD	Reads data in units of words from a device in the programmable controller of another station. After the data reading is completed, the device of another station is turned on. (The other station can recognize that data has been read by the SREAD instruction.)	Page 1461 JP.SREAD, GP.SREAD

■Writing data to the programmable controller on another station

Instruction symbol	Processing details	Reference
JP.WRITE GP.WRITE	Writes data in units of words to a device in the programmable controller of another station.	Page 1469 JP.WRITE, GP.WRITE

■Writing data to the programmable controller on another station (with notification)

Instruction symbol	Processing details	Reference
JP.SWRITE GP.SWRITE	Writes data in units of words to a device in the programmable controller of another station. After the data writing is completed, the device of another station is turned on. (The other station can recognize that data has been written by the SWRITE instruction.)	Page 1477 JP.SWRITE, GP.SWRITE

■Sending data to the programmable controller on another station

Instruction symbol	Processing details	Reference
JP.SEND GP.SEND	Sends data to the programmable controller of another station.	Page 1485 JP.SEND, GP.SEND

■Receiving data from the programmable controller on another station

Instruction symbol	Processing details	Reference
JP.RECV GP.RECV	Reads data received from the programmable controller of another station. (For the main routine program)	Page 1492 JP.RECV, GP.RECV

■Receiving data from the programmable controller on another station (for interrupt programs)

Instruction symbol	Processing details	Reference
G.RECVS Z.RECVS	Reads data received from the programmable controller of another station. (For interrupt programs)	Page 1497 G.RECVS, Z.RECVS

■Reading data from the programmable controller (Q series-compatible)

Instruction symbol	Processing details	Reference
J.ZNRD	Reads data in units of words from a device in the programmable controller of another station.	Page 1501 J(P).ZNRD
JP.ZNRD		

■Writing data to the programmable controller (Q series-compatible)

Instruction symbol	Processing details	Reference
J.ZNWR	Writes data in units of words to a device in the programmable controller of another station.	Page 1506 J(P).ZNWR
JP.ZNWR		

■Remote RUN/STOP

Instruction symbol	Processing details	Reference
J.REQ	Executes remote RUN or STOP for the programmable controller of another station.	Page 1511 J(P).REQ, G(P).REQ
JP.REQ		
G.REQ		
GP.REQ		

■Reading/writing clock data

Instruction symbol	Processing details	Reference
J.REQ	Reads/writes clock data from/to the programmable controller of another station.	Page 1518 J(P).REQ, G(P).REQ
JP.REQ		
G.REQ		
GP.REQ		

CC-Link dedicated instructions

■Reading data from the target station

Instruction symbol	Processing details	Reference
J.RIRD	Reads the specified number of points of data from a device of the target station.	Page 1525 J(P).RIRD, G(P).RIRD
JP.RIRD		
G.RIRD		
GP.RIRD		

■Writing data to the target station

Instruction symbol	Processing details	Reference
J.RIWT	Writes the specified number of points of data to a device of the target station.	Page 1530 J(P).RIWT, G(P).RIWT
JP.RIWT		
G.RIWT		
GP.RIWT		

Restriction

- When the target station is QSCPU, the following network common instructions cannot be used.

SEND, RECV, RECVS, J(P).ZNRD, J(P).ZNWR, and REQ (remote RUN/STOP)

- Data cannot be written from the CPU module on another station to the QSCPU by using the following network common instructions.

WRITE, SWRITE, REQ (reading/writing clock data), and RIWT

- If the CPU module on the target station is AnUCPU, A2USCPU(-S1), or A2ASCPU(-S1) when the J(P).ZNRD or J(P).ZNWR instruction is executed, the CPU module must be the one with the following version or later.

AnUCPU: Version AY (manufactured in July 1995) or later

A2USCPU(-S1): Version CP (manufactured in July 1995) or later

3.2 Ethernet Instructions

Open/close processing instructions

■Opening a connection

Instruction symbol	Processing details	Reference
GP.CONOPEN	Establishes (opens) a connection with an external device for data communication.	Page 1535 GP.CONOPEN

■Closing a connection

Instruction symbol	Processing details	Reference
GP.CONCLOSE	Disconnects (closes) the connection from the external device during data communication.	Page 1539 GP.CONCLOSE

■Opening a connection

Instruction symbol	Processing details	Reference
GP.OPEN	Establishes (opens) a connection with an external device for data communication.	Page 1541 GP.OPEN, ZP.OPEN
ZP.OPEN		

■Closing a connection

Instruction symbol	Processing details	Reference
GP.CLOSE	Disconnects (closes) the connection from the external device during data communication.	Page 1545 GP.CLOSE, ZP.CLOSE
ZP.CLOSE		

Socket communications instructions

■Reading receive data

Instruction symbol	Processing details	Reference
GP.SOCRCV	Reads receive data from the external device through socket communications or fixed buffer communications. This instruction is used in the main program.	Page 1548 GP.SOCRCV

■Reading receive data (for interrupt programs)

Instruction symbol	Processing details	Reference
G.SOCRCVS	Reads receive data from the external device through socket communications or fixed buffer communications. This instruction is used in the interrupt program.	Page 1550 G.SOCRCVS

■Sending data

Instruction symbol	Processing details	Reference
GP.SOCSND	Sends data to the external device through socket communications or fixed buffer communications. This instruction is used in the main program.	Page 1552 GP.SOCSND

Fixed buffer communications instructions

■Reading receive data

Instruction symbol	Processing details	Reference
GP.BUFRCV	Reads receive data from the external device through fixed buffer communications. This instruction is used in the main program.	Page 1554 GP.BUFRCV, ZP.BUFRCV
ZP.BUFRCV		

■Reading receive data (for interrupt programs)

Instruction symbol	Processing details	Reference
G.BUFRCVS Z.BUFRCVS	Reads receive data from the external device through fixed buffer communications. This instruction is used in the interrupt program.	Page 1557 G.BUFRCVS, Z.BUFRCVS

■Sending data

Instruction symbol	Processing details	Reference
GP.BUFSND ZP.BUFSND	Sends data to the external device through fixed buffer communications.	Page 1559 GP.BUFSND, ZP.BUFSND

Reinitializing the module

Instruction symbol	Processing details	Reference
G.UINI GP.UINI Z.UINI ZP.UINI	Changes the setting details such as Ethernet operation settings or reinitializes the module.	Page 1562 G(P).UINI, Z(P).UINI

Executing the protocols registered for the predefined protocol support function

Instruction symbol	Processing details	Reference
GP.ECPRTCL	Executes the protocol that has been set by the predefined protocol support function.	Page 1566 GP.ECPRTCL

Clearing error information

Instruction symbol	Processing details	Reference
GP.ERRCLEAR ZP.ERRCLEAR	Clears error information.	Page 1571 GP.ERRCLEAR, ZP.ERRCLEAR

Reading error information

Instruction symbol	Processing details	Reference
GP.ERRRD ZP.ERRRD	Reads error information.	Page 1574 GP.ERRRD, ZP.ERRRD

3.3 CC-Link IE Controller Network Instructions

Remote RUN

Instruction symbol	Processing details	Reference
J.RRUN	Sends a remote RUN request to the programmable controller of another station.	Page 1576 J(P).RRUN, G(P).RRUN, Z(P).RRUN
JP.RRUN		
G.RRUN		
GP.RRUN		
Z.RRUN		
ZP.RRUN		

Remote STOP

Instruction symbol	Processing details	Reference
J.RSTOP	Sends a remote STOP request to the programmable controller of another station.	Page 1581 J(P).RSTOP, G(P).RSTOP, Z(P).RSTOP
JP.RSTOP		
G.RSTOP		
GP.RSTOP		
Z.RSTOP		
ZP.RSTOP		

Reading clock data from the programmable controller on another station

Instruction symbol	Processing details	Reference
J.RTMRD	Reads clock data from the programmable controller of another station.	Page 1586 J(P).RTMRD, G(P).RTMRD, Z(P).RTMRD
JP.RTMRD		
G.RTMRD		
GP.RTMRD		
Z.RTMRD		
ZP.RTMRD		

Writing clock data to the programmable controller on another station

Instruction symbol	Processing details	Reference
J.RTMWR	Writes clock data to the programmable controller of another station.	Page 1591 J(P).RTMWR, G(P).RTMWR, Z(P).RTMWR
JP.RTMWR		
G.RTMWR		
GP.RTMWR		
Z.RTMWR		
ZP.RTMWR		

Setting the station number to own station

Instruction symbol	Processing details	Reference
G.UINI	Sets the station number of the own station.	Page 1597 G(P).UINI, Z(P).UINI
GP.UINI		
Z.UINI		
ZP.UINI		

Restriction

- When the target station is QSCPU, the RRUN and RSTOP instructions cannot be used.
- The RTMWR instruction cannot write to QSCPU of other stations.

3.4 CC-Link IE Field Network Instructions

Reading data from the intelligent device station/remote device station

Instruction symbol	Processing details	Reference
JP.REMFR ZP.REMFR	Reads data from the buffer memory area of the intelligent device station or the remote device station in units of words (16-bit address specified).	Page 1600 JP.REMFR, ZP.REMFR
JP.REMFRD	Reads data from the buffer memory area of the intelligent device station or the remote device station in units of words (32-bit address specified).	Page 1610 JP.REMFRD

Writing data to the intelligent device station/remote device station

Instruction symbol	Processing details	Reference
JP.REMTO ZP.REMTO	Writes data to the buffer memory area of the intelligent device station or the remote device station in units of words (16-bit address specified).	Page 1605 JP.REMTO, ZP.REMTO
JP.REMTOD	Writes data to the buffer memory area of the intelligent device station or the remote device station in units of words (32-bit address specified).	Page 1615 JP.REMTOD

Reading the module model information of an intelligent device station

Instruction symbol	Processing details	Reference
J.SINFTYRD JP.SINFTYRD G.SINFTYRD GP.SINFTYRD	Reads the model names of modules and units used in an intelligent device station.	Page 1620 J(P).SINFTYRD, G(P)SINFTYRD

Reading the module status information of an intelligent device station

Instruction symbol	Processing details	Reference
J.SINFSTRD JP.SINFSTRD G.SINFSTRD GP.SINFSTRD	Reads the status information of modules and units used in an intelligent device station.	Page 1624 J(P).SINFSTRD, G(P)SINFSTRD

Setting parameters

Instruction symbol	Processing details	Reference
G.CCPASET GP.CCPASET	Sets parameters to the master, submaster, and local stations. Use the G(P).CCPASET instruction to set parameters in the following cases. • To change parameters without resetting the CPU module • To install more modules than can be set by the engineering tool	Page 1629 G(P).CCPASET

Setting the station number to own station

Instruction symbol	Processing details	Reference
G.UINI GP.UINI Z.UINI ZP.UINI	Sets the station number to the local station (own station). Use the engineering tool or UINI instruction to set network parameters excluding the station number.	Page 1635 G(P).UINI, Z(P).UINI

Sending an SLMP frame

Instruction symbol	Processing details	Reference
J.SLMPREQ	Sends an SLMP frame to the SLMP-compatible device in the same network. Data of the target station can be read/written and operated.	Page 1638 J(P).SLMPREQ, G(P).SLMPREQ
JP.SLMPREQ		
G.SLMPREQ		
GP.SLMPREQ		

Restriction

The REMFR, REMTO, REMFRD, and REMTOD instructions cannot be executed in local stations. Execute them in the master station.

The submaster function can be executed only in the master operating station.

Use the own station master/submaster function operating status (SB004E) to apply an interlock so that the submaster function is executed only in the master operating station. (Master operating station when SB004E is off)

3.5 CC-Link Instructions

Reading data from the target station

Instruction symbol	Processing details	Reference
G.RIRD GP.RIRD	Reads data of the specified number of points from the buffer memory area of the target station or the device in the CPU module on the target station.	Page 1643 G(P).RIRD

Writing data to the target station

Instruction symbol	Processing details	Reference
G.RIWT GP.RIWT	Writes data of the specified number of points to the buffer memory area of the target station or the device in the CPU module on the target station.	Page 1647 G(P).RIWT

Reading data from the buffer memory of the specified intelligent device station

Instruction symbol	Processing details	Reference
G.RIRCV GP.RIRCV	Automatically performs handshake with the specified intelligent device station, and reads data from its buffer memory. The instructions can be used for communications with a module supporting a handshake signal such as AJ65BT-R2N.	Page 1651 G(P).RIRCV

Writing data to the buffer memory of the specified intelligent device station

Instruction symbol	Processing details	Reference
G.RISEND GP.RISEND	Automatically performs handshake with the specified intelligent device station, and writes data to its buffer memory. The instructions can be used for communications with a module supporting a handshake signal such as AJ65BT-R2N.	Page 1654 G(P).RISEND

Reading data from the automatic update buffer

Instruction symbol	Processing details	Reference
G.RIFR GP.RIFR	Reads data from an automatic update or random access buffer. The instructions can be used for communications with a module having the automatic update buffer such as AJ65BT-R2N.	Page 1657 G(P).RIFR

Writing data to the automatic update buffer

Instruction symbol	Processing details	Reference
G.RITO GP.RITO	Writes data to an automatic update or random access buffer. The instructions can be used for communications with a module having the automatic update buffer such as AJ65BT-R2N.	Page 1659 G(P).RITO

Setting network parameters

Instruction symbol	Processing details	Reference
G.RLPASET GP.RLPASET	Sets network parameters in the master station and starts up the data link.	Page 1661 G(P).RLPASET

Performing a message transmission to a remote device station

Instruction symbol	Processing details	Reference
G.RDMSG GP.RDMSG	Reads/writes parameters from/to the remote device station and reads the status of the remote device station. The instructions can be executed to a remote device station, for example NZ2AW1C2AL, that supports the message transmission function.	Page 1667 G(P).RDMSG

3.6 Serial Communication Instructions

Sending data using the on-demand function

Instruction symbol	Processing details	Reference
G.ONDEMAND GP.ONDEMAND	Sends data using the on-demand function of SLMP (MC protocol).	Page 1673 G(P).ONDEMAND

Executing the protocols registered for the predefined protocol support function

Instruction symbol	Processing details	Reference
G.CPRTCL GP.CPRTCL	Executes the protocol registered using the engineering tool.	Page 1677 G(P).CPRTCL

Sending data using the nonprocedural protocol

Instruction symbol	Processing details	Reference
G.OUTPUT GP.OUTPUT	Sends data in user-specified message format through communications using the nonprocedural protocol.	Page 1682 G(P).OUTPUT

Receiving data using the nonprocedural protocol

Instruction symbol	Processing details	Reference
G.INPUT	Receives data in user-specified message format through communications using the nonprocedural protocol.	Page 1685 G.INPUT

Sending data using the bidirectional protocol

Instruction symbol	Processing details	Reference
G.BIDOUT GP.BIDOUT	Sends the specified amount of data through communications using the bidirectional protocol.	Page 1688 G(P).BIDOUT

Receiving data using the bidirectional protocol

Instruction symbol	Processing details	Reference
G.BIDIN GP.BIDIN	Receives data through communications using the bidirectional protocol.	Page 1691 G(P).BIDIN

Reading the data send/receive status

Instruction symbol	Processing details	Reference
G.SPBUSY GP.SPBUSY	Reads the status of data sent/received by using the serial communication instruction.	Page 1694 G(P).SPBUSY

Receiving data using the interrupt program

Instruction symbol	Processing details	Reference
Z.BUFRCVS	Receives data using the interrupt program through communications under control of the nonprocedural protocol or bidirectional protocol.	Page 1696 Z.BUFRCVS

Sending data by using user frames

Instruction symbol	Processing details	Reference
G.PRR	Sends data with user frames according to the specification in the user frame specification area for sending, through communication with the nonprocedural protocol.	Page 1699 G(P).PRR
GP.PRR		

Clearing receive data

Instruction symbol	Processing details	Reference
ZP.CSET	Clears the receive data area without stopping send processing by using the nonprocedural protocol.	Page 1702 ZP.CSET

Registering/canceling the programmable controller CPU monitoring

Instruction symbol	Processing details	Reference
ZP.CSET	Registers programmable controller CPU monitoring to use the programmable controller CPU monitoring function, or cancels programmable controller CPU monitoring.	Page 1705 ZP.CSET

Initial setting

Instruction symbol	Processing details	Reference
ZP.CSET	Sets the unit (word/byte) of data to be sent or received.	Page 1710 ZP.CSET

Registering user frames

Instruction symbol	Processing details	Reference
G.PUTE	Registers user frames.	Page 1714 G(P).PUTE
GP.PUTE		

Reading user frames

Instruction symbol	Processing details	Reference
G.GETE	Reads user frames.	Page 1717 G(P).GETE
GP.GETE		

Switching the mode

Instruction symbol	Processing details	Reference
ZP.UINI	Switches the serial communication module mode, transmission specifications, and own station number.	Page 1721 ZP.UINI

3.7 A/D Conversion Instructions

Switching the mode

Instruction symbol	Processing details	Reference
G.OFFGAN GP.OFFGAN	Switches the analog module mode. • Normal mode to offset/gain setting mode • Offset/gain setting mode to normal mode	Page 1724 G(P).OFFGAN

Reading the user range setting values

Instruction symbol	Processing details	Reference
G.OGLOAD GP.OGLOAD	Reads the offset/gain setting values of the user range settings of an analog module into the CPU module.	Page 1727 G(P).OGLOAD

Restoring the user range setting values

Instruction symbol	Processing details	Reference
G.OGSTOR GP.OGSTOR	Restores the offset/gain settings in the user range setting stored in the CPU module into an analog module.	Page 1746 G(P).OGSTOR

3.8 Positioning Instructions

Restoring the absolute position

Instruction symbol	Processing details	Reference
G.ABRST1	Restores the absolute position of specified axis.	Page 1765 G.ABRST□, Z.ABRST□
G.ABRST2		
G.ABRST3		
G.ABRST4		
Z.ABRST1		
Z.ABRST2		
Z.ABRST3		
Z.ABRST4		

Starting the positioning

Instruction symbol	Processing details	Reference
GP.PSTRT1	Starts positioning of the specified axis.	Page 1769 GP.PSTRT□, ZP.PSTRT□
GP.PSTRT2		
GP.PSTRT3		
GP.PSTRT4		
ZP.PSTRT1		
ZP.PSTRT2		
ZP.PSTRT3		
ZP.PSTRT4		

Teaching

Instruction symbol	Processing details	Reference
GP.TEACH1	Performs teaching for the specified axis.	Page 1772 GP.TEACH□, ZP.TEACH□
GP.TEACH2		
GP.TEACH3		
GP.TEACH4		
ZP.TEACH1		
ZP.TEACH2		
ZP.TEACH3		
ZP.TEACH4		

Backing up module data (writing data to the flash ROM)

Instruction symbol	Processing details	Reference
GP.PFWRT	Writes the positioning data and block start data in the buffer memory to the flash ROM.	Page 1775 GP.PFWRT, ZP.PFWRT
ZP.PFWRT		

Initializing the module

Instruction symbol	Processing details	Reference
GP.PINIT	Initializes the setting data in the buffer memory and flash ROM.	Page 1778 GP.PINIT, ZP.PINIT
ZP.PINIT		

3.9 High Speed Data Logger Module Instructions

File access instructions

■Recipe write

Instruction symbol	Processing details	Reference
Z.RCPWRITE	Writes device values of the CPU module to the specified recipe file in the SD memory card.	Page 1781 Z(P).RCPWRITE
ZP.RCPWRITE		

■Recipe read

Instruction symbol	Processing details	Reference
Z.RCPREAD	Reads device values of the specified recipe file in the SD memory card to the CPU module.	Page 1784 Z(P).RCPREAD
ZP.RCPREAD		

3.10 C Intelligent Function Module Instructions

User function execution instruction

Instruction symbol	Processing details	Reference
G.CEXECUTE	Instructs to execute a function for the pre-registered function.	Page 1787 G(P).CEXECUTE
GP.CEXECUTE		

4 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.

4.1 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 1792 BOOL_TO_WORD(_E)
BOOL_TO_WORD_E		
BOOL_TO_DWORD	Converts a value from BOOL data type to DWORD data type.	Page 1794 BOOL_TO_DWORD(_E)
BOOL_TO_DWORD_E		

■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 1795 BOOL_TO_INT(_E)
BOOL_TO_INT_E		
BOOL_TO_DINT	Converts a value from BOOL data type to DINT data type.	Page 1796 BOOL_TO_DINT(_E)
BOOL_TO_DINT_E		

■Converting BOOL to TIME

Function symbol	Processing details	Reference
BOOL_TO_TIME	Converts a value from BOOL data type to TIME data type.	Page 1797 BOOL_TO_TIME(_E)
BOOL_TO_TIME_E		

■Converting BOOL to STRING

Function symbol	Processing details	Reference
BOOL_TO_STRING	Converts a value from BOOL data type to STRING data type.	Page 1798 BOOL_TO_STRING(_E)
BOOL_TO_STRING_E		

■Converting WORD to BOOL

Function symbol	Processing details	Reference
WORD_TO_BOOL	Converts a value from WORD data type to BOOL data type.	Page 1799 WORD_TO_BOOL(_E)
WORD_TO_BOOL_E		

■Converting WORD to DWORD

Function symbol	Processing details	Reference
WORD_TO_DWORD	Converts a value from WORD data type to DWORD data type.	Page 1800 WORD_TO_DWORD(_E)
WORD_TO_DWORD_E		

■Converting WORD to INT/DINT

Function symbol	Processing details	Reference
WORD_TO_INT WORD_TO_INT_E	Converts a value from WORD data type to INT data type.	Page 1801 WORD_TO_INT(_E)
WORD_TO_DINT WORD_TO_DINT_E	Converts a value from WORD data type to DINT data type.	Page 1802 WORD_TO_DINT(_E)

■Converting WORD to TIME

Function symbol	Processing details	Reference
WORD_TO_TIME WORD_TO_TIME_E	Converts a value from WORD data type to TIME data type.	Page 1804 WORD_TO_TIME(_E)

■Converting WORD to STRING

Function symbol	Processing details	Reference
WORD_TO_STRING WORD_TO_STRING_E	Converts a value from WORD data type to STRING data type.	Page 1805 WORD_TO_STRING(_E)

■Converting DWORD to BOOL

Function symbol	Processing details	Reference
DWORD_TO_BOOL DWORD_TO_BOOL_E	Converts a value from DWORD data type to BOOL data type.	Page 1806 DWORD_TO_BOOL(_E)

■Converting DWORD to WORD

Function symbol	Processing details	Reference
DWORD_TO_WORD DWORD_TO_WORD_E	Converts a value from DWORD data type to WORD data type.	Page 1807 DWORD_TO_WORD(_E)

■Converting DWORD to INT/DINT

Function symbol	Processing details	Reference
DWORD_TO_INT DWORD_TO_INT_E	Converts a value from DWORD data type to INT data type.	Page 1809 DWORD_TO_INT(_E)
DWORD_TO_DINT DWORD_TO_DINT_E	Converts a value from DWORD data type to DINT data type.	Page 1811 DWORD_TO_DINT(_E)

■Converting DWORD to TIME

Function symbol	Processing details	Reference
DWORD_TO_TIME DWORD_TO_TIME_E	Converts a value from DWORD data type to TIME data type.	Page 1812 DWORD_TO_TIME(_E)

■Converting DWORD to STRING

Function symbol	Processing details	Reference
DWORD_TO_STRING DWORD_TO_STRING_E	Converts a value from DWORD data type to STRING data type.	Page 1813 DWORD_TO_STRING(_E)

■Converting INT to BOOL

Function symbol	Processing details	Reference
INT_TO_BOOL INT_TO_BOOL_E	Converts a value from INT data type to BOOL data type.	Page 1814 INT_TO_BOOL(_E)

■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 1815 INT_TO_WORD(_E)
INT_TO_WORD_E		
INT_TO_DWORD	Converts a value from INT data type to DWORD data type.	Page 1816 INT_TO_DWORD(_E)
INT_TO_DWORD_E		

■Converting INT to DINT

Function symbol	Processing details	Reference
INT_TO_DINT	Converts a value from INT data type to DINT data type.	Page 1818 INT_TO_DINT(_E)
INT_TO_DINT_E		

■Converting INT to BCD

Function symbol	Processing details	Reference
INT_TO_BCD	Converts a value from INT data type to BCD data type.	Page 1819 INT_TO_BCD(_E)
INT_TO_BCD_E		

■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 1821 INT_TO_REAL(_E)
INT_TO_REAL_E		
INT_TO_LREAL	Converts a value from INT data type to LREAL data type.	Page 1822 INT_TO_LREAL(_E)
INT_TO_LREAL_E		

■Converting INT to TIME

Function symbol	Processing details	Reference
INT_TO_TIME	Converts a value from INT data type to TIME data type.	Page 1823 INT_TO_TIME(_E)
INT_TO_TIME_E		

■Converting INT to STRING

Function symbol	Processing details	Reference
INT_TO_STRING	Converts a value from INT data type to STRING data type.	Page 1824 INT_TO_STRING(_E)
INT_TO_STRING_E		

■Converting DINT to BOOL

Function symbol	Processing details	Reference
DINT_TO_BOOL	Converts a value from DINT data type to BOOL data type.	Page 1826 DINT_TO_BOOL(_E)
DINT_TO_BOOL_E		

■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 1827 DINT_TO_WORD(_E)
DINT_TO_WORD_E		
DINT_TO_DWORD	Converts a value from DINT data type to DWORD data type.	Page 1829 DINT_TO_DWORD(_E)
DINT_TO_DWORD_E		

■Converting DINT to INT

Function symbol	Processing details	Reference
DINT_TO_INT	Converts a value from DINT data type to INT data type.	Page 1830 DINT_TO_INT(_E)
DINT_TO_INT_E		

■Converting DINT to BCD

Function symbol	Processing details	Reference
DINT_TO_BCD DINT_TO_BCD_E	Converts a value from DINT data type to BCD data type.	Page 1831 DINT_TO_BCD(_E)

■Converting DINT to REAL/LREAL

Function symbol	Processing details	Reference
DINT_TO_REAL DINT_TO_REAL_E	Converts a value from DINT data type to REAL data type.	Page 1833 DINT_TO_REAL(_E)
DINT_TO_LREAL DINT_TO_LREAL_E	Converts a value from DINT data type to LREAL data type.	Page 1834 DINT_TO_LREAL(_E)

■Converting DINT to TIME

Function symbol	Processing details	Reference
DINT_TO_TIME DINT_TO_TIME_E	Converts a value from DINT data type to TIME data type.	Page 1835 DINT_TO_TIME(_E)

■Converting DINT to STRING

Function symbol	Processing details	Reference
DINT_TO_STRING DINT_TO_STRING_E	Converts a value from DINT data type to STRING data type.	Page 1836 DINT_TO_STRING(_E)

■Converting BCD to INT/DINT

Function symbol	Processing details	Reference
BCD_TO_INT BCD_TO_INT_E	Converts a value from BCD data type to INT data type.	Page 1838 BCD_TO_INT(_E)
BCD_TO_DINT BCD_TO_DINT_E	Converts a value from BCD data type to DINT data type.	Page 1840 BCD_TO_DINT(_E)

■Converting BCD to STRING

Function symbol	Processing details	Reference
BCD_TO_STRING BCD_TO_STRING_E	Converts a value from BCD data type to STRING data type.	Page 1843 BCD_TO_STRING(_E)

■Converting REAL to INT/DINT

Function symbol	Processing details	Reference
REAL_TO_INT REAL_TO_INT_E	Converts a value from REAL data type to INT data type.	Page 1845 REAL_TO_INT(_E)
REAL_TO_DINT REAL_TO_DINT_E	Converts a value from REAL data type to DINT data type.	Page 1847 REAL_TO_DINT(_E)

■Converting REAL to LREAL

Function symbol	Processing details	Reference
REAL_TO_LREAL REAL_TO_LREAL_E	Converts a value from REAL data type to LREAL data type.	Page 1849 REAL_TO_LREAL(_E)

■Converting REAL to STRING

Function symbol	Processing details	Reference
REAL_TO_STRING REAL_TO_STRING_E	Converts a REAL data type value to STRING data type (exponential form).	Page 1851 REAL_TO_STRING(_E)

■Converting LREAL to INT/DINT

Function symbol	Processing details	Reference
LREAL_TO_INT LREAL_TO_INT_E	Converts a value from LREAL data type to INT data type.	Page 1854 LREAL_TO_INT(_E)
LREAL_TO_DINT LREAL_TO_DINT_E	Converts a value from LREAL data type to DINT data type.	Page 1856 LREAL_TO_DINT(_E)

■Converting LREAL to REAL

Function symbol	Processing details	Reference
LREAL_TO_REAL LREAL_TO_REAL_E	Converts a value from LREAL data type to REAL data type.	Page 1858 LREAL_TO_REAL(_E)

■Converting TIME to BOOL

Function symbol	Processing details	Reference
TIME_TO_BOOL TIME_TO_BOOL_E	Converts a value from TIME data type to BOOL data type.	Page 1860 TIME_TO_BOOL(_E)

■Converting TIME to WORD/DWORD

Function symbol	Processing details	Reference
TIME_TO_WORD TIME_TO_WORD_E	Converts a value from TIME data type to WORD data type.	Page 1861 TIME_TO_WORD(_E)
TIME_TO_DWORD TIME_TO_DWORD_E	Converts a value from TIME data type to DWORD data type.	Page 1862 TIME_TO_DWORD(_E)

■Converting TIME to INT/DINT

Function symbol	Processing details	Reference
TIME_TO_INT TIME_TO_INT_E	Converts a value from TIME data type to INT data type.	Page 1863 TIME_TO_INT(_E)
TIME_TO_DINT TIME_TO_DINT_E	Converts a value from TIME data type to DINT data type.	Page 1864 TIME_TO_DINT(_E)

■Converting TIME to STRING

Function symbol	Processing details	Reference
TIME_TO_STRING TIME_TO_STRING_E	Converts a value from TIME data type to STRING data type.	Page 1865 TIME_TO_STRING(_E)

■Converting STRING to BOOL

Function symbol	Processing details	Reference
STRING_TO_BOOL STRING_TO_BOOL_E	Converts a value from STRING data type to BOOL data type.	Page 1867 STRING_TO_BOOL(_E)

■Converting STRING to WORD/DWORD

Function symbol	Processing details	Reference
STRING_TO_WORD STRING_TO_WORD_E	Converts a value from STRING data type to WORD data type.	Page 1868 STRING_TO_WORD(_E)
STRING_TO_DWORD STRING_TO_DWORD_E	Converts a value from STRING data type to DWORD data type.	Page 1869 STRING_TO_DWORD(_E)

■Converting STRING to INT/DINT

Function symbol	Processing details	Reference
STRING_TO_INT STRING_TO_INT_E	Converts a value from STRING data type to INT data type.	Page 1870 STRING_TO_INT(_E)
STRING_TO_DINT STRING_TO_DINT_E	Converts a value from STRING data type to DINT data type.	Page 1872 STRING_TO_DINT(_E)

■Converting STRING to BCD

Function symbol	Processing details	Reference
STRING_TO_BCD STRING_TO_BCD_E	Converts a value from STRING data type to BCD data type.	Page 1874 STRING_TO_BCD(_E)

■Converting STRING to REAL

Function symbol	Processing details	Reference
STRING_TO_REAL STRING_TO_REAL_E	Converts a value from STRING data type to REAL data type.	Page 1876 STRING_TO_REAL(_E)

■Converting STRING to TIME

Function symbol	Processing details	Reference
STRING_TO_TIME STRING_TO_TIME_E	Converts a value from STRING data type to TIME data type.	Page 1879 STRING_TO_TIME(_E)

■Converting bit array to INT/DINT

Function symbol	Processing details	Reference
BITARR_TO_INT BITARR_TO_INT_E	Converts the specified number of bits in a bit array to an INT data type value.	Page 1881 BITARR_TO_INT(_E)
BITARR_TO_DINT BITARR_TO_DINT_E	Converts the specified number of bits in a bit array to a DINT data type value.	Page 1882 BITARR_TO_DINT(_E)

■Converting INT/DINT to bit array

Function symbol	Processing details	Reference
INT_TO_BITARR INT_TO_BITARR_E	Outputs the lower n bits of the INT data type value to the bit array.	Page 1883 INT_TO_BITARR(_E)
DINT_TO_BITARR DINT_TO_BITARR_E	Outputs the lower n bits of the DINT data type value to the bit array.	Page 1884 DINT_TO_BITARR(_E)

■Copying the bit array

Function symbol	Processing details	Reference
CPY_BITARR CPY_BITARR_E	Copies the bit array by the specified number of bits.	Page 1885 CPY_BITARR(_E)

■Reading/writing/copying the specified bit of the word label

Function symbol	Processing details	Reference
GET_BIT_OF_INT GET_BIT_OF_INT_E	Reads a value from the specified bit of a word label.	Page 1886 GET_BIT_OF_INT(_E)
SET_BIT_OF_INT SET_BIT_OF_INT_E	Writes a value to the specified bit of a word label.	Page 1888 SET_BIT_OF_INT(_E)
CPY_BIT_OF_INT CPY_BIT_OF_INT_E	Copies the specified bit of the word label to the specified bit of another word label.	Page 1890 CPY_BIT_OF_INT(_E)

■Getting the start data

Function symbol	Processing details	Reference
GET_BOOL_ADDR	Outputs the top data of the specified data as the BOOL, INT, or WORD type data.	Page 1892 GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR
GET_INT_ADDR		
GET_WORD_ADDR		

Single variable functions

■Calculating the absolute value

Function symbol	Processing details	Reference
ABS	Outputs the absolute value of an input value.	Page 1893 ABS(_E)
ABS_E		

■Calculating the square root

Function symbol	Processing details	Reference
SQRT	Calculates the square root of an input value.	Page 1895 SQRT(_E)
SQRT_E		

■Calculating the natural logarithm

Function symbol	Processing details	Reference
LN	Outputs the natural logarithm (logarithm with base e) of an input value.	Page 1896 LN(_E)
LN_E		

■Calculating the common logarithm

Function symbol	Processing details	Reference
LOG	Outputs the common logarithm (logarithm with base 10) of an input value.	Page 1897 LOG(_E)
LOG_E		

■Calculating the exponent

Function symbol	Processing details	Reference
EXP	Outputs the exponent of an input value.	Page 1899 EXP(_E)
EXP_E		

■Calculating the sine/cosine/tangent

Function symbol	Processing details	Reference
SIN	Outputs the sine of an input value.	Page 1900 SIN(_E)
SIN_E		
COS	Outputs the cosine of an input value.	Page 1901 COS(_E)
COS_E		
TAN	Outputs the tangent of an input value.	Page 1902 TAN(_E)
TAN_E		

■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 1903 ASIN(_E)
ASIN_E		
ACOS	Outputs the arc cosine (COS^{-1}) of an input value.	Page 1904 ACOS(_E)
ACOS_E		
ATAN	Outputs the arc tangent (TAN^{-1}) of an input value.	Page 1905 ATAN(_E)
ATAN_E		

Arithmetic operation functions

■Addition

Function symbol	Processing details	Reference
ADD	Outputs the sum of input values ((s1)+(s2)+···+(s28)).	Page 1906 ADD(_E)
ADD_E		

■Multiplication

Function symbol	Processing details	Reference
MUL	Outputs the product of input values ((s1)×(s2)×···×(s28)).	Page 1908 MUL(_E)
MUL_E		

■Subtraction

Function symbol	Processing details	Reference
SUB	Outputs the difference between input values ((s1)-(s2)).	Page 1910 SUB(_E)
SUB_E		

■Division

Function symbol	Processing details	Reference
DIV	Outputs the quotient of input values ((s1)÷(s2)).	Page 1912 DIV(_E)
DIV_E		

■Remainder

Function symbol	Processing details	Reference
MOD	Outputs the remainder of input values ((s1)÷(s2)).	Page 1914 MOD(_E)
MOD_E		

■Exponentiation

Function symbol	Processing details	Reference
EXPT	Outputs the exponentiation of an input value.	Page 1916 EXPT(_E)
EXPT_E		

■Assignment (move operation)

Function symbol	Processing details	Reference
MOVE	Outputs the assignment value of an input value.	Page 1917 MOVE(_E)
MOVE_E		

Bit shift functions

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

Function symbol	Processing details	Reference
SHL	Shifts the input value to the left by (n) bit(s), and outputs the operation result.	Page 1919 SHL(_E)
SHL_E		
SHR	Shifts the input value to the right by (n) bit(s), and outputs the operation result.	Page 1921 SHR(_E)
SHR_E		

■Rotating data to the left/right by n bit(s)

Function symbol	Processing details	Reference
ROL	Rotates the input value to the left by (n) bit(s), and outputs the operation result.	Page 1923 ROL(_E)
ROL_E		
ROR	Rotates the input value to the right by (n) bit(s), and outputs the operation result.	Page 1925 ROR(_E)
ROR_E		

Boolean functions

■AND operation, OR operation, XOR operation

Function symbol	Processing details	Reference
AND	Outputs the logical product of input values.	Page 1927 AND(_E), OR(_E), XOR(_E)
AND_E		
OR	Outputs the logical sum of input values.	
OR_E		
XOR	Outputs the exclusive logical sum of input values.	
XOR_E		

■NOT operation

Function symbol	Processing details	Reference
NOT	Outputs the logical NOT of input values.	Page 1930 NOT(_E)
NOT_E		

Selection functions

■Selecting a value

Function symbol	Processing details	Reference
SEL	Outputs the selected input value.	Page 1931 SEL(_E)
SEL_E		

■Selecting the maximum/minimum value

Function symbol	Processing details	Reference
MAX	Outputs the maximum input value.	Page 1933 MAX(_E), MIN(_E)
MAX_E		
MIN	Outputs the minimum input value.	
MIN_E		

■Controlling the upper/lower limit

Function symbol	Processing details	Reference
LIMIT	Outputs an input value that has been controlled in terms of the upper and lower limits.	Page 1935 LIMIT(_E)
LIMIT_E		

■Multiplexer

Function symbol	Processing details	Reference
MUX	Outputs one of the input values.	Page 1938 MUX(_E)
MUX_E		

Comparison functions

■Comparing data

Function symbol	Processing details	Reference
GT	Outputs the comparison result of input values.	Page 1940 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
GT_E		
GE		
GE_E		
EQ		
EQ_E		
LE		
LE_E		
LT		
LT_E		
NE		Page 1942 NE(_E)
NE_E		

String functions

■Detecting a string length

Function symbol	Processing details	Reference
LEN	Detects and outputs the length of the string input.	Page 1944 LEN(_E)
LEN_E		

■Extracting string data from the left/right

Function symbol	Processing details	Reference
LEFT	Extracts and outputs the specified number of characters, starting from the left end of the string input.	Page 1946 LEFT(_E), RIGHT(_E)
LEFT_E		
RIGHT	Extracts and outputs the specified number of characters, starting from the right end of the string input.	
RIGHT_E		

■Extracting string data

Function symbol	Processing details	Reference
MID	Extracts and outputs the specified number of characters, starting from the specified position of the string input.	Page 1949 MID(_E)
MID_E		

■Concatenating string data

Function symbol	Processing details	Reference
CONCAT	Concatenates character strings, and outputs the operation result.	Page 1951 CONCAT(_E)
CONCAT_E		

■Inserting string data

Function symbol	Processing details	Reference
INSERT	Inserts a character string into another string, and outputs the operation result.	Page 1953 INSERT(_E)
INSERT_E		

■Deleting string data

Function symbol	Processing details	Reference
DELETE	Deletes the specified range in a character string, and outputs the operation result.	Page 1955 DELETE(_E)
DELETE_E		

■Replacing string data

Function symbol	Processing details	Reference
REPLACE	Replaces the specified range in a character string, and outputs the operation result.	Page 1957 REPLACE(_E)
REPLACE_E		

■Searching string data

Function symbol	Processing details	Reference
FIND	Searches a character string, and outputs the operation result.	Page 1960 FIND(_E)
FIND_E		

Time data type functions

■Addition

Function symbol	Processing details	Reference
ADD_TIME	Outputs the sum ((s1)+(s2)) of the TIME data type input values.	Page 1962 ADD_TIME(_E)
ADD_TIME_E		

■Subtraction

Function symbol	Processing details	Reference
SUB_TIME	Outputs the difference ((s1)-(s2)) between the TIME data type input values.	Page 1964 SUB_TIME(_E)
SUB_TIME_E		

■Multiplication

Function symbol	Processing details	Reference
MUL_TIME	Outputs the product ((s1)×(s2)) of the TIME data type input values.	Page 1966 MUL_TIME(_E)
MUL_TIME_E		

■Division

Function symbol	Processing details	Reference
DIV_TIME	Outputs the quotient ((s1)÷(s2))of the TIME data type input values.	Page 1968 DIV_TIME(_E)
DIV_TIME_E		

4.2 Standard Function Flocks

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 1972 SR(_E)
SR_E		

■Bistable function block (reset-dominant)

Function block symbol	Processing details	Reference
RS	Discriminates between two input values, and outputs1 (TRUE) or 0 (FALSE).	Page 1974 RS(_E)
RS_E		

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 1976 R_TRIG(_E)
R_TRIG_E		

■Detecting a falling edge

Function block symbol	Processing details	Reference
F_TRIG	Detects a signal falling edge, and outputs the pulse signal.	Page 1978 F_TRIG(_E)
F_TRIG_E		

Counter function blocks

■Up counter

Function block symbol	Processing details	Reference
CTU	Counts up the number of rising edges of a signal.	Page 1980 CTU(_E)
CTU_E		

■Down counter

Function block symbol	Processing details	Reference
CTD	Counts down the number of rising edges of a signal.	Page 1982 CTD(_E)
CTD_E		

■Up/down counter

Function block symbol	Processing details	Reference
CTUD	Counts up or down the number of rising edges of a signal.	Page 1984 CTUD(_E)
CTUD_E		

■Counter function block

Function block symbol	Processing details	Reference
COUNTER_FB_M	Starts counting up when the execution condition is satisfied.	Page 1987 COUNTER_FB_M

Timer function blocks

■Pulse timer

Function block symbol	Processing details	Reference
TP	Keeps the signal on for the specified period of time.	Page 1989 TP(_E)
TP_E		

■On delay timer

Function block symbol	Processing details	Reference
TON	Turns on a signal after the specified period of time.	Page 1992 TON(_E)
TON_E		

■Off delay timer

Function block symbol	Processing details	Reference
TOF	Turns off a signal after the specified period of time.	Page 1995 TOF(_E)
TOF_E		

■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 1997 TIMER_□_M
TIMER_100_FB_M		
TIMER_HIGH_FB_M		
TIMER_LOW_FB_M		
TIMER_CONT_FB_M		
TIMER_CONTHFB_M		

PART 3

CPU MODULE INSTRUCTIONS

This part consists of the following chapters.

5 SEQUENCE INSTRUCTIONS

5.1 Contact Instructions

Operation start, series connection, parallel connection

LD, LDI, AND, ANI, OR, ORI

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- LD: Normally open contact operation start, LDI: Normally closed contact operation start

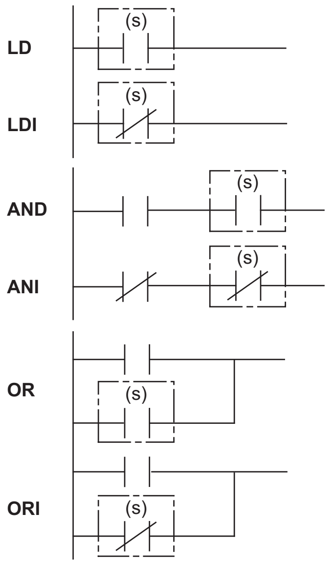
These instructions output the on/off information of the specified device as the operation result.

- AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result.

- OR: Single normally open contact parallel connection, ORI: Single normally closed contact parallel connection

These instructions perform an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
LD LDI AND ANI OR ORI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	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)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

Processing details

■LD, LDI

- LD is a normally open contact operation start instruction and LDI is a normally closed contact operation start instruction. These instructions read the on/off information[1] of the specified bit device, and output it as the operation result.

[1] When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

■AND, ANI

- AND is a normally open contact series connection instruction and ANI is a normally closed contact series connection instruction. These instructions read the on/off information[1] of the specified bit device, perform an AND operation with the previous operation result, and output the operation result.

[1] When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

- Note the following when creating or displaying a program using the engineering tool (in ladder edit mode).
  - Write mode: When the AND and ANI instructions are connected in series, a ladder with up to 24 steps can be created.
  - Read mode: When the AND and ANI instructions are connected in series, a ladder with up to 24 steps can be displayed. If there are more than 24 steps, up to 24 steps are displayed.

■OR, ORI

- OR is a single normally open contact parallel connection instruction and ORI is a single normally open contact parallel connection instruction. These instructions read the on/off information[1] of the specified bit device, perform an OR operation with the previous operation result, and output the operation result.

[1] When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

- Note the following when creating or displaying a program using the engineering tool (in ladder edit mode).
  - Write mode: Up to 23 OR and ORI instructions can be connected consecutively.
  - Read mode: Up to 23 OR and ORI instructions connected consecutively can be displayed. A ladder with more than 23 instructions cannot be displayed correctly.

Point

- Specify a bit in a word device in hexadecimal. (For example, specify "D0.0B" for b11 in D0.)

Operation error

There is no operation error.

Pulse operation start, pulse series connection, pulse parallel connection

LDP, LDF, ANDP, ANDF, ORP, ORF

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- LDP: Rising edge pulse operation start

This instruction turns on only at the rising edge (off to on) of the specified bit device.

- LDF: Falling edge pulse operation start

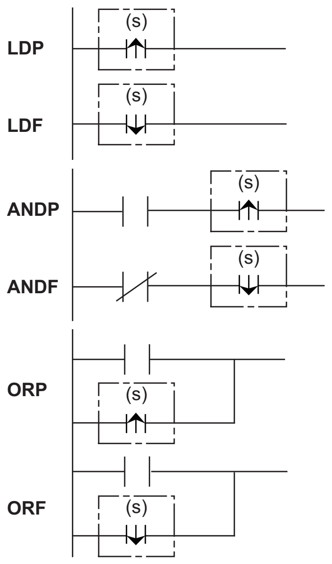
This instruction turns on only at the falling edge (on to off) of the specified bit device.

- ANDP: Rising edge pulse series connection, ANDF: Falling edge pulse series connection

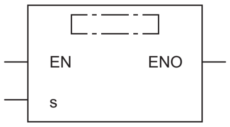
These instructions perform an AND operation with the previous operation result.

- ORP: Rising edge pulse parallel connection, ORF: Falling edge pulse parallel connection

These instructions perform an OR operation with the previous operation result.

Ladder	ST
	ENO:=LDP(EN,s); ENO:=LDF(EN,s); ENO:=ANDP(EN,s); ENO:=ANDF(EN,s); ENO:=ORP(EN,s); ENO:=ORF(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
LDP LDF ANDP ANDF ORP ORF	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	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)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

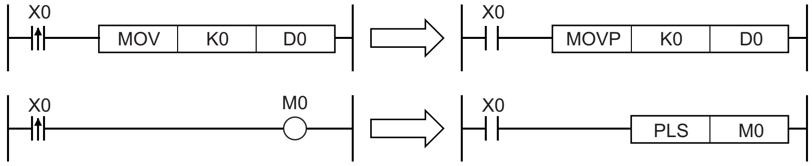
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

Processing details

■LDP, LDF

- LDP is a rising edge pulse operation start instruction, and turns on only at the rising edge (off to on) of the specified bit device. When a bit-specified word device is used, this instruction turns on only when the specified bit changes from 0 to 1. In cases where there is an LDP instruction only, it acts identically to instructions for conversion to pulses that are executed during on (□P).

The following figure shows an example when a ladder using the LDP instruction is replaced with a ladder not using the LDP instruction.



- LDF is a falling edge pulse operation start instruction and turns on only at the falling edge (on to off) of the specified bit device. When a bit-specified word device is used, this instruction turns on only when the specified bit changes from 1 to 0.
- If the LDP instruction is used in the program written in ST language or FBD/LD, ENO turns on at the rising edge (off to on) of the specified bit device (s).
- If the LDF instruction is used in the program written in ST language or FBD/LD, ENO turns on at the falling edge (on to off) of the specified bit device (s).
- If the LDP or LDF instruction is used in the program written in ST language, set EN to be always on.
- If the LDP or LDF instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.

■ANDP, ANDF

- ANDP is a rising edge pulse series connection instruction and ANDF is a falling edge pulse series connection instruction. These instructions perform an AND operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ANDP and ANDF instructions.

Device specified by ANDP or ANDF		ANDP status	ANDF status
Bit device	**Bit-specified word device**		
Off→On	0→1	On	Off
Off	0	Off	Off
On	1	Off	Off
On→Off	1→0	Off	On

- If the ANDP instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the rising edge of the specified bit device (s) is on. EN will not be an execution condition.
- If the ANDF instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the falling edge of the specified bit device (s) is on. EN will not be an execution condition.

■ORP, ORF

- ORP is a rising edge pulse parallel connection instruction and ORF is a falling edge pulse parallel connection instruction. These instructions perform an OR operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ORP and ORF instructions.

Device specified by ORP or ORF		ORP status	ORF status
Bit device	**Bit-specified word device**		
Off→On	0→1	On	Off
Off	0	Off	Off
On	1	Off	Off
On→Off	1→0	Off	On

- If the ORP instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the rising edge of the specified bit device (s) is on. EN will not be an execution condition.
- If the ORF instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the falling edge of the specified bit device (s) is on. EN will not be an execution condition.

Operation error

There is no operation error.

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

LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

- LDPI: Rising edge pulse NOT operation start

This instruction turns on when the specified device is off, on, or at the falling edge (on to off).

- LDFI: Falling edge pulse NOT operation start

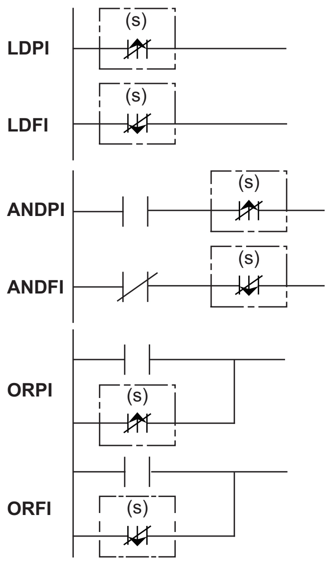
This instruction turns on when the specified bit device is at the rising edge (off to on), off, or on.

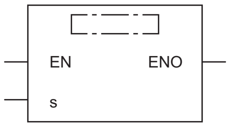
- ANDPI: Rising edge pulse NOT series connection, ANDFI: Falling edge pulse NOT series connection

These instructions perform an AND operation with the previous operation result.

- ORPI: Rising edge pulse NOT parallel connection, ORFI: Falling edge pulse NOT parallel connection

These instructions perform an OR operation with the previous operation result.

Ladder	ST
	ENO:=LDPI(EN,s); ENO:=LDFI(EN,s); ENO:=ANDPI(EN,s); ENO:=ANDFI(EN,s); ENO:=ORPI(EN,s); ENO:=ORFI(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
LDPI LDFI ANDPI ANDFI ORPI ORFI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	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)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

Processing details

■LDPI, LDFI

- LDFI is a rising edge pulse NOT operation start instruction, and turns on when the specified device is off, on, or at the falling edge (on to off). When a bit-specified word device is used, this instruction turns on when the specified bit is 0 or 1 or when the bit changes from 1 to 0.
- LDFI is a falling edge pulse NOT operation start instruction, and turns on when the specified bit device is at the rising edge (off to on), off, or on. When a bit-specified word device is used, this instruction turns on when the specified bit is 0 or 1 or when the bit changes from 0 to 1. The following table lists the on/off information used by the LDPI and LDFI instructions.

Device specified by LDPI or LDFI		LDPI status	LDFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the LDPI instruction is used in the program written in ST language or FBD/LD, ENO turns on at the timing except the rising edge (off to on) of the specified bit device (s).
- If the LDFI instruction is used in the program written in ST language or FBD/LD, ENO turns on at the timing except the falling edge (on to off) of the specified bit device (s).
- If the LDPI or LDFI instruction is used in the program written in ST language, set EN to be always on.
- If the LDPI or LDFI instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.

■ANDPI, ANDFI

- ANDPI is a rising edge pulse NOT series connection instruction and ANDFI is a falling edge pulse NOT series connection instruction. These instructions perform an AND operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ANDPI and ANDFI instructions.

Device specified by ANDPI or ANDFI		ANDPI status	ANDFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the ANDPI instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the rising edge of the specified bit device (s) is not on. EN will not be an execution condition.
- If the ANDFI instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the falling edge of the specified bit device (s) is not on. EN will not be an execution condition.

■ORPI, ORFI

- ORPI is a rising edge pulse NOT parallel connection instruction and ORFI is a falling edge pulse NOT parallel connection instruction. These instructions perform an OR operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ORPI and ORFI instructions.

Device specified by ORPI or ORFI		ORPI status	ORFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the ORPI instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the rising edge of the specified bit device (s) is not on. EN will not be an execution condition.
- If the ORFI instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the falling edge of the specified bit device (s) is not on. EN will not be an execution condition.

Operation error

There is no operation error.

5.2 Association Instructions

Ladder block series/parallel connection

ANB, ORB

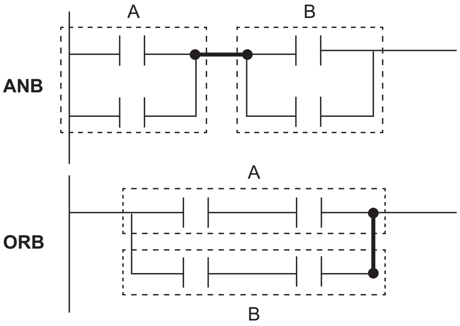
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- ANB: Ladder block series connection

This instruction performs an AND operation between block A and block B.

- ORB: Ladder block parallel connection

This instruction performs an OR operation between block A and block B.

<table>
<tr><th>Ladder</th><th>ST</th></tr>
<tr><td>


A: A block
B: B block</td><td>Not supported</td></tr>
</table>

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
ANB ORB	Every scan

Processing details

■ANB

- This instruction performs an AND operation between block A and block B, and outputs the operation result.
- The symbol of the ANB instruction is not a contact but a connection.

■ORB

- This instruction performs an OR operation between block A and block B, and outputs the operation result.
- Ladder blocks, each having two or more contacts, are connected in parallel. Use the OR or ORI instruction for connection of blocks, each having only one contact. The ORB instruction is not required in this case.
- The symbol of the ORB instruction is not a contact but a connection.

Operation error

There is no operation error.

Storing/reading/clearing the operation result

MPS, MRD, MPP

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- MPS: Storing the operation result

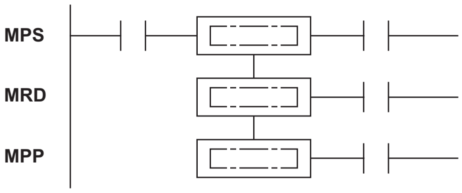
This instruction stores the operation result (on/off) immediately before the MPS instruction.

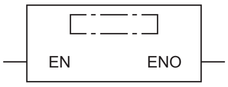
- MRD: Reading the operation result

This instruction reads the operation result stored by using the MPS instruction.

- MPP: Clearing the operation result

This instruction clears the operation result stored by using the MPS instruction.

Ladder	ST
 (In the ladder display, these instructions are hidden.)	ENO:=MPS(EN); ENO:=MRD(EN); ENO:=MPP(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
MPS MRD MPP	Every scan

Processing details

■MPS

- This instruction stores the operation result (on/off) immediately before the MPS instruction.
- Up to 16 MPS instructions can be used consecutively. If the MPP instruction is used in the middle of the program, the number of MPS instructions used is decremented by one.

■MRD

- This instruction reads the operation result stored by using the MPS instruction, and performs operations from the next step based on the operation result.

■MPP

- This instruction reads the operation result stored by using the MPS instruction, and performs operations from the next step based on the operation result.
- This instruction clears the operation result stored by using the MPS instruction.
- This instruction decrements the number of MPS instructions used in the program by one.

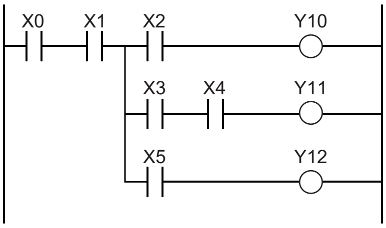
Operation error

There is no operation error.

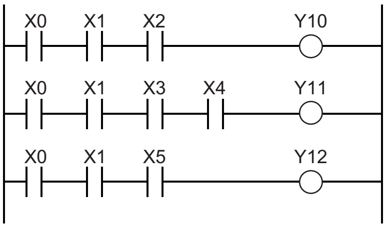
Point

- The following are the ladder program examples.

[Ladder program using the MPS, MRD, and MPP instructions]



[Ladder program not using the MPS, MRD, or MPP instruction]



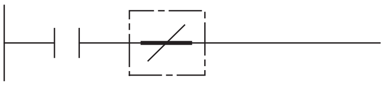
- Use the same number of MPS instructions as that of MPP instructions. If the numbers of MPS and MPP instructions are different, the ladder is not displayed correctly on the engineering tool (ladder mode).

Inverting the operation result

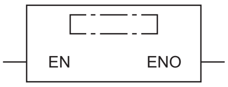
INV

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction inverts the operation result up to just before the INV instruction.

Ladder	ST
	ENO:=INV(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
INV	Every scan

Processing details

- This instruction inverts the operation result up to just before the INV instruction.

Operation result up to just before the INV instruction	Operation result after execution of the INV instruction
Off	On
On	Off

Operation error

There is no operation error.

Point

- The INV instruction operates based on the result of operations performed up to just before the INV instruction. Use it at the same position as the AND instruction. The instruction cannot be used at the LD or OR instruction position.
- When a ladder block is used, the operation result is inverted within the range of the ladder block. When the INV instruction and the ANB instruction are used together in the same ladder, pay attention to the inversion range.

Broken line part: Inversion range

For details on the ANB instruction, refer to the following.

☞ Page 168 ANB, ORB

Converting the operation result into a pulse

MEP, MEF

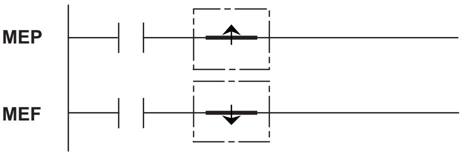
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

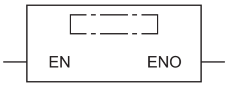
- MEP: Converting the operation result into a pulse (rising edge)

This instruction turns on at the rising edge (off to on) of the operation result up to the MEP instruction.

- MEP: Converting the operation result into a pulse (falling edge)

This instruction turns on at the falling edge (on to off) of the operation result up to the MEF instruction.

Ladder	ST
	ENO:=MEP(EN); ENO:=MEF(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
MEP MEF	Every scan

Processing details

■MEP

- This instruction turns on (continuity state) at the rising edge (off to on) of the operation result up to the MEP instruction. The instruction turns off (non-continuity state) when the operation result is in another state (not rising edge).
- Use of the MEP instruction eases pulse conversion processing when multiple contacts are connected in series.

■MEF

- This instruction turns on (continuity state) at the falling edge (on to off) of the operation result up to the MEF instruction. The instruction turns off (non-continuity state) when the operation result is in another state (not falling edge).
- Use of the MEF instruction eases pulse conversion processing when multiple contacts are connected in series.

Operation error

There is no operation error.

Point

- The MEP or MEF instruction may not operate correctly if pulse conversion is performed for an index-modified contact in the subroutine program or in the area between the FOR and NEXT instructions. To perform pulse conversion for an index-modified contact in the subroutine program or in the area between the FOR and NEXT instructions, refer to the following.

☞ Page 173 EGP, EGF

- The MEP and MEF instructions operate based on the result of operations performed from the LD instruction just before the MEP or MEF instruction to just before the MEP or MEF instruction. Use them at the same position as the AND instruction. The instructions cannot be used at the LD or OR instruction position.

Converting the edge relay operation result into a pulse

EGP, EGF

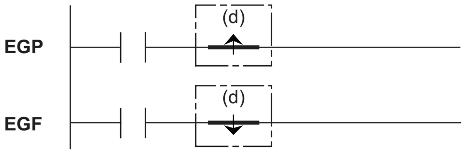
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- EGP: Converting the edge relay operation result into a pulse (rising edge)

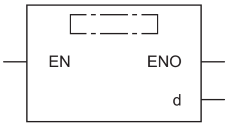
This instruction stores the operation result up to the EGP instruction in the edge relay (V). The instruction turns on at the rising edge (off to on) of the operation result.

- EGF: Converting the edge relay operation result into a pulse (falling edge)

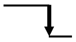
This instruction stores the operation result up to the EGF instruction in the edge relay (V). The instruction turns on at the falling edge (on to off) of the operation result.

Ladder	ST
	ENO:=EGP(EN,d); ENO:=EGF(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EGP	
EGF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Edge relay number for storing operation result	—	Bit	ANY_BOOL[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] Only bit type labels assigned to the device (V) can be used.

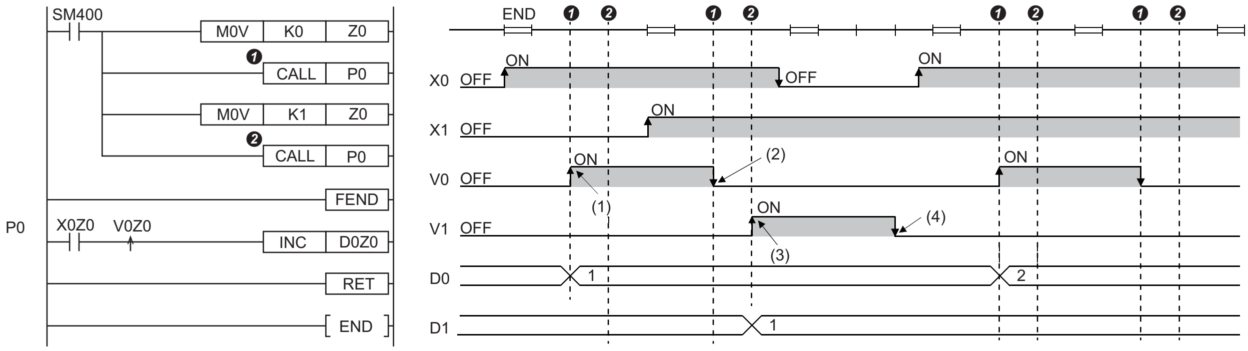
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (V)
	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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

■EGP

- This instruction stores the operation result up to the EGP instruction in the edge relay (V).
- The instruction turns on (continuity state) at the rising edge (off to on) of the operation result up to the EGP instruction. The instruction turns off (non-continuity state) when the operation result is in another state (staying on, falling edge (on to off), or staying off).
- The instruction is used to perform pulse conversion for index-modified programs in the subroutine program or in the area between the FOR and NEXT instructions.
- The instruction can be used in the same way as the AND instruction.
- The following figure shows the operation performed when the instruction is used in the subroutine program.



(1) The device turns on at the rising edge of X0.

(2) The device turns off since X0 remains on.

(3) The device turns on at the rising edge of X1.

(4) The device turns off since X1 remains on.

■EGF

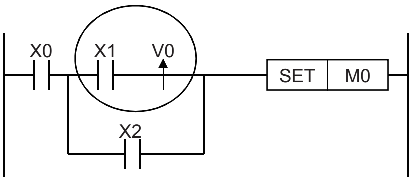
- This instruction stores the operation result up to the EGF instruction in the edge relay (V).
- The instruction turns on (continuity state) at the falling edge (on to off) of the operation result up to the EGF instruction. The instruction turns off (non-continuity state) when the operation result is in another state (staying on, rising edge (off to on), or staying off).
- The instruction is used to perform pulse conversion for index-modified programs in the subroutine program or in the area between the FOR and NEXT instructions.
- The instruction can be used in the same way as the AND instruction.

Operation error

There is no operation error.

Point

- The EGP and EGF instructions operate based on the result of operations performed from the LD instruction just before the EGP or EGF instruction to just before the EGP or EGF instruction. Use them at the same position as the AND instruction. The instructions cannot be used at the LD or OR instruction position.
- The instructions cannot be used at the ladder block position shown below.



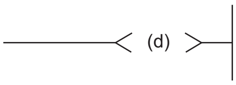
5.3 Output Instructions

Out (excluding the timer, counter, and annunciator)

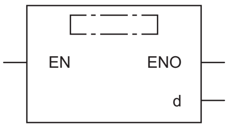
OUT

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction outputs the operation result to the specified device.

Ladder	ST
	ENO:=OUT(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
OUT	Every scan

Setting data

■Description, range, data type

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

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○*1	○	○*2	○	—	—	—	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 188 OUT F

*2 When T or ST is used, refer to the following.
☞ Page 177 OUT T, OUTH T, OUT ST, OUTH ST
When C is used, refer to the following.
☞ Page 184 OUT C

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○*3	—

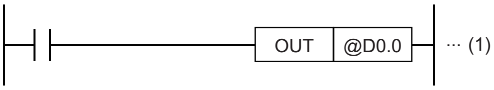
*3 When SA\T and SA\ST is used, refer to the following.
☞ Page 177 OUT T, OUTH T, OUT ST, OUTH ST
When SA\C is used, refer to the following.
☞ Page 184 OUT C

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

- When indirect specification is used, specify the bit as shown below.



(1) The operation result is output to bit 0 of the indirect address stored in D0.

Operation error

There is no operation error.

Timer

OUT T, OUTH T, OUT ST, OUTH ST

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

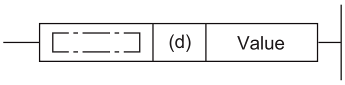
Point

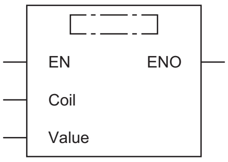
When the safety timer is used in safety programs executed by the Safety CPU, unless otherwise specified, replace some words as follows:

- "Timer" → "Safety timer"
- "Retentive timer" → "Safety retentive timer"
- "T" → "SA\T", "ST" → "SA\ST", "M0" → "SA\M0"
- "Scan" → "Safety cycle processing"
- "Scan time" → "Safety cycle time"

- 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).

Ladder	ST
 Value: Set value	ENO:=OUT_T(EN,Coil,Value); ENO:=OUTH(EN,Coil,Value);

FBD/LD
 Value: Set value (□ is to be replaced by either of the following: OUT_T, OUTH.)

■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	Data type (label)
(d) Coil	Timer device or timer type label	—	Bit	ANY_BOOL
Value	Value set for the timer	0 to 32767	16-bit unsigned binary*1	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d) Coil	—	—	○*1	—	—	—	—	—	—	—	—	—
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

*1 Only T and ST can be used.

*2 T, ST, and C cannot be used.

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

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d) Coil	—	○*4	—
Value	—	○*5	○*6

*4 Only SA\T and SA\ST can be used.

*5 SA\T, SA\ST, and SA\C cannot be used.

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

Processing details

- These instructions start time measurement, triggered by the coil specified by (d) (in SD language or FBD/LD, displayed as 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						

- To clear the current value of the retentive timer and turn off the contact after time is up, use the RST instruction.
- 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 OUT T instruction is skipped by using such as the JMP instruction while the OUT T, OUTH T, OUT ST, or OUTH ST instruction is on, the current value is not updated or the contact is not turned on or off.
- 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 1000ms (in increments of 1ms) (Default: 100ms)

High-speed timer/high-speed retentive timer: 0.01 to 100.0ms (in increments of 0.01ms) (Default: 10.0ms)

- For the counting method, refer to the following.

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

Precautions

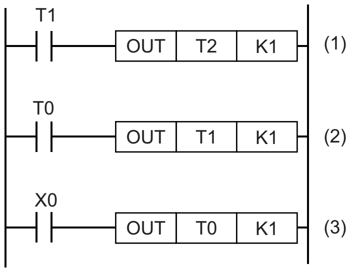
To create a program in which the operation of a timer contact triggers the operation of another timer, program the timers in order from the one that operates last.

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.

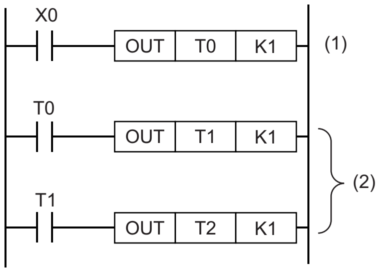
When timers T0 to T2 are programmed in order from the one that measures last



(1) Timer T2 starts measurement from the next scan after the contact of timer T1 turns on.
(2) Timer T1 starts measurement from the next scan after the contact of timer T0 turns on.
(3) Timer T0 starts measurement when X0 turns on.

Ex.

When timers T0 to T2 are programmed in order of measurement



(1) Timer T0 starts measurement when X0 turns on.

(2) When the contact of timer T0 turns on, the contacts of timers T1 and T2 also turn on.

Operation error

There is no operation error.

Long timer

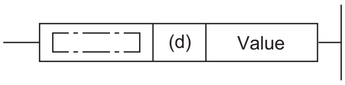
OUT LT, OUT LST

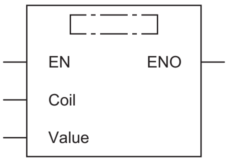
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **✕ RnSFCPU (Safety)**

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

- 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).

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

FBD/LD
 Value: Set value (□ is to be replaced by OUT_T.)

■Execution condition

Instruction	Execution condition
OUT LT OUT LST	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Long timer device or long timer type label	—	Bit	ANY_BOOL
Coil				
Value	Value set for the long timer	0 to 4294967295	32-bit unsigned binary[*1]	ANY32[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	—	—	—	○[*1]	—	—	—	—	—	—
Coil												
Value	—	—	○[*2]	○	—	—	—	—	○[*3]	—	—	—

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

Processing details

- These instructions start time measurement, triggered by the coil specified by (d) (in SD language or FBD/LD, displayed as 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

- To clear the current value of the long retentive timer and turn off the contact after time is up, use the RST instruction.
- 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 OUT LT instruction is skipped by using such as the JMP instruction while the OUT LT or OUT LST instruction is on, the current value is not updated or the contact is not turned on or off.
- 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: 0.001 to 1000ms (in increments of 0.001ms) (Default: 0.001ms)

- For the counting method, refer to the following.

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

Precautions

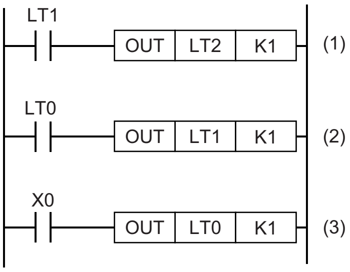
To create a program in which the operation of a long timer contact triggers the operation of another long timer, program the long timers in order from the one that operates last.

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.

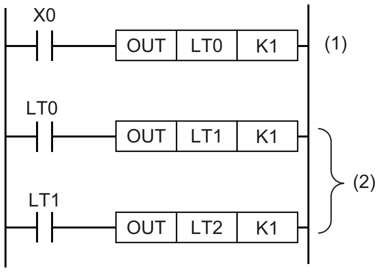
When timers LT0 to LT2 are programmed in order from the one that measures last



(1) Long timer LT2 starts measurement from the next scan after the contact of long timer LT1 turns on.
(2) Long timer LT1 starts measurement from the next scan after the contact of long timer LT0 turns on.
(3) Long timer LT0 starts measurement when X0 turns on.

Ex.

When long timers LT0 to LT2 are programmed in order of measurement



(1) Long timer LT0 starts measurement when X0 turns on.

(2) When the contact of timer LT0 turns on, the contacts of timers LT1 and LT2 also turn on.

Operation error

There is no operation error.

Counter

OUT C

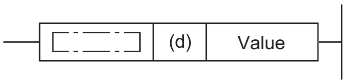
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

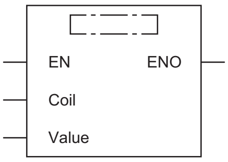
Point

When the safety counter is used in safety programs executed by the Safety CPU, unless otherwise specified, replace some words as follows:

- "OUT C" → "OUT SA\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).

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

FBD/LD
 Value: Set value

■Execution condition

Instruction	Execution condition
OUT C	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Counter number	—	Bit	ANY_BOOL*1
Coil				
Value	Value set for the counter	0 to 65535	16-bit unsigned binary*2	ANY16*2
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only counter type labels can be used.

*2 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○*1	—	—	—	—	—	—	—	—	—
Coil												
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

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

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	—	○*4	—
(Set value)	—	○*5	○*6

*4 Only SA\C can be used.
*5 SA\T, SA\ST, and SA\C cannot be used.
*6 Only K (decimal constant) can be used.

Processing details

- This instruction increments the current counter value (count value) in the device specified by (d) (in SD language or FBD/LD, displayed as 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

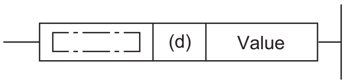
There is no operation error.

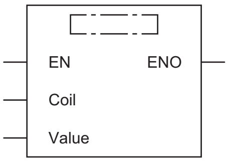
Long counter

OUT LC

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

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).

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

FBD/LD
 Value: Set value (□ is to be replaced by OUT_C.)

■Execution condition

Instruction	Execution condition
OUT LC	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Long counter number	—	Bit	ANY_BOOL[1]
Coil				
Value	Set value for the long counter	0 to 4294967295	32-bit unsigned binary[2]	ANY32[2]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Only long counter type labels can be used.

[2] If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	—	—	—	○[1]	—	—	—	—	—	—
Coil												
Value	—	—	○[2]	○	—	—	—	—	○[3]	—	—	—

[1] Only LC can be used.

[2] T, ST, and C cannot be used.

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

Processing details

- This instruction increments the current long counter value (count value) in the device specified by (d) (in SD language or FBD/LD, displayed as 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

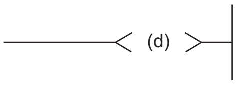
There is no operation error.

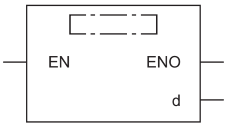
Annunciator

OUT F

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

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

Ladder	ST
	ENO:=OUT(EN,d);

FBD/LD


(□ is to be replaced by OUT.)

■Execution condition

Instruction	Execution condition
OUT F	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Target annunciator number	—	Bit	ANY_BOOL[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the annunciator can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[*1]	—	—	—	—	—	—	—	—	—	—	—

*1 Only F can be used.

Processing details

- This instruction outputs the operation result up to the OUT F instruction to the specified annunciator.
- When the annunciator (F) is turned on by this instruction, the following are performed.
  - The USER LED of the CPU module turns on.
  - The annunciator number (F number) to be turned on is stored in the special register (SD64 to SD79).
  - The value in SD63 is incremented by one.
- If the value in SD63 is 16 (meaning 16 annunciators are already on), the annunciator number will not be stored in the special register (SD64 to SD79) even when a new annunciator turns on.
- When the annunciator (F) is turned off by this instruction, the following are performed.
  - The coil turns off, but the USER LED status and the data in SD63 to SD79 remain unchanged.
  - To turn off the USER LED or delete the annunciator number that has been turned off by this instruction from SD63 to SD79, use the RST F instruction.

Operation error

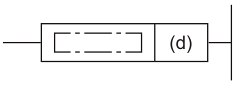
There is no operation error.

Setting devices (excluding annunciator)

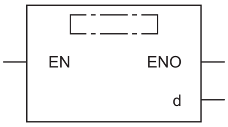
SET

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

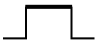
This instruction turns on the specified bit.

Ladder	ST
	ENO:=SET(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SET	

Setting data

■Description, range, data type

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

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○[*1]	○	—	—	—	—	—	○	—	—	—	○

[*1] When F is used, refer to the following.
☞ Page 193 SET F

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

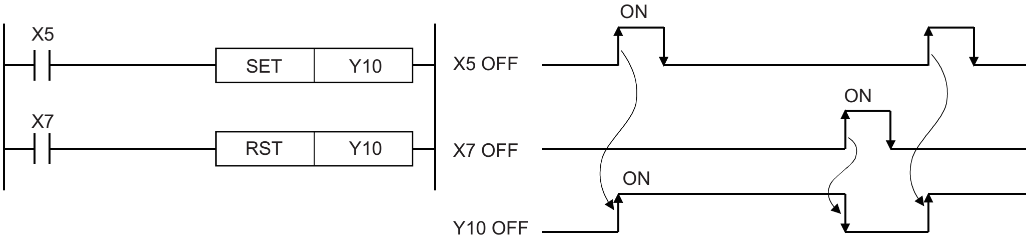
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	—	—

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.



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

Operation error

There is no operation error.

Point

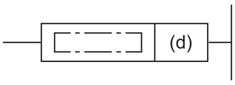
When X is used, specify a device number that is not used in actual input. If the number that is used in actual input is specified, the data of actual input is written over the input device (X) specified by the SET instruction.

Resetting devices (excluding annunciator)

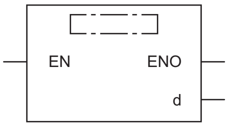
RST

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction turns off the specified device. For the timer and counter, the instruction clears the current value to 0 and turns off the contact or coil.

Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Reset target bit device number, bit specification of word device, or reset target word device number	—	Bit/Word/Double word	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○*1	○	○	○	○	○	○	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 195 RST F

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

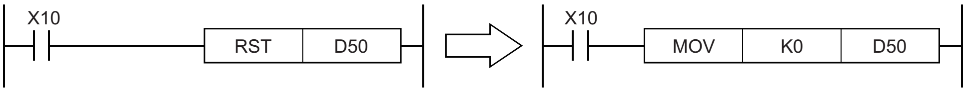
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

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.
Timer, counter	Clears the current value to 0 and turns off the coil or contact.
Bit-specified word device	Sets the specified bit to 0.
Word device other than timer and counter	Clears the data to 0.

- When the execution command is off, the device status does not change.
- The RST instruction specifying a word device operates in the same way as the following ladder.



Operation error

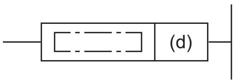
There is no operation error.

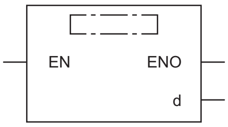
Setting annunciator

SET F

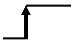
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction turns on the specified annunciator.

Ladder	ST
	ENO:=SET(EN,d);

FBD/LD

(□ is to be replaced by SET.)

■Execution condition

Instruction	Execution condition
SET F	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Set target annunciator number (F number)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

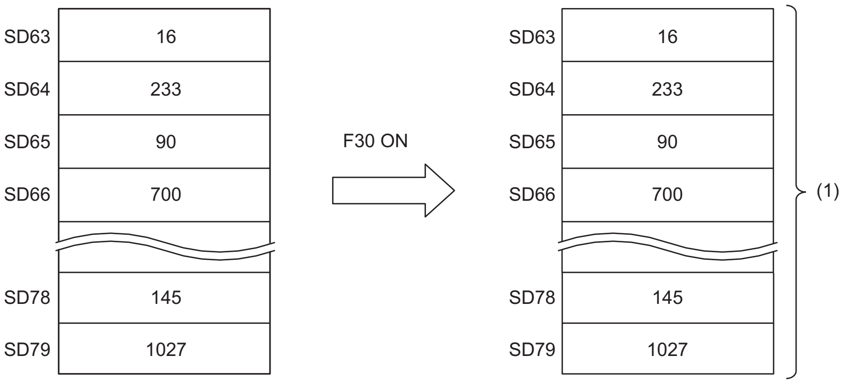
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[*1]	—	—	—	—	—	—	—	—	—	—	—

[*1] Only F can be used.

Processing details

- This instruction turns on the annunciator specified by (d) when the execution command turns on.
- When the annunciator (F) is turned on, the following are performed.
  - The USER LED turns on.
  - The annunciator number (F number) turned on is stored in the special register (SD64 to SD79).
  - The value in SD63 is incremented by one.
- If the value in SD63 is 16 (meaning 16 annunciators are already on), the annunciator number will not be stored in the special register (SD64 to SD79) even when a new annunciator turns on.



(1) The data remain the same.

Operation error

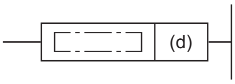
There is no operation error.

Resetting annunciator

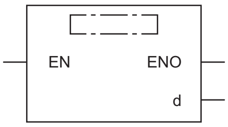
RST F

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction turns off the specified annunciator.

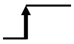
Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



(□ is to be replaced by RST.)

■Execution condition

Instruction	Execution condition
RST F	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Reset target annunciator number (F number)	—	Bit[1]	ANY_BOOL[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_ELEMENTARY.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[1]	—	—	—	—	—	—	—	—	—	—	—

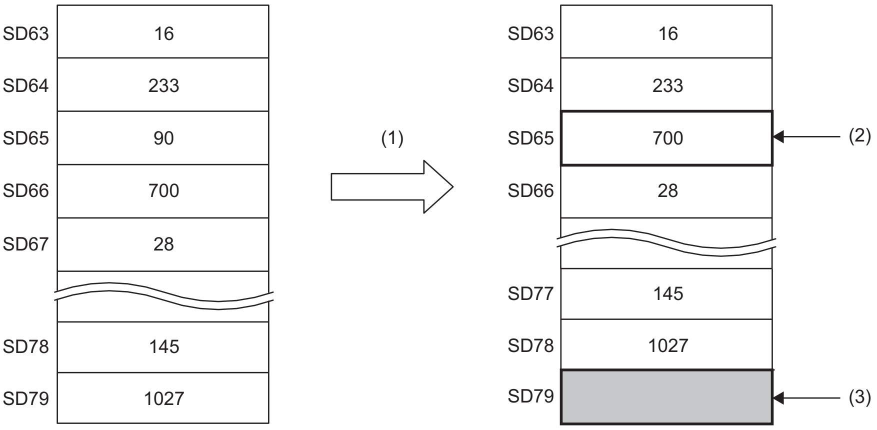
*1 Only F can be used.

Processing details

- This instruction turns off the annunciator specified by (d) when the execution command turns on.
- The annunciator number (F number) turned off is deleted from the special register (SD64 to SD79), and the value in SD63 is decremented by one.
- If the value in SD63 is 16, the corresponding annunciator number is deleted from SD64 to SD79 by the RST instruction. If an annunciator with a number not registered in SD64 to SD79 has been turned on, the number is newly registered. If all annunciator numbers in SD64 to SD79 are reset (turned off), the USER LED of the CPU module turns off.

Ex.

When the value in SD63 is 16 and there is an annunciator number that is not registered



(1) Reset F90.

(2) The F number in SD66 is shifted to this area.

(3) New F number is stored.

Operation error

There is no operation error.

Rising edge output

PLS

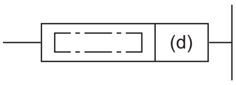
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

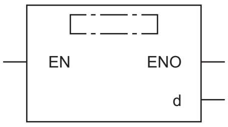
Point

When this instruction is used in safety programs executed by the Safety CPU, unless otherwise specified, replace some words as follows:

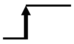
- "X0" → "SA\X0", "X5" → "SA\X5", "M0" → "SA\M0"
- "Scan" → "Safety cycle processing"

This instruction turns on the specified device for one scan on the rising edge (off to on) of the execution command.

Ladder	ST
	ENO:=PLS(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
PLS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

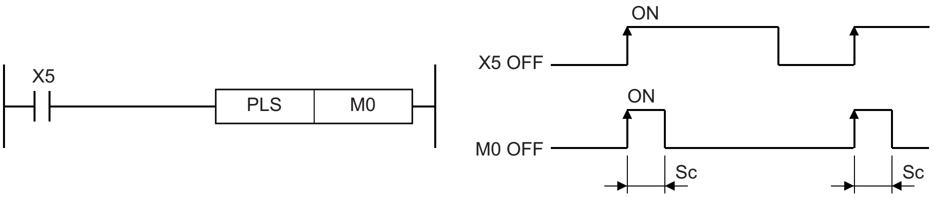
- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

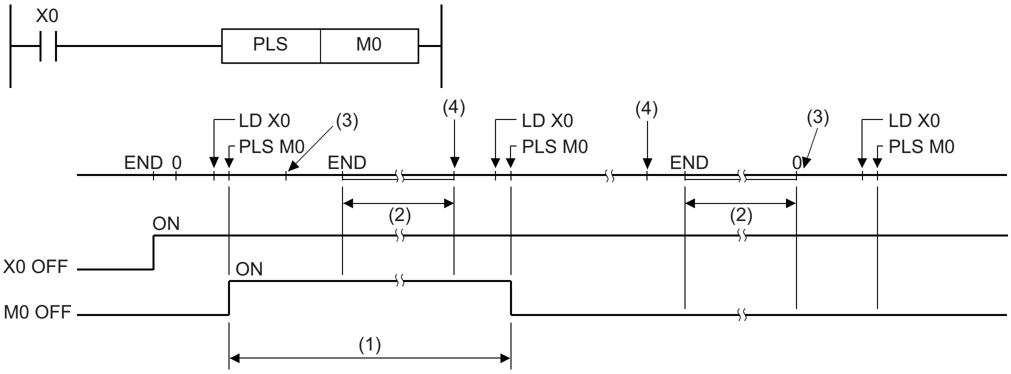
- This instruction turns on the specified device on the rising edge (off to on) of the execution command. When the execution command is in another state (staying on, falling edge (on to off), or staying off), the instruction turns off the specified device. If only one PLS instruction in the device specified by (d) is executed in a single scan, the specified device turns on for one scan. For the operation to be performed if more than one PLS instruction is executed during one scan, refer to the following.

☞ Page 59 Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used



Sc: 1 scan

- Once after execution of the PLS instruction, even if the switch of the CPU module is moved to the STOP position and then the RUN position again, the PLS instruction is not executed.



(1) M0 turns on for one scan.
(2) The CPU module operation stops.
(3) Change the RUN/STOP/RESET switch of the CPU module from RUN to STOP.
(4) Change the RUN/STOP/RESET switch of the CPU module from STOP to RUN.

- If the latch relay (L) is specified as the execution command and the system is powered on while the latch relay is on, the execution command turns on in the first scan, triggering execution of the PLS instruction and turning on the specified device. The device that has been turned on in the first scan after power-on can be turned off by the next PLS instruction.
- The PLS instruction performs OFF processing at the execution of the next instruction after the instruction execution. However, in safety programs executed by the Safety CPU, the PLS instruction turns on one safety cycle processing for the specified device/label until the safety program of next safety cycle processing starts and the instruction is executed. If the standard/safety shared label is used in the argument of the PLS instruction, the instruction may fail to detect the ON state of the standard/safety shared label or may detect the ON state for plural periods in the safety program or standard program that uses the corresponding standard/safety shared label depending on the timing to interrupt the safety cycle processing.

Point

- Note that if the PLS instruction is jumped by using the CJ instruction or the executed subroutine program is not called by using the CALL(P) instruction, the device specified by (d) may be on for more than one scan.

Operation error

There is no operation error.

Falling edge output

PLF

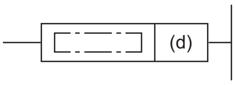
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

Point

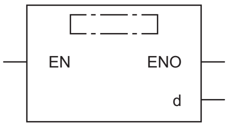
When this instruction is used in safety programs executed by the Safety CPU, unless otherwise specified, replace some words as follows:

- "X5" → "SA\X5"
- "Scan" → "Safety cycle processing"

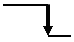
This instruction turns on the specified device for one scan on the falling edge (on to off) of the execution command.

Ladder	ST
 (d)	ENO:=PLF(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PLF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

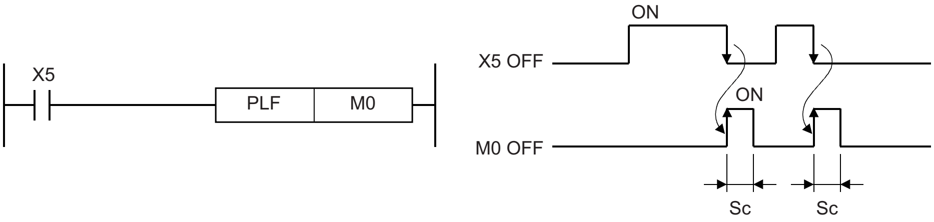
- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

- This instruction turns on the specified device on the falling edge (on to off) of the execution command. When the execution command is in another state (staying off, rising edge (off to on), or staying on), the instruction turns off the specified device. If only one PLF instruction in the device specified by (d) is executed during one scan, the specified device turns on for one scan. For the operation to be performed if more than one PLF instruction is executed during one scan, refer to the following.

☞ Page 59 Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used



Sc: 1 scan

- Once after execution of the PLF instruction, even if the switch of the CPU module is moved to the STOP position and then the RUN position again, the PLF instruction is not executed.
- The PLF instruction performs OFF processing at the execution of the next instruction after the instruction execution. However, in safety programs executed by the Safety CPU, the PLF instruction turns on one safety cycle processing for the specified device/label until the safety program of next safety cycle processing starts and the instruction is executed. If the standard/safety shared label is used in the argument of the PLF instruction, the instruction may fail to detect the ON state of the standard/safety shared label or may detect the ON state for plural periods in the safety program or standard program that uses the corresponding standard/safety shared label depending on the timing to interrupt the safety cycle processing.

Point

- Note that if the PLF instruction is jumped by using the CJ instruction or the executed subroutine program is not called by using the CALL(P) instruction, the device specified by (d) may be on for more than one scan.

Operation error

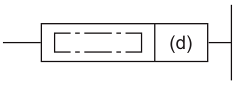
There is no operation error.

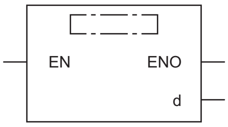
Inverting the bit device output

FF

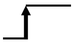
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction inverts the status of the specified device.

Ladder	ST
	ENO:=FF(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
FF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Inversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

- This instruction inverts the status of the device specified by (d) on the rising edge of the execution command.

Device	Device status	
	Before execution of the FF instruction	After execution of the FF instruction
Bit device	Off	On
	On	Off
Bit-specified word device	0	1
	1	0

Operation error

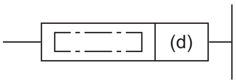
There is no operation error.

Converting the direct access output into a pulse

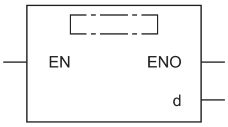
DELTA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

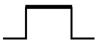
These instructions convert the specified direct access output (DY) into pulse output.

Ladder	ST
	ENO:=DELTA(EN,d); ENO:=DELTAP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DELTA	
DELTAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the device (DY) can be used.

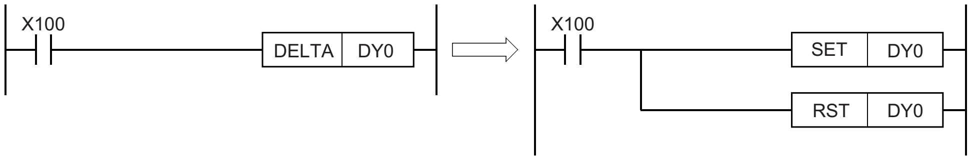
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○

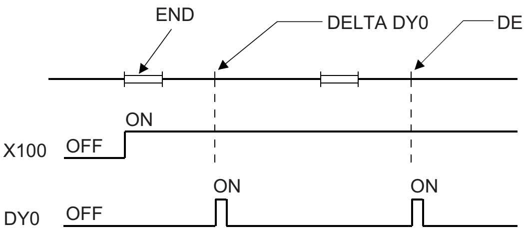
Processing details

- These instructions convert the direct access output (DY) specified by (d) into pulse output. If DY0 is specified by (d), the program operates in the same way as the one that uses the SET and RST instructions.

The following figure shows an example when a ladder using the DELTA instruction is replaced with a ladder using the SET/RST instructions.



The following figure shows the operation of the instruction.



Precautions

These instructions are used as an execution command (rising edge execution) for intelligent function modules.
These instructions cannot be used as an actual output command for output modules.

Operation error

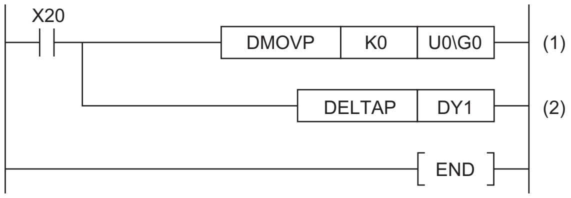
There is no operation error.

Point

The DELTA(P) instruction is used to set a preset value of the high-speed counter module.

[Example]

A program that presets the CH1 of the high-speed counter module (RD62P2) mounted in slot 0 of the base unit when X20 turns on



(1) Store the preset value (0) in the buffer memory areas 0 and 1 of the RD62P2.

(2) Output the preset command.

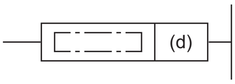
5.4 Shift Instructions

Shifting bit devices

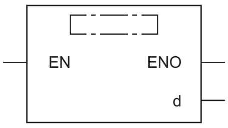
SFT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

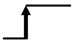
These instructions shift the on/off state of the device area just before the one specified to the specified device area, and turn off the shift source device.

Ladder	ST
	ENO:=SFT(EN,d); ENO:=SFTP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFT	
SFTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	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	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

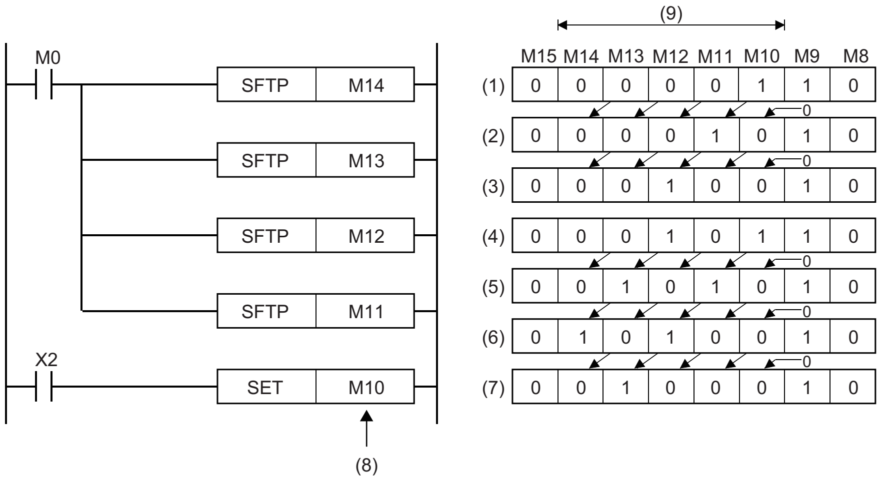
■Bit device

- These instructions shift the on/off state of the device area just before the one specified by (d) to the device area specified by (d). After the data is shifted, the data of the shift source device area is turned off.

Ex.

When the SFTP instruction that specifies M11 is executed, it shifts the on/off state of M10 to M11, and turns off M10.

- Turn on the shift target start device by using the SET instruction.
- When the SFT(P) instruction is used consecutively, program devices in descending order of the device numbers.



(1) X02 ON

(2) After the 1st shift input

(3) After the 2nd shift input

(4) X02 ON

(5) After the 3rd shift input

(6) After the 4th shift input

(7) After the 5th shift input

(8) Shift target start device

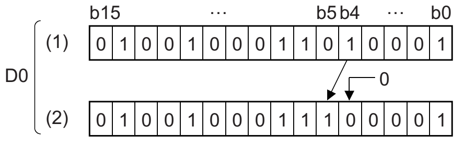
(9) Shift range

■Bit-specified word device

- These instructions shift the 1/0 state of the bit just before the one specified by (d) to the bit specified by (d). After the data is shifted, the data of the shift source bit is set to 0.

Ex.

The SFT(P) instruction that specifies D0.5 (b5 in D0) is executed, it shifts the 1/0 state of b4 in D0 to b5, and sets b4 to 0.



(1) Before shifting the bit

(2) After shifting the bit

Operation error

There is no operation error.

5.5 Master Control Instructions

Setting/resetting a master control

MC, MCR

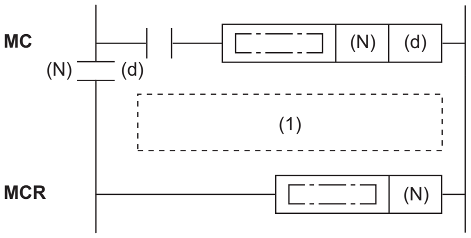
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

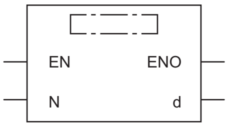
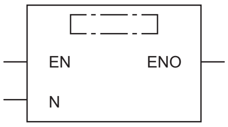
Point

When this instruction is used in safety programs executed by the Safety CPU, unless otherwise specified, replace some words as follows:

- "Timer" → "Safety timer"
- "Retentive timer" → "Safety retentive timer"
- "Counter" → "Safety counter"
- Add "SA\" to the devices "X" and "M" in Figures. (Example: "X0" → "SA\X0", "M0" → "SA\M0")
- "Scan time" → "Safety cycle time"

- MC: This instruction starts a master control.
- MCR: This instruction ends a master control.

Ladder	ST
 (1) Master control ladder	ENO:=MC(EN,N,d); ENO:=MCR(EN,N);

FBD/LD	
MC 	MCR 

■Execution condition

Instruction	Execution condition
MC MCR	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(N)	Nesting	N0 to N14	Device name	ANY16_S[*1]
(d)	Number of the device to be turned on	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] Only labels assigned to the device (N) or to which constants are assigned can be used.

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others	
	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	$	N	DY
(N)	—	—	—	—	—	—	—	—	—	—	—	○	—
(d)	○	○	○	—	—	—	—	○	—	—	—	—	○

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

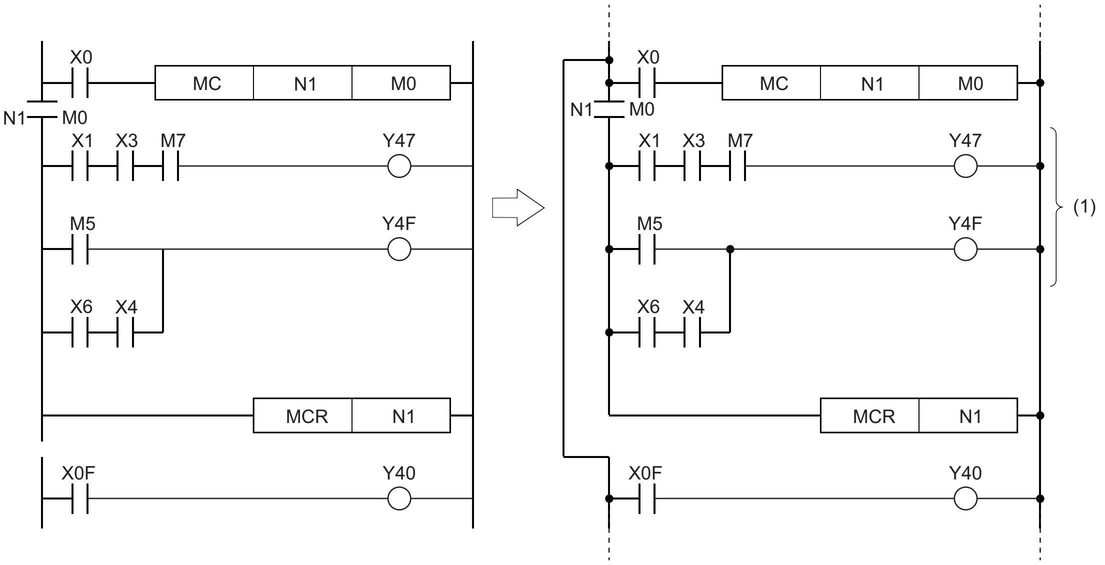
Operand	Bit	Word	Indirect specification	Constant	Others
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD		K, H	N
(N)	—	—	—	—	○
(d)	○	○	○	—	—

Processing details

These instructions are used to create an efficient ladder switching program by opening and closing the common rails of the ladder.

The following is the program example using the master control instructions.

(Left: Display on the engineering tool, Right: Actual operation)



(1) Executed only when X0 is on.

■MC

- If the execution command of the MC instruction is on when a master control starts, the operation result between the MC and MCR instructions will be the one as programmed. If the execution command is off, the operation result between the MC and MCR instructions will be as follows.

Device	Status
High-speed timer Low-speed timer	The count value is set to 0, and both the coil and contact are turned off.
High-speed retentive timer Low-speed retentive timer Counter	The coil is turned off, but both the count value and contact maintain the current status.
Device used by the OUT instruction	Forcibly turned off.
Device used by the SET and RST instructions Device used by the SFT(P) instruction Device used by basic instructions and application instructions	Maintains the current status.

- Even if the MC instruction is off, the instructions between the MC and MCR instructions are executed and therefore the scan time is not shortened.

Point

- When a ladder performing a master control includes an instruction which does not require a contact instruction (such as the FOR to NEXT instruction), the CPU module executes the instruction regardless of the execution command of the MC instruction.
- To create an easy-to-understand program, use the MC and MCR instructions within a single program block.

- The MC instruction can use the same nesting (N) number as many times as needed by specifying different devices in (d).
- When the MC instruction is on, the coil of the device specified by (d) turns on. Using the same device for the OUT instruction causes double coils. Do not use the device specified by (d) in other instructions.

■MCR

- This instruction is a master control reset instruction which indicates the end of the master control area.
- Do not place any contact instruction before the MCR instruction.
- Use the MC and MCR instructions with the same nesting number as a set. Note that if the MCR instructions are nested in one place, all master controls can be terminated by specifying the lowest nesting (N) number. (Refer to "Precautions".)

Operation error

There is no operation error.

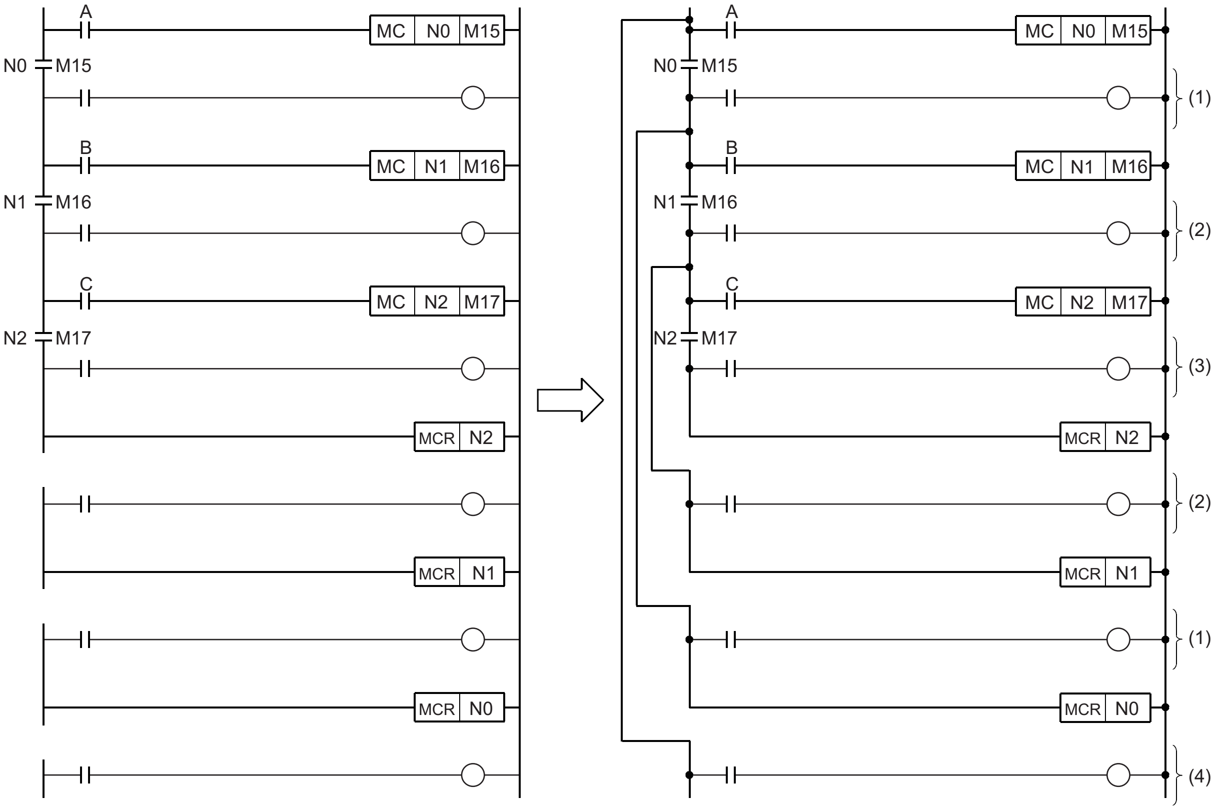
Point

The master control instructions can be nested. Individual master control areas are distinguished by nesting (N) numbers. Nesting can be set from N0 to N14.

Using the nesting structure enables the creation of a ladder which can sequentially constrain the program execution conditions.

The following figure shows a ladder program example using the nesting structure.

(Left: Display on the engineering tool, Right: Actual operation)



(1) Executed when A is on.

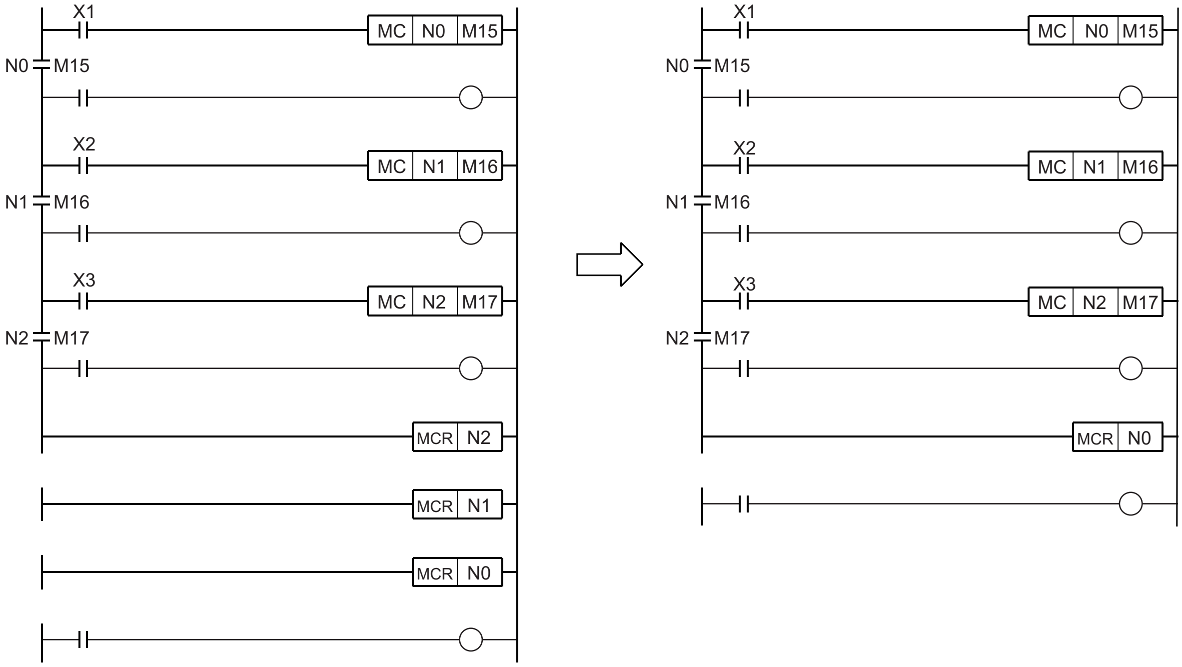
(2) Executed when A and B are on.

(3) Executed when A, B, and C are on.

(4) Executed regardless of the status of A, B, and C

Precautions

- Up to 15 nests (N0 to N14) are allowed. When nesting is performed, the MC instruction should use nesting (N) numbers in order from lower numbers and the MCR instruction should use them in order from higher numbers.
- If the MCR instructions are nested in one place, all master controls can be terminated by specifying the lowest nesting (N) number.



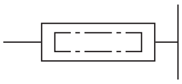
5.6 Termination Instructions

Ending the main routine program

FEND

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction is used to separate the main routine program from subroutine programs and interrupt programs in a program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

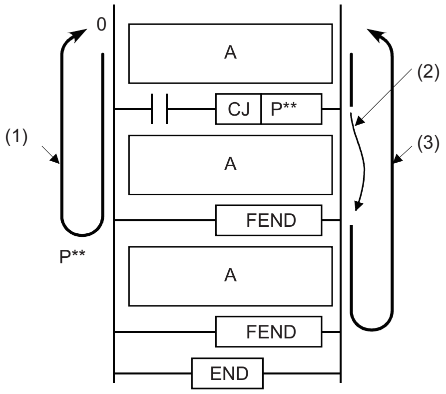
■Execution condition

Instruction	Execution condition
FEND	Every scan

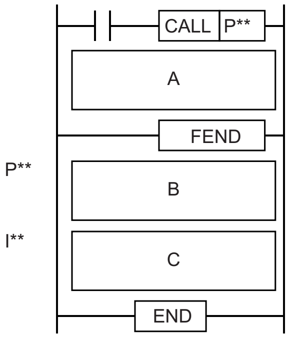
Processing details

- This instruction is used to divide sequence program operations by using a program branch instruction such as the CJ instruction or to separate the main routine program from subroutine programs and interrupt programs specified by the interrupt pointer (I).
- When the instruction is executed, the CPU module terminates the running program.
- Sequence programs following the FEND instruction can be displayed on the engineering tool (ladder mode).

When the CJ instruction is used



When there are subroutine and interrupt programs



A: Main routine program

B: Subroutine program

C: Interrupt program

(1) Operation performed when the CJ instruction is not executed

(2) Jump caused by the CJ instruction

(3) Operation performed when the CJ instruction is executed

Operation error

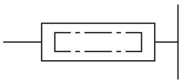
Error code (SD0)	Description
3340H	After execution of the FOR instruction, the FEND instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the FEND instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the FEND instruction is executed before the IRET instruction.

Ending the sequence program

END

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction indicates the end of a program.

Ladder	ST
	Not supported

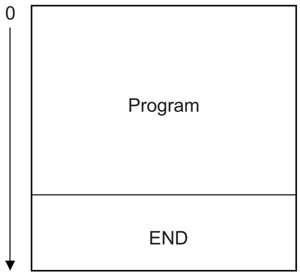
FBD/LD
Not supported

■Execution condition

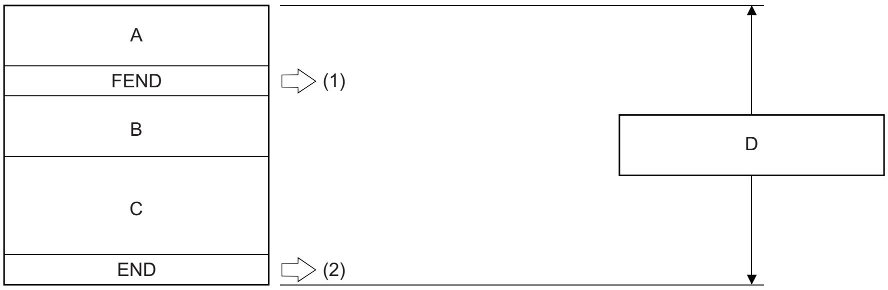
Instruction	Execution condition
END	Every scan

Processing details

- This instruction indicates the end of a program including a main routine program, subroutine programs, and interrupt programs.
- When the instruction is executed, the CPU module terminates the running program.



- If END processing is required in the middle of a program, use the FEND instruction.
- If the program is created using the engineering tool (in ladder edit mode), the END instruction is automatically input and cannot be edited.
- The following figure shows how to use the termination instructions when a main routine program, subroutine program, and interrupt program exist.



A: Main routine program

B: Subroutine program

C: Interrupt program

D: Main sequence program area

(1) The FEND instruction is required.

(2) The END instruction is required.

Point

When a program is divided into multiple program blocks, the END instruction indicates the end of a program block.

The END instruction within the program registered at the end of the program setting performs END processing.

Operation error

Error code (SD0)	Description
3340H	After execution of the FOR instruction, the END instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the END instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the END instruction is executed before the IRET instruction.

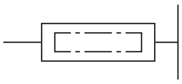
5.7 Stop Instruction

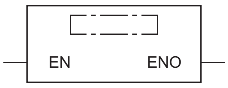
Stopping the sequence program

STOP

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

This instruction stops the operation of the CPU module. (The operation of this instruction is the same as setting the switch of the CPU module to the STOP position.)

Ladder	ST
	ENO:=STOP(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
STOP	

Processing details

- This instruction resets the output (Y) and stops the operation of the CPU module when the execution command turns on. (The operation of this instruction is the same as setting the switch of the CPU module to the STOP position.)
- To restart the operation of the CPU module after execution of the STOP instruction, set the switch back to STOP, and then set it to RUN again.

Operation error

Error code (SD0)	Description
3340H	After execution of the FOR instruction, the STOP instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction, the STOP instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the STOP instruction is executed before the IRET instruction.
33A3H	The STOP instruction is executed within a fixed scan execution type program.
33A4H	The STOP instruction is executed within an event execution type program.

5.8 No Operation Instruction

No operation (NOP)

NOP

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction is used to insert a space for debugging.

Ladder	ST
—	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
NOP	Every scan

Processing details

- This instruction is a no-operation instruction and has no impact on the previous operations.
- The instruction is used for the following purposes:
  - To insert a space for debugging
  - To delete an instruction without changing the number of steps (The relevant instruction is replaced with the NOP instruction.)
  - To delete an instruction temporarily

Point

For inserting or deleting the NOP instruction, refer to the following.

📖 GX Works3 Operating Manual

Operation error

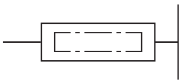
There is no operation error.

No operation (NOPLF)

NOPLF

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction is a no-operation instruction and has no impact on the previous operations.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
NOPLF	Every scan

Processing details

- This instruction is a no-operation instruction and has no impact on the previous operations.

Operation error

There is no operation error.

6 BASIC INSTRUCTIONS

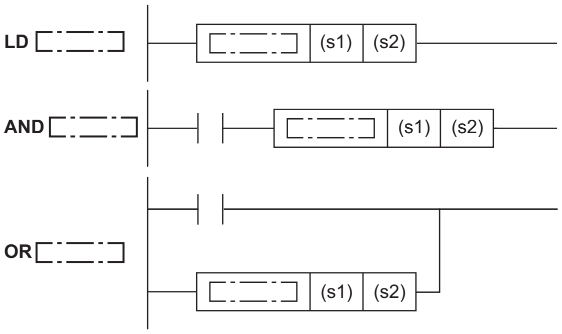
6.1 Comparison Operation Instructions

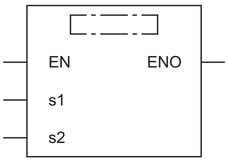
Comparing 16-bit binary data

LD□(_U), AND□(_U), OR□(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions compare the two sets of 16-bit binary data specified. (Devices are used as normally open contacts.)

Ladder	ST[1]	
	ENO:=LD_□(EN,s1,s2); ENO:=AND_□(EN,s1,s2); ENO:=OR_□(EN,s1,s2);	ENO:=LD_□_U(EN,s1,s2); ENO:=AND_□_U(EN,s1,s2); ENO:=OR_□_U(EN,s1,s2);
(□ is to be replaced by any of the following: =(_U), <>(_U), >(_U), <=(_U), <(_U), >=(_U).)	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[2]	

FBD/LD

(□ is to be replaced by combination of any of the following: LD_, AND_, OR_ and EQ(_U), NE(_U), GT(_U), LE(_U), LT(_U), GE(_U).)[2]

[1] The engineering tool with version "1.035M" or later supports the ST.

[2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LD□(_U), AND□(_U), OR□(_U)	Every scan

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	LD□, AND□, OR□	Comparison data or the device where comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	LD□_U, AND□_U, OR□_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	LD□, AND□, OR□	Comparison data or the device where comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	LD□_U, AND□_U, OR□_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○

Processing details

- These instructions perform a comparison operation between the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2). (Devices are used as normally open contacts.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
=(_U), EQ(_U)	(s1)=(s2)	Continuity state (ENO is on.)
<>(_U), NE(_U)	(s1)≠(s2)	
>(_U), GT(_U)	(s1)>(s2)	
<=(_U), LE(_U)	(s1)≤(s2)	
<(_U), LT(_U)	(s1)<(s2)	
>=(_U), GE(_U)	(s1)≥(s2)	
=(_U), EQ(_U)	(s1)≠(s2)	Non-continuity state (ENO is off.)
<>(_U), NE(_U)	(s1)=(s2)	
>(_U), GT(_U)	(s1)≤(s2)	
<=(_U), LE(_U)	(s1)>(s2)	
<(_U), LT(_U)	(s1)≥(s2)	
>=(_U), GE(_U)	(s1)<(s2)	

- When hexadecimal constants are used for (s1) and (s2) and the numerical value (8 to F) whose most significant bit (b15) is 1 is specified as a constant, the value is considered as a negative binary value in comparison operation.
- If the LD□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the OR□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

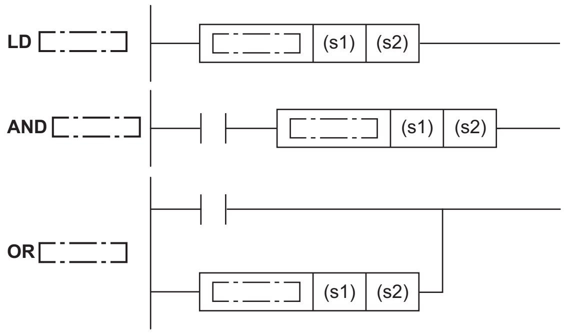
There is no operation error.

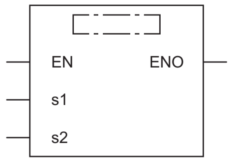
Comparing 32-bit binary data

LDD□(_U), ANDD□(_U), ORD□(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions compare the two sets of 32-bit binary data specified. (Devices are used as normally open contacts.)

Ladder	ST[*1]	
 (□ is to be replaced by any of the following: D=(_U), D<>(_U), D>(_U), D<=(_U), D<(_U), D>=(_U).)	ENO:=LDD_□(EN,s1,s2); ENO:=ANDD_□(EN,s1,s2); ENO:=ORD_□(EN,s1,s2);	ENO:=LDD_□_U(EN,s1,s2); ENO:=ANDD_□_U(EN,s1,s2); ENO:=ORD_□_U(EN,s1,s2);
	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[*2]	

FBD/LD
 (□ is to be replaced by combination of any of the following: LDD_, ANDD_, ORD_ and EQ(_U), NE(_U), GT(_U), LE(_U), LT(_U), GE(_U).)[*2]

[*1] The engineering tool with version "1.035M" or later supports the ST.

[*2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDD□(_U), ANDD□(_U), ORD□(_U)	Every scan

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	LDD□, ANDD□, ORD□	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	LDD□_U, ANDD□_U, ORD□_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	LDD□, ANDD□, ORD□	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	LDD□_U, ANDD□_U, ORD□_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○

Processing details

- These instructions perform a comparison operation between the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2). (Devices are used as normally open contacts.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
D=(_U), EQ(_U)	(s1)=(s2)	Continuity state (ENO is on.)
D<>(_U), NE(_U)	(s1)≠(s2)	
D>(_U), GT(_U)	(s1)>(s2)	
D<=(_U), LE(_U)	(s1)≤(s2)	
D<(_U), LT(_U)	(s1)<(s2)	
D>=(_U), GE(_U)	(s1)≥(s2)	
D=(_U), EQ(_U)	(s1)≠(s2)	Non-continuity state (ENO is off.)
D<>(_U), NE(_U)	(s1)=(s2)	
D>(_U), GT(_U)	(s1)≤(s2)	
D<=(_U), LE(_U)	(s1)>(s2)	
D<(_U), LT(_U)	(s1)≥(s2)	
D>=(_U), GE(_U)	(s1)<(s2)	

- When hexadecimal constants are specified for (s1) and (s2) and the numerical value (8 to F) whose most significant bit (b31) is 1 is specified as a constant, the value is considered as a negative binary value in comparison operation.
- To specify the compare target data, use an instruction which handles 32-bit data, such as the DMOV(P) instruction. If an instruction which handles 16-bit data, such as the MOV(P) instruction, is used, comparison cannot be performed normally.
- If the LDD□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORD□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

There is no operation error.

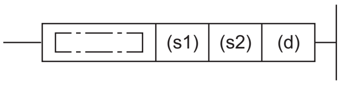
Outputting a comparison result of 16-bit binary data

CMP(P)(_U)

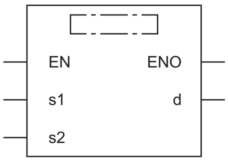
RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

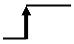
These instructions compare the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=CMP(EN,s1,s2,d); ENO:=CMPP(EN,s1,s2,d);	ENO:=CMP_U(EN,s1,s2,d); ENO:=CMPP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CMP CMP_U	
CMPP CMPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	CMP(P)	Comparison data or the device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	CMP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	CMP(P)	Comparison data or the device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	CMP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

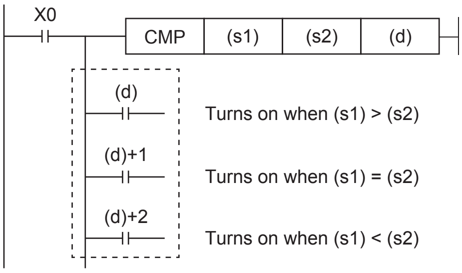
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

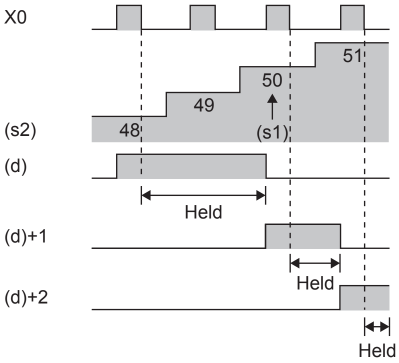
*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.





Operation error

There is no operation error.

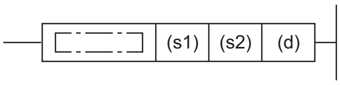
Outputting a comparison result of 32-bit binary data

DCMP(P)(_U)

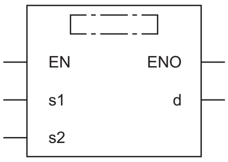
RnCPU RnENCPU ✕RnPCPU (Process) ✕RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

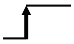
These instructions compare the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=DCMP(EN,s1,s2,d); ENO:=DCMPP(EN,s1,s2,d);	ENO:=DCMP_U(EN,s1,s2,d); ENO:=DCMPP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DCMP DCMP_U	
DCMPP DCMPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DCMP(P)	Comparison data or the device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DCMP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DCMP(P)	Comparison data or the device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DCMP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)		The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

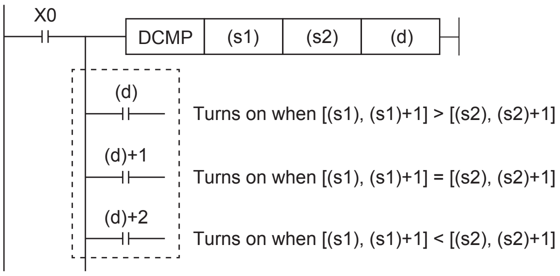
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

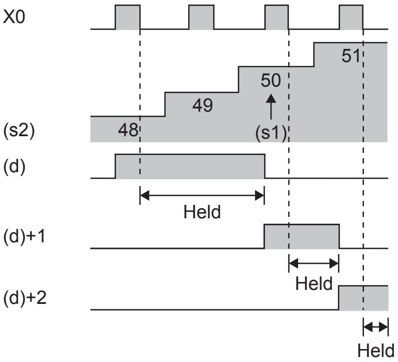
*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.





Operation error

There is no operation error.

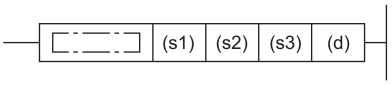
Outputting a band comparison result of 16-bit binary data

ZCP(P)(_U)

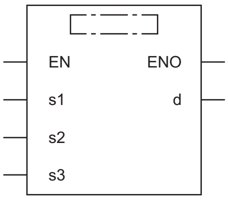
RnCPU **RnENCPU** ✕ **RnPCPU (Process)** ✕ **RnPCPU (Redundant)** ✕ **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

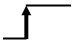
These instructions compare the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=ZCP(EN,s1,s2,s3,d); ENO:=ZCPP(EN,s1,s2,s3,d);	ENO:=ZCP_U(EN,s1,s2,s3,d); ENO:=ZCPP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZCP ZCP_U	
ZCPP ZCPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	ZCP(P)	Lower limit value or the start device for storing the lower limit value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	ZCP(P)	Upper limit value or the start device for storing the upper limit value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	ZCP(P)	Comparison data or the start device for storing the comparison data	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

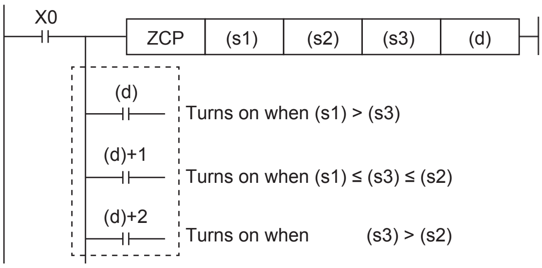
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s3)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

There is no operation error.

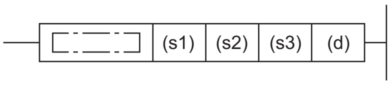
Outputting a band comparison result of 32-bit binary data

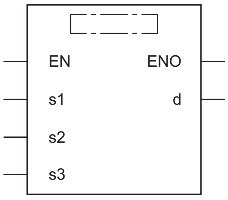
DZCP(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)** ✗

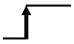
- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

These instructions compare the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
 (s1) (s2) (s3) (d)	ENO:=DZCP(EN,s1,s2,s3,d); ENO:=DZCPP(EN,s1,s2,s3,d);	ENO:=DZCP_U(EN,s1,s2,s3,d); ENO:=DZCPP_U(EN,s1,s2,s3,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DZCP DZCP_U	
DZCPP DZCPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DZCP(P)	Lower limit value or the start device for storing the lower limit value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DZCP(P)	Upper limit value or the start device for storing the upper limit value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DZCP(P)	Comparison data or the start device for storing the comparison data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)		The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

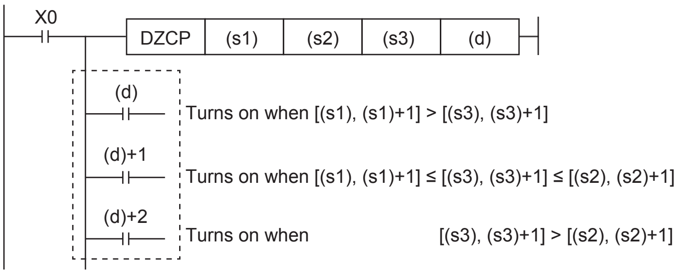
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s3)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

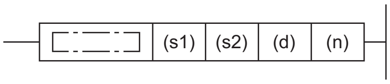
There is no operation error.

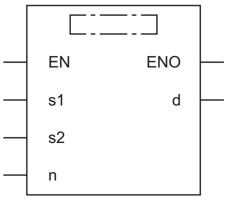
Comparing 16-bit binary block data

BKCMP□(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

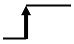
Ladder	ST[1]	
 (□ is replaced by any of the following: BKCMP=(P)(_U), BKCMP<>(P)(_U), BKCMP>(P)(_U), BKCMP<=(P)(_U), BKCMP<(P)(_U), BKCMP>=(P)(_U).)	ENO:=BKCMP_□(EN,s1,s2,n,d); ENO:=BKCMP_□P(EN,s1,s2,n,d);	ENO:=BKCMP_□_U(EN,s1,s2,n,d); ENO:=BKCMP_□P_U(EN,s1,s2,n,d) ;
	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[2]	

FBD/LD
 (□ is to be replaced by combination of any of the following: BKCMP_ and EQ(P)(_U), NE(P)(_U), GT(P)(_U), LE(P)(_U), LT(P)(_U), GE(P)(_U).)[2]

[1] The engineering tool with version "1.035M" or later supports the ST.

[2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
BKCMP□(_U)	
BKCMP□P(_U)	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BKCMP□(P)	Comparison data or the start device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S[1]
	BKCMP□(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U[1]
(s2)	BKCMP□(P)	Start device where the comparison data is stored	—	16-bit signed binary	ANY16_S[1]
	BKCMP□(P)_U			16-bit unsigned binary	ANY16_U[1]
(d)		Start device for storing the comparison operation result	—	Bit	ANY_BOOL[1]
(n)		Number of data points to be compared	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

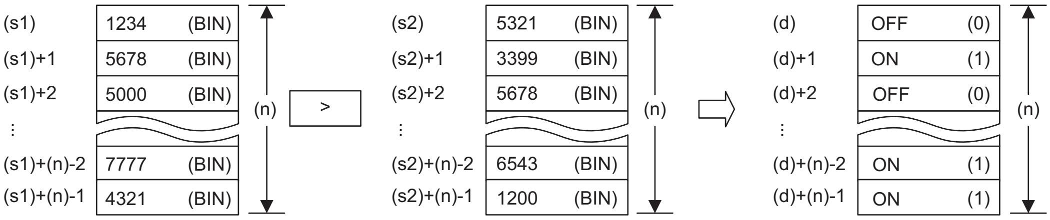
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

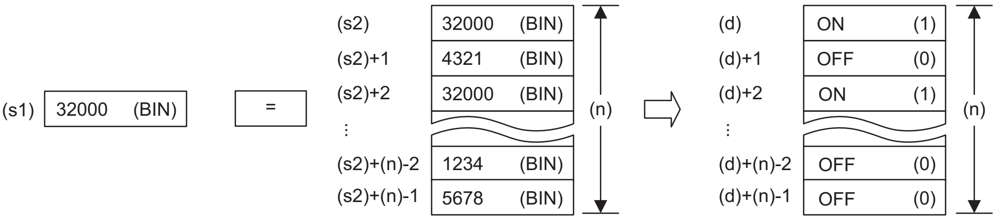
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the (n) points of 16-bit binary data from the device specified by (s1) with the (n) points of 16-bit binary data from the device specified by (s2), and stores the operation result in the device specified by (d) and later.
- If the comparison condition is satisfied, the relevant device specified by (d) turns on; otherwise, the device turns off.



- Specify data in units of 16 bits.
- A constant can be specified for (s1).



- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
BKCMP=(P)(_U), BKCMP_EQ(P)(_U)	(s1)=(s2)	On (1)
BKCMP<>(P)(_U), BKCMP_NE(P)(_U)	(s1)≠(s2)	
BKCMP>(P)(_U), BKCMP_GT(P)(_U)	(s1)>(s2)	
BKCMP<=(P)(_U), BKCMP_LE(P)(_U)	(s1)≤(s2)	
BKCMP<(P)(_U), BKCMP_LT(P)(_U)	(s1)<(s2)	
BKCMP>=(P)(_U), BKCMP_GE(P)(_U)	(s1)≥(s2)	
BKCMP=(P)(_U), BKCMP_EQ(P)(_U)	(s1)≠(s2)	Off (0)
BKCMP<>(P)(_U), BKCMP_NE(P)(_U)	(s1)=(s2)	
BKCMP>(P)(_U), BKCMP_GT(P)(_U)	(s1)≤(s2)	
BKCMP<=(P)(_U), BKCMP_LE(P)(_U)	(s1)>(s2)	
BKCMP<(P)(_U), BKCMP_LT(P)(_U)	(s1)≥(s2)	
BKCMP>=(P)(_U), BKCMP_GE(P)(_U)	(s1)<(s2)	

- When the comparison operation results stored in (n) points from the device specified by (d) are all on (1), SM704 turns on.

Operation error

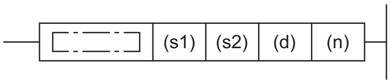
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping.
	The device ranges starting from the ones specified by (s2) and (d) are overlapping.

Comparing 32-bit binary block data

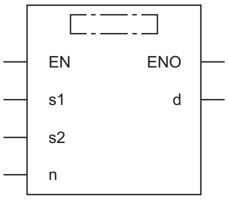
DBKCMP□(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST[*1]	
 (□ is replaced by any of the following: DBKCMP=(P)(_U), DBKCMP<>(P)(_U), DBKCMP>(P)(_U), DBKCMP<=(P)(_U), DBKCMP<(P)(_U), DBKCMP>=(P)(_U).)	ENO:=DBKCMP_□(EN,s1,s2,n,d); ENO:=DBKCMP_□P(EN,s1,s2,n,d);	ENO:=DBKCMP_□_U(EN,s1,s2,n,d); ENO:=DBKCMP_□P_U(EN,s1,s2,n,d);
	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[*2]	

FBD/LD

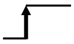


(□ is to be replaced by combination of any of the following: DBKCMP_ and EQ(P)(_U), NE(P)(_U), GT(P)(_U), LE(P)(_U), LT(P)(_U), GE(P)(_U).)[*2]

[*1] The engineering tool with version "1.035M" or later supports the ST.

[*2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
DBKCMP□(_U)	
DBKCMP□P(_U)	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBKCMP□(P)	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S[*1]
	DBKCMP□(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U[*1]
(s2)	DBKCMP□(P)	Start device where the comparison data is stored	—	32-bit signed binary	ANY32_S[*1]
	DBKCMP□(P)_U			32-bit unsigned binary	ANY32_U[*1]
(d)		Start device for storing the comparison operation result	—	Bit	ANY_BOOL[*1]
(n)		Number of data points to be compared	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

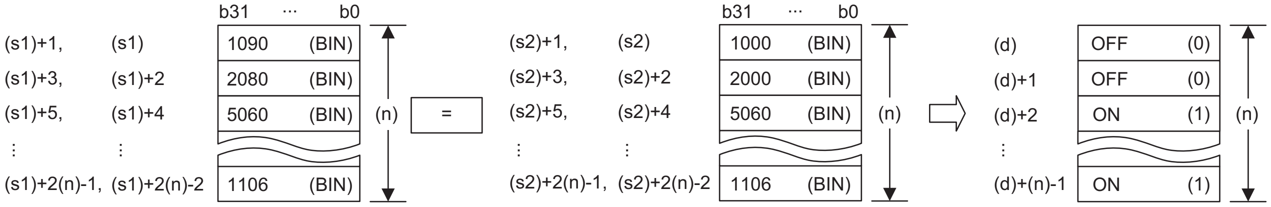
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

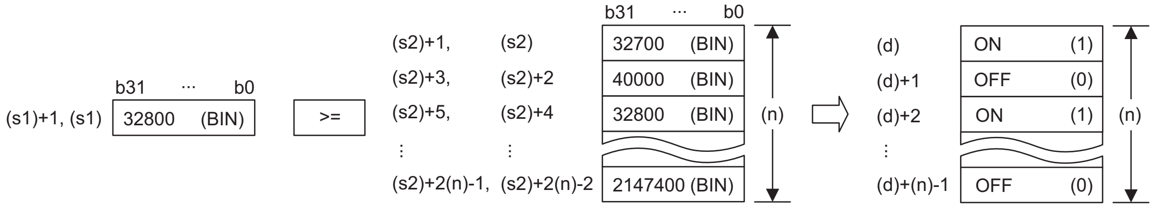
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the (n) points of 32-bit binary data from the device specified by (s1) with the (n) points of 32-bit binary data from the device specified by (s2), and stores the operation result in the device specified by (d) and later.
- If the comparison condition is satisfied, the relevant device specified by (d) turns on; otherwise, the device turns off.



- Comparison operation is performed in units of 32 bits.
- A constant can be specified for (s1).



- Specify (d) outside the device ranges for (n) points from the device specified by (s1) and those from the device specified by (s2).
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
DBKCMP=(P)(_U), DBKCMP_EQ(P)(_U)	(s1)=(s2)	On (1)
DBKCMP<>(P)(_U), DBKCMP_NE(P)(_U)	(s1)≠(s2)	
DBKCMP>(P)(_U), DBKCMP_GT(P)(_U)	(s1)>(s2)	
DBKCMP<=(P)(_U), DBKCMP_LE(P)(_U)	(s1)≤(s2)	
DBKCMP<(P)(_U), DBKCMP_LT(P)(_U)	(s1)<(s2)	
DBKCMP>=(P)(_U), DBKCMP_GE(P)(_U)	(s1)≥(s2)	
DBKCMP=(P)(_U), DBKCMP_EQ(P)(_U)	(s1)≠(s2)	Off (0)
DBKCMP<>(P)(_U), DBKCMP_NE(P)(_U)	(s1)=(s2)	
DBKCMP>(P)(_U), DBKCMP_GT(P)(_U)	(s1)≤(s2)	
DBKCMP<=(P)(_U), DBKCMP_LE(P)(_U)	(s1)>(s2)	
DBKCMP<(P)(_U), DBKCMP_LT(P)(_U)	(s1)≥(s2)	
DBKCMP>=(P)(_U), DBKCMP_GE(P)(_U)	(s1)<(s2)	

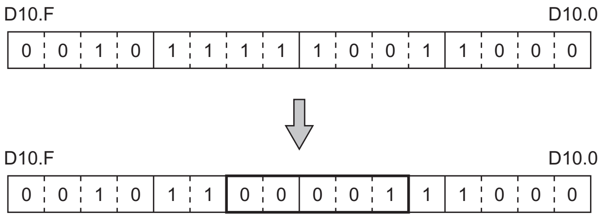
- When the comparison operation results stored in (n) points from the device specified by (d) are all on (1), SM704 turns on.
- If (n) is 0, no processing is performed.

Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping.
	The device ranges starting from the ones specified by (s2) and (d) are overlapping.

Point

When bits of a word device are specified, the bits other than the specified ones for storing the operation result do not change.



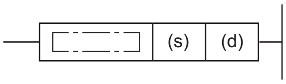
6.2 Arithmetic Operation Instructions

Adding 16-bit binary data

+(P)(_U) [when two operands are set]

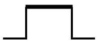
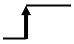
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST
	Not supported (☞ Page 237 +(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 237 +(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
+ +_U	
+P +P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	+(P)	Second addend data or the device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	+(P)	Device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

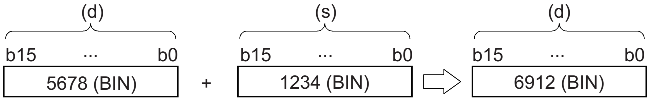
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

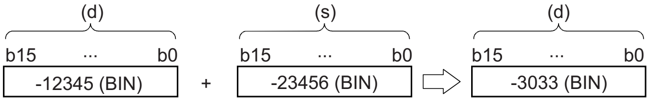
Processing details

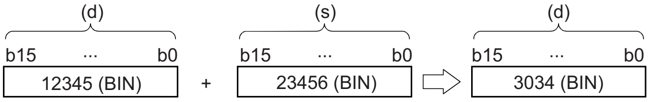
- These instructions add the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



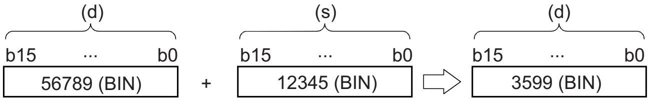
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+(P) instruction]





[+(P)_U instruction]



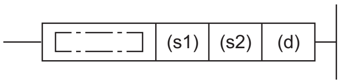
Operation error

There is no operation error.

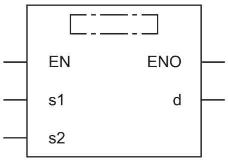
+(P)(_U) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST
	ENO:=PLUS(EN,s1,s2,d); ENO:=PLUSP(EN,s1,s2,d); ENO:=PLUS_U(EN,s1,s2,d); ENO:=PLUSP_U(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by any of the following: PLUS, PLUSP, PLUS_U, PLUSP_U.)

■Execution condition

Instruction	Execution condition
+ +_U	
+P +P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	+(P)	First addend data or the device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	+(P)	Second addend data or the device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	+(P)	Device for storing the operation result	—	16-bit signed binary	ANY16_S
	+(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

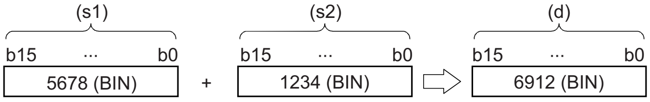
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

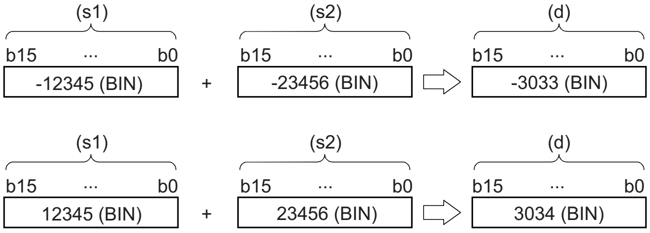
Processing details

- 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).

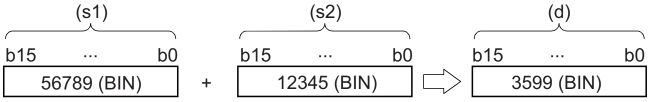


- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+(P) instruction]



[+(P)_U instruction]



Operation error

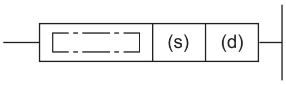
There is no operation error.

Subtracting 16-bit binary data

-(P)(_U) [when two operands are set]

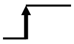
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

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

Ladder	ST
	Not supported (☞ Page 241 -(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 241 -(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
- -_U	
-P -P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	-(P)	Subtrahend data or the device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	-(P)	Device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

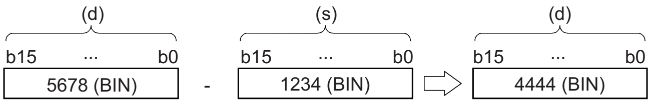
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

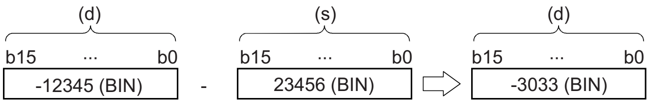
Processing details

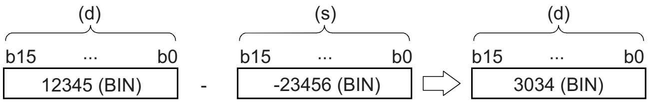
- These instructions subtract the 16-bit binary data in the device specified by (s) from the 16-bit binary data in the device specified by (d), and store the operation result in the device specified by (d).



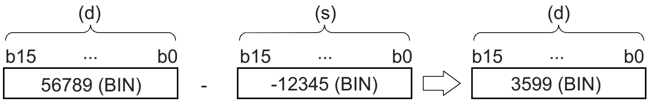
- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]





[-(P)_U instruction]



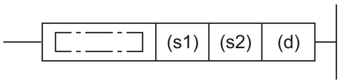
Operation error

There is no operation error.

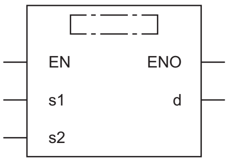
-(P)(_U) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

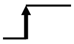
Ladder	ST
	ENO:=MINUS(EN,s1,s2,d); ENO:=MINUSP(EN,s1,s2,d); ENO:=MINUS_U(EN,s1,s2,d); ENO:=MINUSP_U(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by any of the following: MINUS, MINUSP, MINUS_U, MINUSP_U.)

■Execution condition

Instruction	Execution condition
- -_U	
-P -P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	-(P)	Minuend data or the device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	-(P)	Subtrahend data or the device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	-(P)	Device for storing the operation result	—	16-bit signed binary	ANY16_S
	-(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

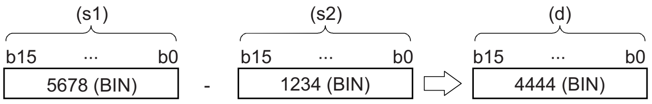
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

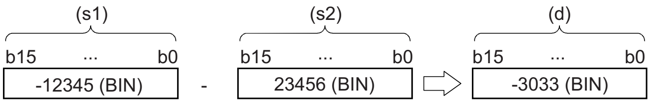
Processing details

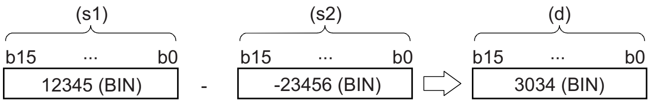
- These instructions subtract the 16-bit binary data in the device specified by (s2) from the 16-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



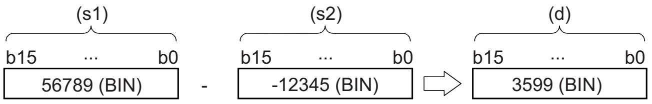
- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]





[-(P)_U instruction]



Operation error

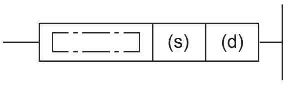
There is no operation error.

Adding 32-bit binary data

D+(P)(_U) [when two operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST
	Not supported (☞ Page 245 D+(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 245 D+(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
D+ D+_U	
D+P D+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	D+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D+(P)	Start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

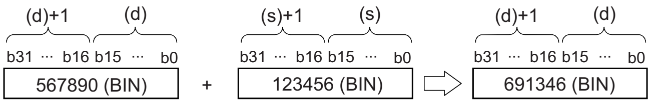
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

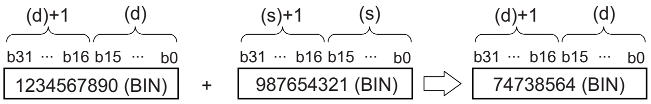
Processing details

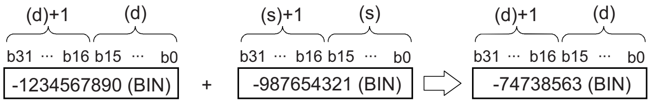
- These instructions add the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



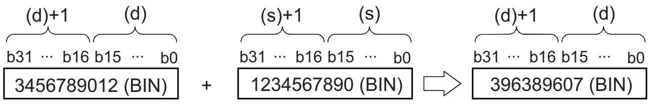
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]



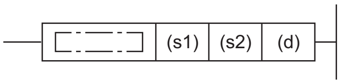
Operation error

There is no operation error.

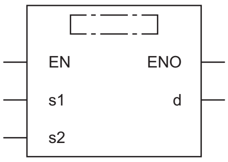
D+(P)(_U) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

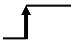
Ladder	ST
	ENO:=DPLUS(EN,s1,s2,d); ENO:=DPLUSP(EN,s1,s2,d); ENO:=DPLUS_U(EN,s1,s2,d); ENO:=DPLUSP_U(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by any of the following: DPLUS, DPLUSP, DPLUS_U, DPLUSP_U.)

■Execution condition

Instruction	Execution condition
D+ D+_U	
D+P D+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	D+(P)	First addend data or the start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D+(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	D+(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

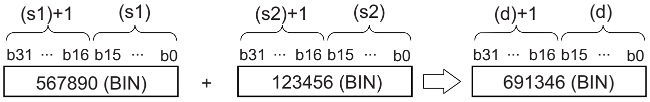
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

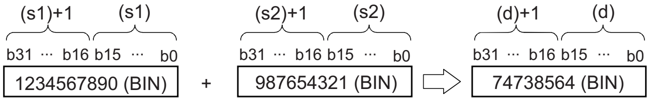
Processing details

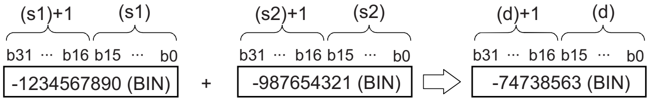
- These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



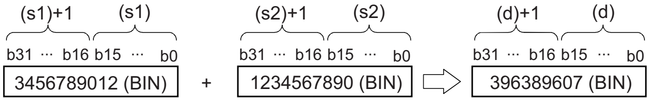
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]



Operation error

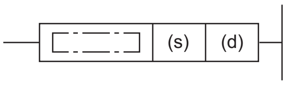
There is no operation error.

Subtracting 32-bit binary data

D-(P)(_U) [when two operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST
	Not supported (☞ Page 249 D-(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 249 D-(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
D- D-_U	
D-P D-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	D-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D-(P)	Start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

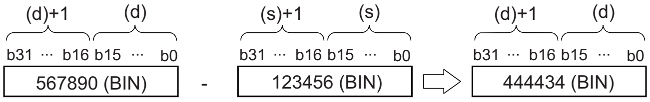
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

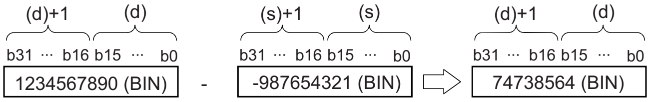
Processing details

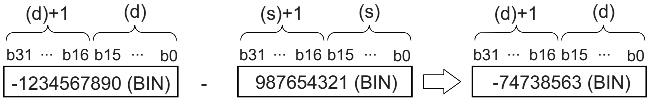
- These instructions subtract the 32-bit binary data in the device specified by (s) from the 32-bit binary data in the device specified by (d) and, and store the operation result in the device specified by (d).



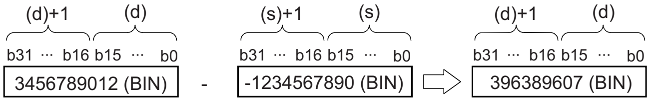
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]



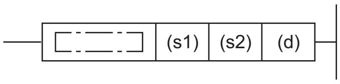
Operation error

There is no operation error.

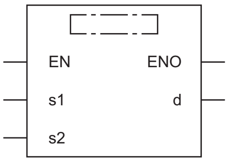
D-(P)(_U) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

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

Ladder	ST
	ENO:=DMINUS(EN,s1,s2,d); ENO:=DMINUSP(EN,s1,s2,d); ENO:=DMINUS_U(EN,s1,s2,d); ENO:=DMINUSP_U(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by any of the following: DMINUS, DMINUSP, DMINUS_U, DMINUSP_U.)

■Execution condition

Instruction	Execution condition
D- D-_U	
D-P D-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	D-(P)	Minuend data or the start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D-(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	D-(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

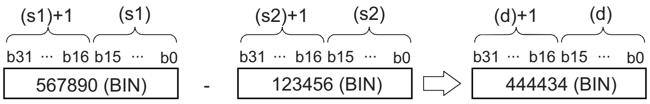
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

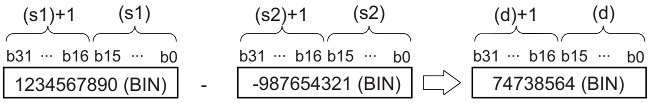
Processing details

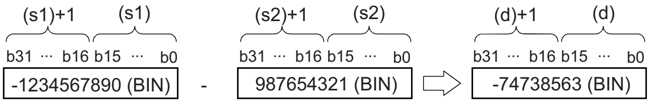
- These instructions subtracts the 32-bit binary data in the device specified by (s2) from the 32-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



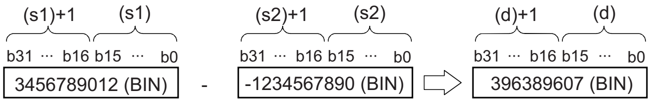
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]



Operation error

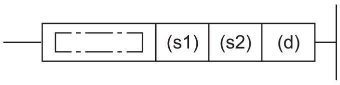
There is no operation error.

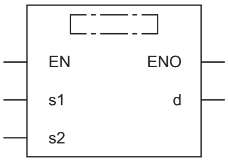
Multiplying 16-bit binary data

*(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

Ladder	ST[*1]	
	ENO:=MULTI(EN,s1,s2,d); ENO:=MULTIP(EN,s1,s2,d);	ENO:=MULTI_U(EN,s1,s2,d); ENO:=MULTIP_U(EN,s1,s2,d);

FBD/LD

(☐ is to be replaced by any of the following: MULTI, MULTIP, MULTI_U, MULTIP_U.)

[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
* *_U	
*P *P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	*(P)	Multiplicand data or the device where multiplicand data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	*(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	*(P)	Multiplier data or the device where multiplier data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	*(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	*(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	*(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

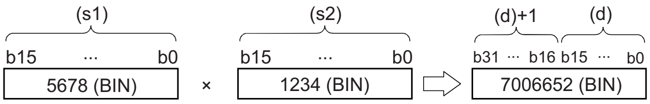
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	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

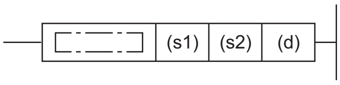
There is no operation error.

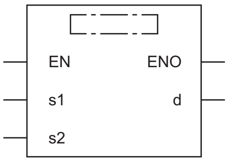
Dividing 16-bit binary data

/(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

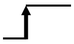
These instructions perform division between the two sets of 16-bit binary data specified.

Ladder	ST[1]	
	ENO:=DIVISION(EN,s1,s2,d); ENO:=DIVISIONP(EN,s1,s2,d);	ENO:=DIVISION_U(EN,s1,s2,d); ENO:=DIVISIONP_U(EN,s1,s2,d);

FBD/LD

(☐ is to be replaced by any of the following: DIVISION, DIVISIONP, DIVISION_U, DIVISIONP_U.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
/ /_U	
/P /P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	/(P)	Dividend data or the device where dividend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	/(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	/(P)	Divisor data or the device where divisor data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	/(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	/(P)	Start device for storing the operation result	—	32-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	/(P)_U			32-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

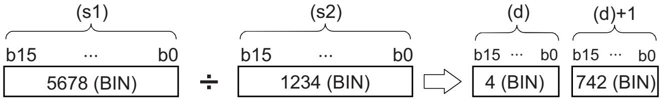
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	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

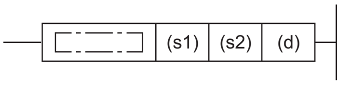
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.

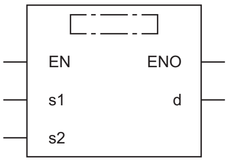
Multiplying 32-bit binary data

D*(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

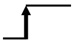
These instructions multiply the two sets of 32-bit binary data specified.

Ladder	ST[1]	
	ENO:=DMULTI(EN,s1,s2,d); ENO:=DMULTIP(EN,s1,s2,d);	ENO:=DMULTI_U(EN,s1,s2,d); ENO:=DMULTIP_U(EN,s1,s2,d);

FBD/LD

(☐ is to be replaced by any of the following: DMULTI, DMULTIP, DMULTI_U, DMULTIP_U.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
D* D*_U	
D*P D*P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	D*(P)	Multiplicand data or the start device where multiplicand data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D*(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D*(P)	Multiplier data or the start device where multiplier data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D*(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D*(P)	Start device for storing the operation result	—	64-bit signed binary	ANY32_S_ARRAY (Number of elements: 2)
	D*(P)_U			64-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

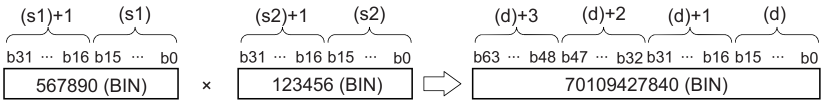
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	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

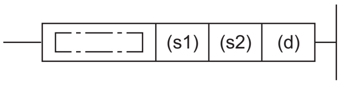
There is no operation error.

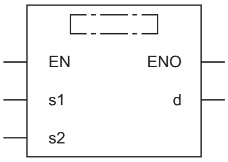
Dividing 32-bit binary data

D/(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

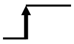
These instructions perform division between the two sets of 32-bit binary data specified.

Ladder	ST[1]	
	ENO:=DDIVISION(EN,s1,s2,d); ENO:=DDIVISIONP(EN,s1,s2,d);	ENO:=DDIVISION_U(EN,s1,s2,d); ENO:=DDIVISIONP_U(EN,s1,s2,d);

FBD/LD

(□ is to be replaced by any of the following: DDIVISION, DDIVISIONP, DDIVISION_U, DDIVISIONP_U.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
D/ D/_U	
D/P D/P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	D/(P)	Dividend data or the start device where dividend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D/(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D/(P)	Divisor data or the start device where divisor data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D/(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D/(P)	Start device for storing the operation result	—	64-bit signed binary	ANY32_S_ARRAY (Number of elements: 2)
	D/(P)_U			64-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

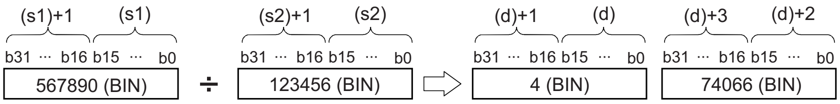
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	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

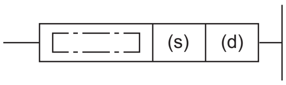
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.

Adding BCD 4-digit data

B+(P) [when two operands are set]

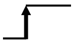
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions add the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 260 B+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 260 B+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B+	
B+P	

Setting data

■Description, range, data type

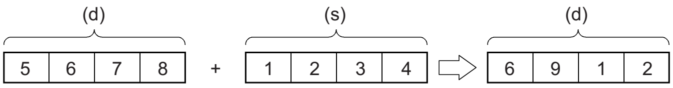
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

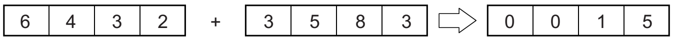
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the BCD 4-digit data in the device specified by (d) and the BCD 4-digit data in the device specified by (s), and store the operation result in the device specified by (d).



- If the result exceeds 9999, the carry bit is ignored. In this case, SM700 does not turn on.



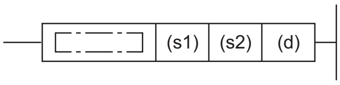
Operation error

Error code (SD0)	Description
3405H	The BCD data in the device specified by (s) is out of the range, 0 to 9999.
	The BCD data in the device specified by (d) is out of the range, 0 to 9999.

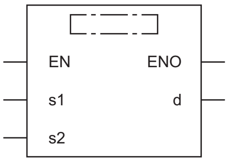
B+(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add the two sets of BCD 4-digit data specified.

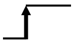
Ladder	ST
	ENO:=BPLUS(EN,s1,s2,d); ENO:=BPLUSP(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by either of the following: BPLUS, BPLUSP.)

■Execution condition

Instruction	Execution condition
B+	
B+P	

Setting data

■Description, range, data type

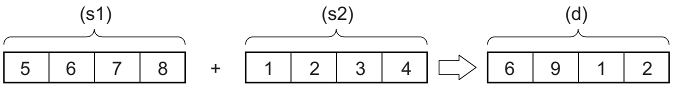
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

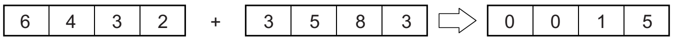
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



- If the result exceeds 9999, the carry bit is ignored. In this case, SM700 does not turn on.



Operation error

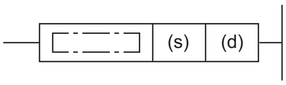
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Subtracting BCD 4-digit data

B-(P) [when two operands are set]

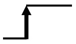
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform subtraction between the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 264 B-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 264 B-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B-	
B-P	

Setting data

■Description, range, data type

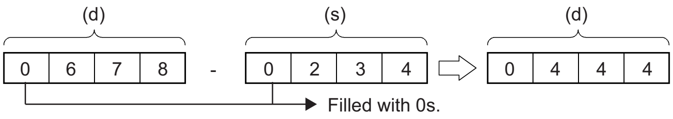
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

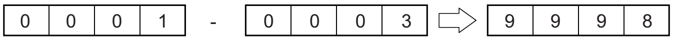
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s) from the 32-bit binary data in the device specified by (d), and store the operation result in the device specified by (d).



- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



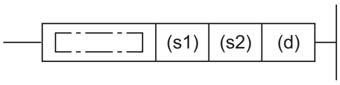
Operation error

Error code (SD0)	Description
3405H	The BCD data in the device specified by (s) is out of the range, 0 to 9999.
	The BCD data in the device specified by (d) is out of the range, 0 to 9999.

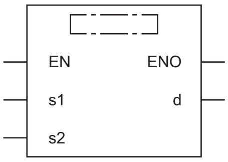
B-(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two sets of BCD 4-digit data specified.

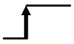
Ladder	ST
	ENO:=BMINUS(EN,s1,s2,d); ENO:=BMINUSP(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by either of the following: BMINUS, BMINUSP.)

■Execution condition

Instruction	Execution condition
B-	
B-P	

Setting data

■Description, range, data type

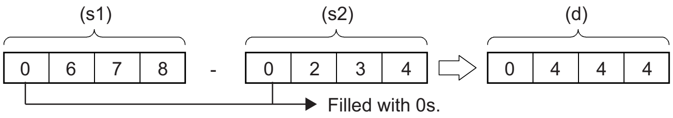
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

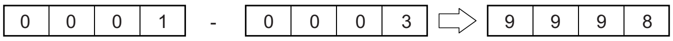
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s2) from the BCD 4-digit data in the device specified by (s1), and store the operation result in the device specified by (d).



- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



Operation error

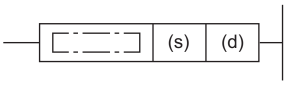
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Adding BCD 8-digit data

DB+(P) [when two operands are set]

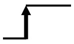
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions add the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 268 DB+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 268 DB+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB+	
DB+P	

Setting data

■Description, range, data type

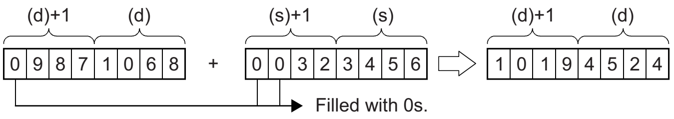
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, 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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

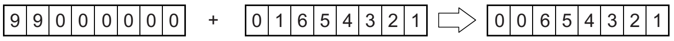
Processing details

- These instructions add the BCD 8-digit data in the device specified by (d) and the BCD 8-digit data in the device specified by (s), and store the operation result in the device specified by (d).



(d)+1, (s)+1: Upper 4 digits
(d), (s): Lower 4 digits

- If the result exceeds 99999999, the carry bit is ignored. In this case, SM700 does not turn on.



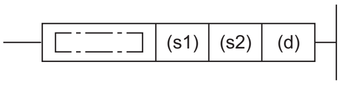
Operation error

Error code (SD0)	Description
3405H	The BCD data in the device specified by (s) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (d) is out of the range, 0 to 99999999.

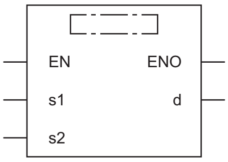
DB+(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add the two sets of BCD 8-digit data specified.

Ladder	ST
	ENO:=DBPLUS(EN,s1,s2,d); ENO:=DBPLUSP(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by either of the following: DBPLUS, DBPLUSP.)

■Execution condition

Instruction	Execution condition
DB+	
DB+P	

Setting data

■Description, range, data type

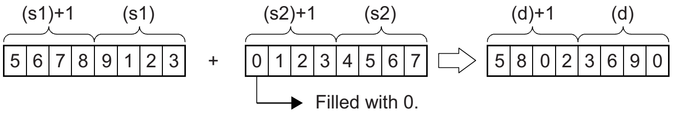
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

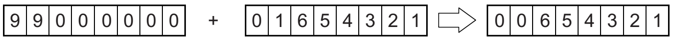
Processing details

- These instructions add the BCD 8-digit data in the device specified by (s1) and the BCD 8-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



(d)+1, (s1)+1, (s2)+1: Upper 4 digits
(d), (s1), (s2): Lower 4 digits

- If the result exceeds 99999999, the carry bit is ignored. In this case, SM700 does not turn on.



Operation error

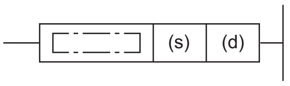
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Subtracting BCD 8-digit data

DB-(P) [when two operands are set]

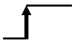
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 272 DB-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 272 DB-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB-	
DB-P	

Setting data

■Description, range, data type

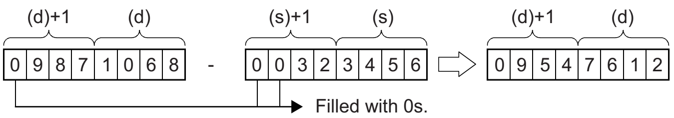
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, 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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

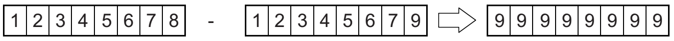
Processing details

- These instructions subtract the BCD 8-digit data in the device specified by (s) from the BCD 8-digit data in the device specified by (d), and store the operation result in the device specified by (d).



(d)+1, (s)+1: Upper 4 digits
(d), (s): Lower 4 digits

- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



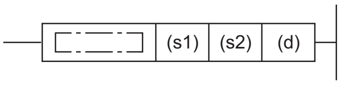
Operation error

Error code (SD0)	Description
3405H	The BCD data in the device specified by (s) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (d) is out of the range, 0 to 99999999.

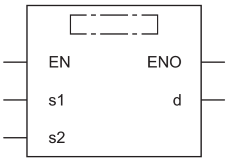
DB-(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two sets of BCD 8-digit data specified.

Ladder	ST
	ENO:=DBMINUS(EN,s1,s2,d); ENO:=DBMINUSP(EN,s1,s2,d);

FBD/LD



(□ is to be replaced by either of the following: DBMINUS, DBMINUSP.)

■Execution condition

Instruction	Execution condition
DB-	
DB-P	

Setting data

■Description, range, data type

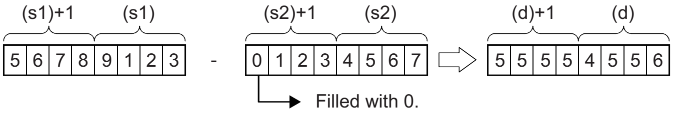
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

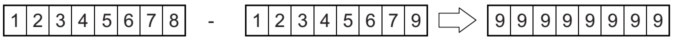
Processing details

- These instructions subtract the BCD 8-digit data in the device specified by (s2) from the BCD 8-digit data in the device specified by (s1), and store the operation result in the device specified by (d).



(d)+1, (s1)+1, (s2)+1: Upper 4 digits
(d), (s1), (s2): Lower 4 digits

- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



Operation error

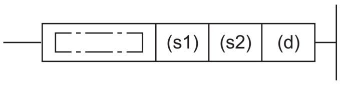
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Multiplying BCD 4-digit data

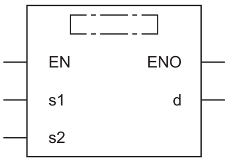
B*(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions multiply the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(□ is to be replaced by either of the following: BMULTI, BMULTIP.)

■Execution condition

Instruction	Execution condition
B*	
B*P	

Setting data

■Description, range, data type

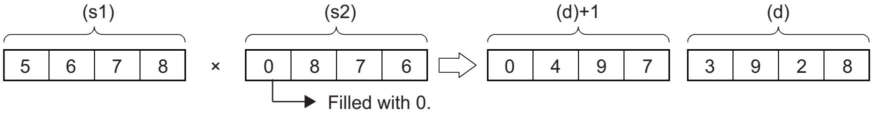
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the device where multiplicand data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Multiplier data or the device where multiplier data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions multiply the BCD 4-digit data in the device specified by (s1) by the BCD 4-digit data in the device specified by (s2), and store the operation result in the device specified by (d). ((d)+1: Upper 4 digits, (d): Lower 4 digits)



Operation error

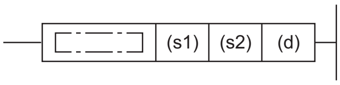
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Dividing BCD 4-digit data

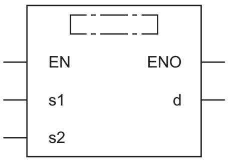
B/(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform division between the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(□ is to be replaced by either of the following: BDIVISION, BDIVISIONP.)

■Execution condition

Instruction	Execution condition
B/	
B/P	

Setting data

■Description, range, data type

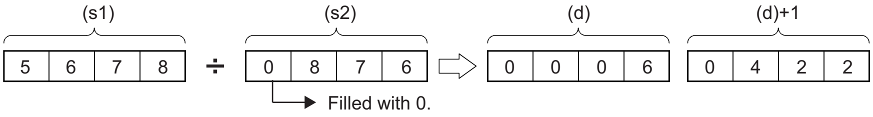
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the device where dividend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Divisor data or the device where divisor data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions divide the BCD 4-digit data in the device specified by (s1) by the BCD 4-digit 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.
  - Quotient (BCD 4 digits)···Stored in lower 16 bits.
  - Remainder (BCD 4 digits)···Stored in upper 16 bits.

Operation error

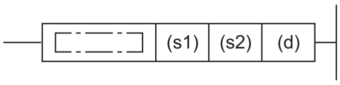
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Multiplying BCD 8-digit data

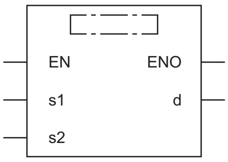
DB*(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions multiply the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(□ is to be replaced by either of the following: DBMULTI, DBMULTIP.)

■Execution condition

Instruction	Execution condition
DB*	
DB*P	

Setting data

■Description, range, data type

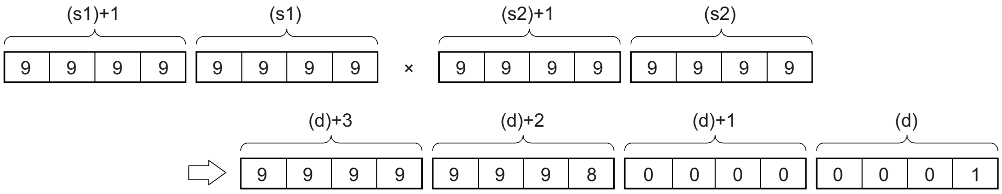
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Multiplier data or the start device where multiplier data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 16-digit	ANY32_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions multiply the BCD 8-digit data in the device specified by (s1) by the BCD 8-digit 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 8 digits (lower 32 bits) of the operation result are stored.

Ex.

Operation result when (d) is a bit device

- K1···Lower 1 digit (b0 to b3)
- K4···Lower 4 digits (b0 to b15)
- K8···Lower 8 digits (b0 to b31)

Operation error

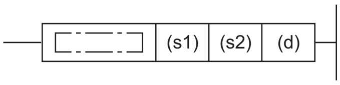
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

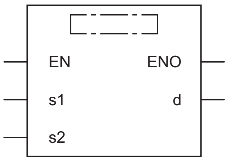
Dividing BCD 8-digit data

DB/(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions perform division between the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported

FBD/LD

(□ is to be replaced by either of the following: DBDIVISION, DBDIVISIONP.)

■Execution condition

Instruction	Execution condition
DB/	
DB/P	

Setting data

■Description, range, data type

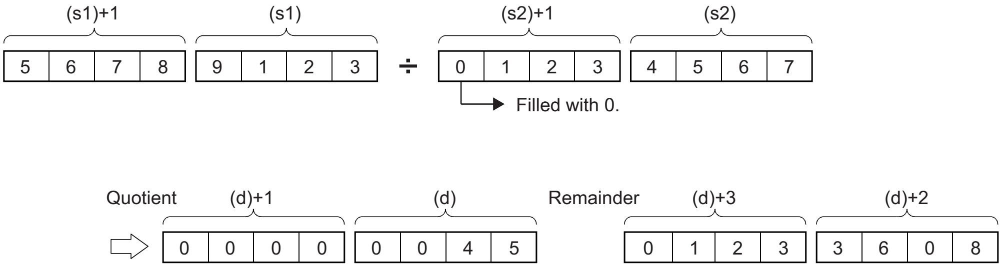
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Divisor data or the start device where divisor data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 16-digit	ANY32_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions divide the BCD 8-digit data in the device specified by (s1) by the BCD 8-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



(d)+1, (d)+3: Upper 4 digits
(d), (d)+2: Lower 4 digits

- As the operation result, the quotient and remainder are stored in 64 bits.
  - Quotient (BCD 8 digits)···Stored in lower 32 bits.
  - Remainder (BCD 8 digits)···Stored in upper 32 bits.
- When (d) is a bit device, the remainder is not stored.

Operation error

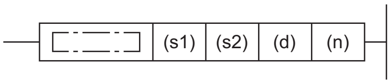
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Adding 16-bit binary block data

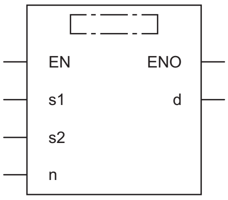
BK+(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

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

Ladder	ST[1]	
	ENO:=BKPLUS(EN,s1,s2,n,d); ENO:=BKPLUSP(EN,s1,s2,n,d);	ENO:=BKPLUS_U(EN,s1,s2,n,d); ENO:=BKPLUSP_U(EN,s1,s2,n,d);

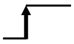
FBD/LD



(□ is to be replaced by any of the following: DMINUS, DMINUSP, DMINUS_U, DMINUSP_U.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
BK+ BK+_U	
BK+P BK+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BK+(P)	First addend data or the start device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S[1]
	BK+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U[1]
(s2)	BK+(P)	Second addend data or the start device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S[1]
	BK+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U[1]
(d)	BK+(P)	Start device for storing the operation result	—	16-bit signed binary	ANY16_S[1]
	BK+(P)_U			16-bit unsigned binary	ANY16_U[1]
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

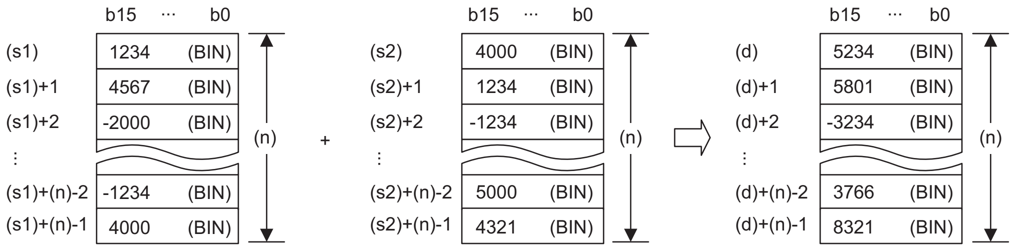
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

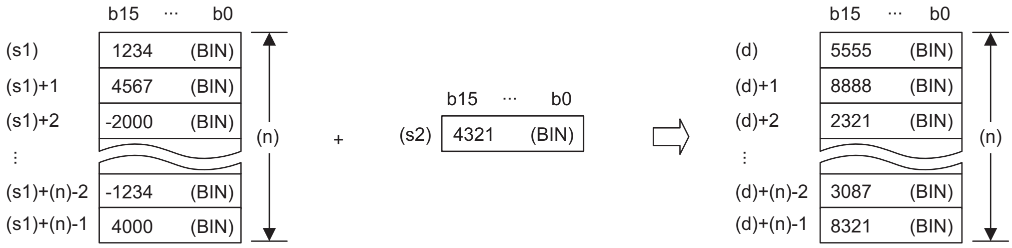
- These instructions add the (n) points of 16-bit binary data from the device specified by (s1) and the (n) points of 16-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

Ex.

When a device is specified by (s2) (signed value specification)



When a constant is specified by (s2) (signed value specification)



- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K32767 (H7FFF) + K2 (H0002) ⇨ K-32767 (H8001) K-32767 (H8001) + K-2 (HFFFE) ⇨ K32767 (H7FFF)	K65535 (HFFFF) + K1 (H0001) ⇨ K0 (H0000)

Operation error

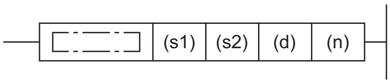
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Subtracting 16-bit binary block data

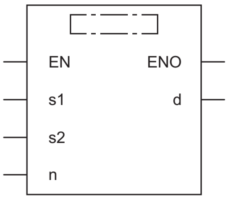
BK-(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

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

Ladder	ST[1]	
	ENO:=BKMINUS(EN,s1,s2,n,d); ENO:=BKMINUSP(EN,s1,s2,n,d);	ENO:=BKMINUS_U(EN,s1,s2,n,d); ENO:=BKMINUSP_U(EN,s1,s2,n,d);

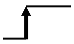
FBD/LD



(□ is to be replaced by any of the following: BKMINUS, BKMINUSP, BKMINUS_U, BKMINUSP_U.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
BK- BK-_U	
BK-P BK-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BK-(P)	Minuend data or the start device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S[1]
	BK-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U[1]
(s2)	BK-(P)	Subtrahend data or the start device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S[1]
	BK-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U[1]
(d)	BK-(P)	Start device for storing the operation result	—	16-bit signed binary	ANY16_S[1]
	BK-(P)_U			16-bit unsigned binary	ANY16_U[1]
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

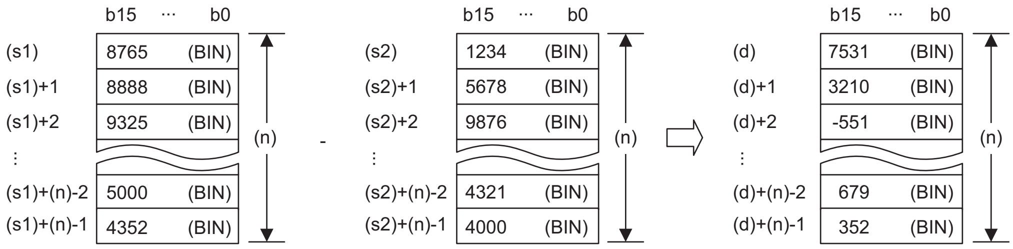
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

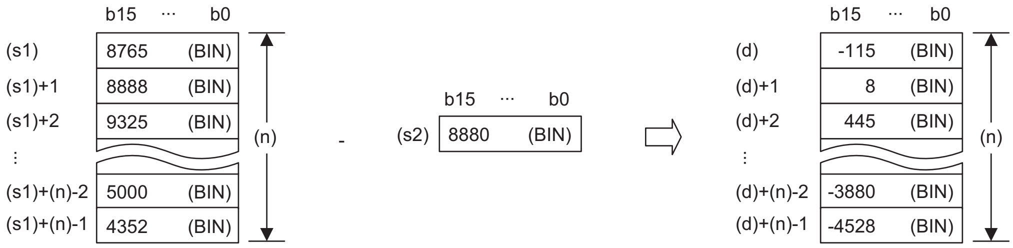
- These instructions subtract the (n) points of 16-bit binary data from the device specified by (s2) or the constant from the (n) points of 16-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

Ex.

When a device is specified by (s2)



When a constant is specified by (s2)



- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K-32767 (H8001) - K2 (H0002) ⇨ K32766 (H7FFE)	K0 (H0000) - K1 (H0001) ⇨ K65535 (HFFFF)
K32767 (H7FFF) - K-2 (HFFFE) ⇨ K-32767 (H8001)	

Operation error

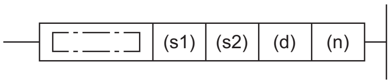
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Adding 32-bit binary block data

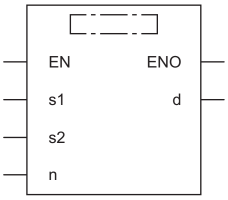
DBK+(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

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

Ladder	ST[*1]	
	ENO:=DBKPLUS(EN,s1,s2,n,d); ENO:=DBKPLUSP(EN,s1,s2,n,d);	ENO:=DBKPLUS_U(EN,s1,s2,n,d); ENO:=DBKPLUSP_U(EN,s1,s2,n,d);

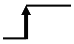
FBD/LD



(□ is to be replaced by any of the following: DBKPLUS, DBKPLUSP, DBKPLUS_U, DBKPLUSP_U.)

[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBK+ DBK+_U	
DBK+P DBK+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBK+(P)	First addend data or the start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S[*1]
	DBK+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U[*1]
(s2)	DBK+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S[*1]
	DBK+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U[*1]
(d)	DBK+(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S[*1]
	DBK+(P)_U			32-bit unsigned binary	ANY32_U[*1]
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

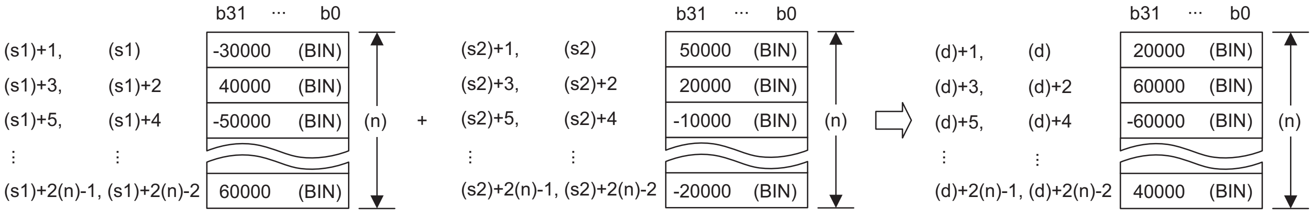
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

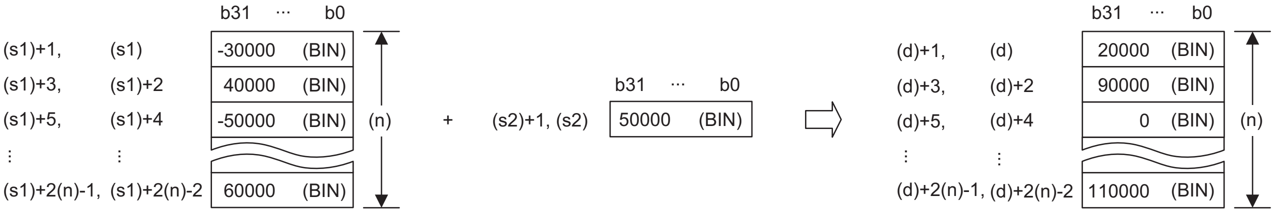
- These instructions add the (n) points of 32-bit binary data from the device specified by (s1) and the (n) points of 32-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



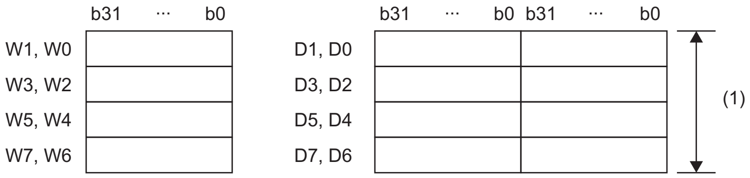
When a constant is specified by (s2) (signed value specification)



- Operation is possible when the same device is specified by (s1) or (s2) and (d). However, if the device range of (n) points from (s1) or (s2) and the device range of (n) points from (d) are partly overlapped, an error results.

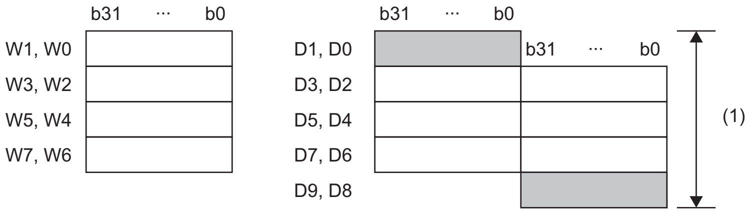
Ex.

When the four points of device from that specified by (s2) and (d) exactly match



(1) Operation is possible because they exactly match.

When four points of device from that specified by (s2) and (d) are partly overlapped



(1) An operation error results because they partly match.

- If (n) is 0, no processing is performed.
- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K2147483647 (H7FFFFFFF) + K2 (H00000002) ⇨ K-2147483647 (H80000001) K-2147483647 (H80000001) + K-2 (HFFFFFFFE) ⇨ K2147483647 (H7FFFFFFF)	K4294967295 (HFFFFFFFF) + K1 (H00000001) ⇨ K0 (H00000000)

Operation error

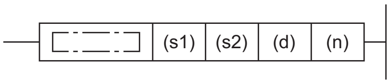
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Subtracting 32-bit binary block data

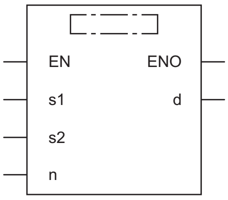
DBK-(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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

Ladder	ST*1	
	ENO:=DBKMINUS(EN,s1,s2,n,d); ENO:=DBKMINUSP(EN,s1,s2,n,d);	ENO:=DBKMINUS_U(EN,s1,s2,n,d); ENO:=DBKMINUSP_U(EN,s1,s2,n,d);

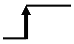
FBD/LD



(□ is to be replaced by any of the following: DBKMINUS, DBKMINUSP, DBKMINUS_U, DBKMINUSP_U.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBK- DBK-_U	
DBK-P DBK-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBK-(P)	Minuend data or the start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(s2)	DBK-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(d)	DBK-(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S*1
	DBK-(P)_U			32-bit unsigned binary	ANY32_U*1
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

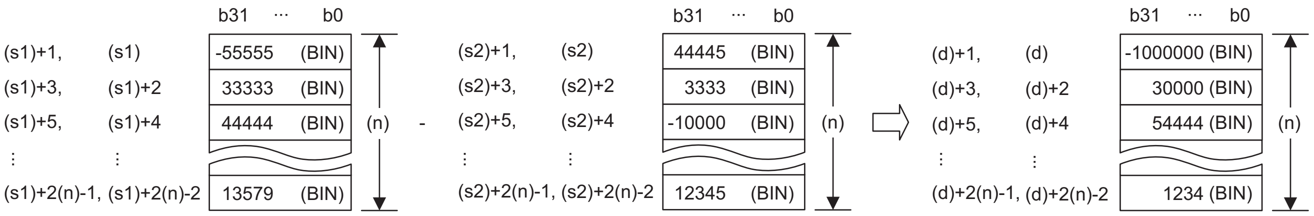
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

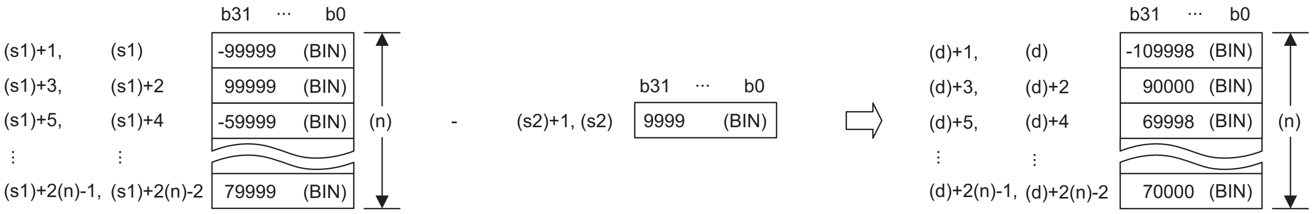
- These instructions subtract the (n) points of 32-bit binary data from the device specified by (s2) or the constant from the (n) points of 32-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



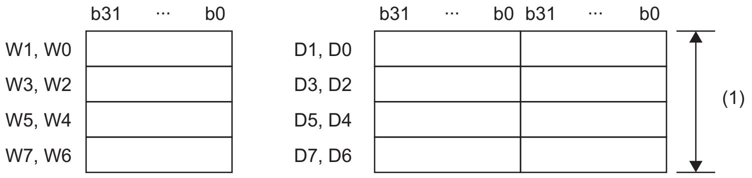
When a constant is specified by (s2) (signed value specification)



- Operation is possible when the same device is specified by (s1) or (s2) and (d). However, if the device range of (n) points from (s1) or (s2) and the device range of (n) points from (d) are partly overlapped, an error results.

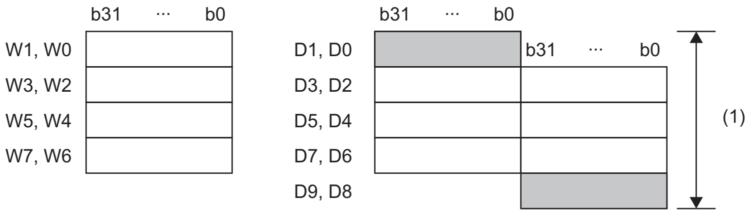
Ex.

When the four points of device from that specified by (s2) and (d) exactly match



(1) Operation is possible because they exactly match.

When four points of device from that specified by (s2) and (d) are partly overlapped



(1) An operation error results because they partly match.

- If (n) is 0, no processing is performed.
- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K2147483647 (H7FFFFFFF) - K-2 (HFFFFFFFE) ⇨ K-2147483647 (H80000001) K-2147483647 (H80000001) - K2 (H00000002) ⇨ K2147483647 (H7FFFFFFF)	K0 (H00000000) - K1 (H00000001) ⇨ K4294967295 (HFFFFFFFF)

Operation error

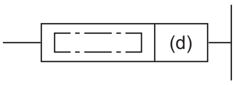
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Incrementing 16-bit binary data

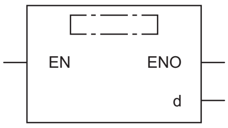
INC(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

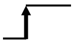
These instructions increment the specified 16-bit binary data by one.

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

FBD/LD



■Execution condition

Instruction	Execution condition
INC INC_U	
INCP INCP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(d)	INC(P)	Increment target device	-32768 to 32767	16-bit signed binary	ANY16_S
	INC(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

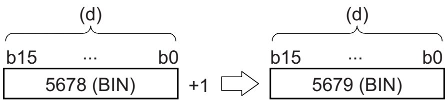
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

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



- When the INC(P) 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(P)_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

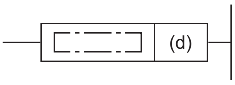
There is no operation error.

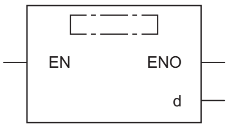
Decrementing 16-bit binary data

DEC(P)(_U)

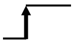
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

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

FBD/LD


■Execution condition

Instruction	Execution condition
DEC DEC_U	
DECP DECP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(d)	DEC(P)	Decrement target device	-32768 to 32767	16-bit signed binary	ANY16_S
	DEC(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

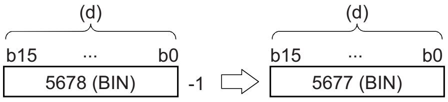
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

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



- When the DEC(P) 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(P)_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

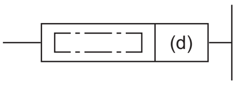
There is no operation error.

Incrementing 32-bit binary data

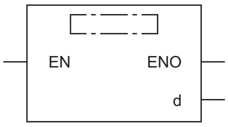
DINC(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

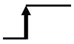
These instructions increment the specified 32-bit binary data by one.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DINC DINC_U	
DINCP DINCP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(d)	DINC(P)	Increment target start device	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DINC(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

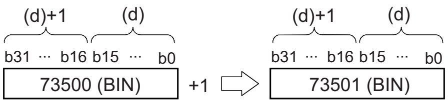
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

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



- When the DINC(P) 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(P)_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

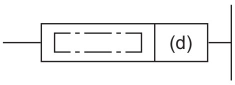
There is no operation error.

Decrementing 32-bit binary data

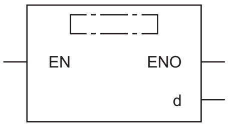
DDEC(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

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

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

FBD/LD



■Execution condition

Instruction	Execution condition
DDEC DDEC_U	
DDECP DDECP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(d)	DDEC(P)	Decrement target start device	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DDEC(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

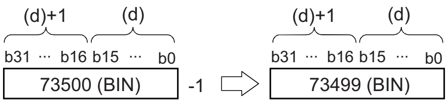
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

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



- When the DDEC(P) 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(P) 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(P)_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.

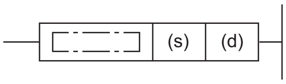
6.3 Logical Operation Instructions

Performing an AND operation on 16-bit data

WAND(P) [when two operands are set]

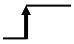
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an AND operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 302 WAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 302 WAND(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WAND	
WANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

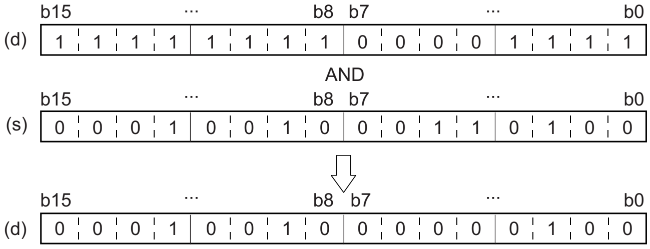
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the 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.

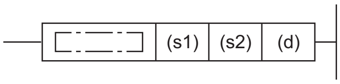
Operation error

There is no operation error.

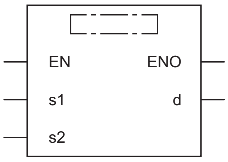
WAND(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

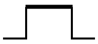
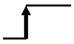
These instructions perform an AND operation on the two sets of 16-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
WAND	
WANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

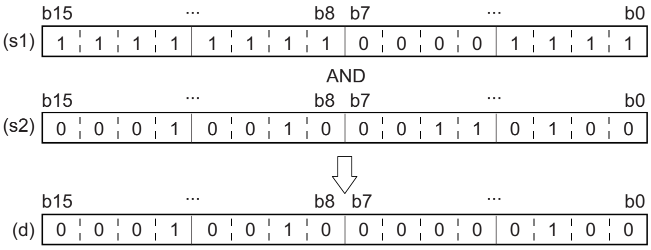
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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

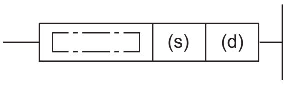
There is no operation error.

Performing an AND operation on 32-bit data

DAND(P) [when two operands are set]

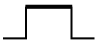
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an AND operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 306 DAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 306 DAND(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DAND	
DANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

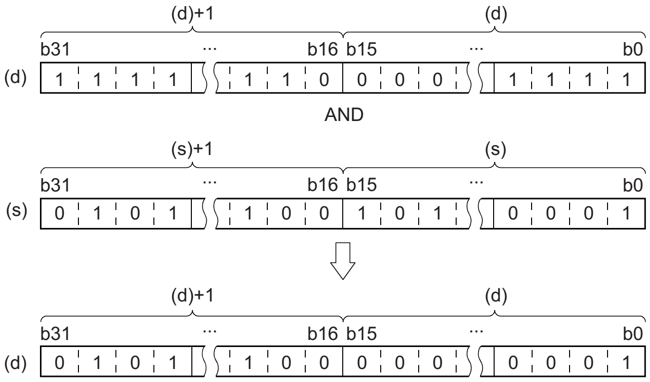
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the 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.

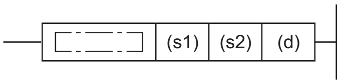
Operation error

There is no operation error.

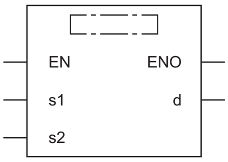
DAND(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

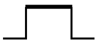
These instructions perform an AND operation on the two sets of 32-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DAND	
DANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

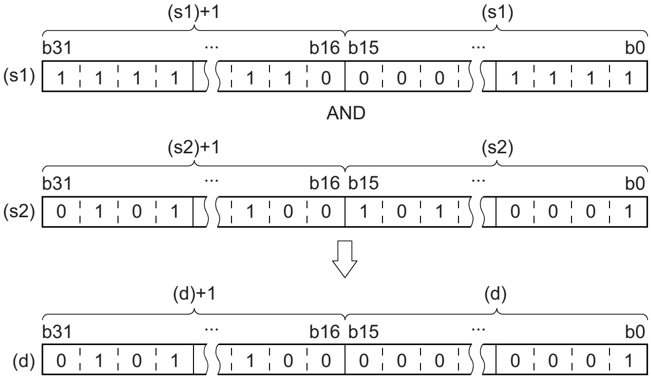
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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

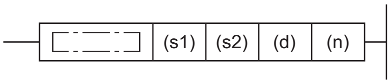
There is no operation error.

Performing an AND operation on 16-bit block data

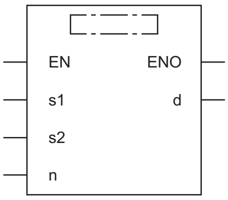
BKAND(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

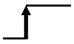
These instructions perform an AND operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKAND(EN,s1,s2,n,d); ENO:=BKANDP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKAND	
BKANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the start device where logical AND data is stored	—	16-bit signed binary	ANY16[*1]
(s2)	Logical AND data or the start device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16[*1]
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

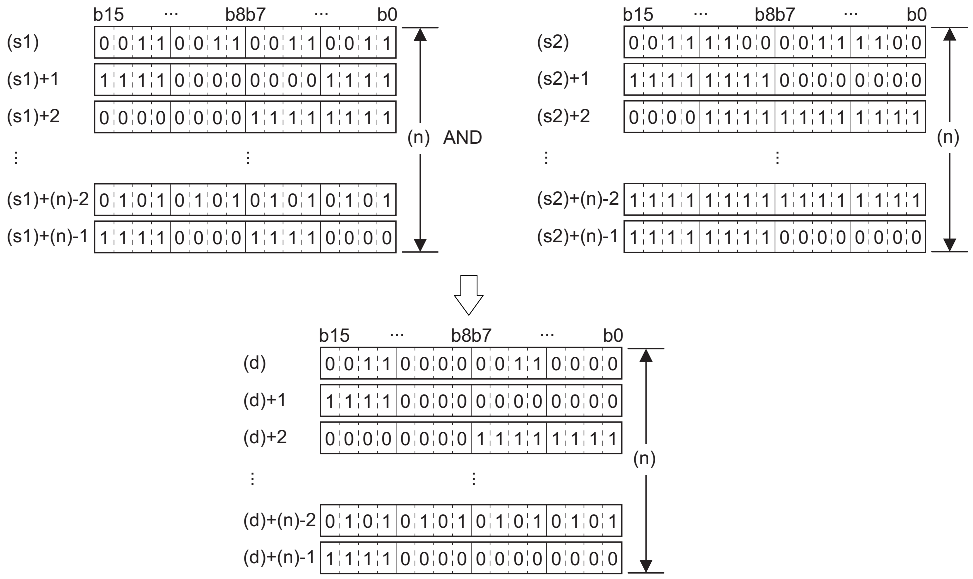
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(s2)[*1]	—	—	○	—	—	—	—	○	○	—	—	—
(d)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

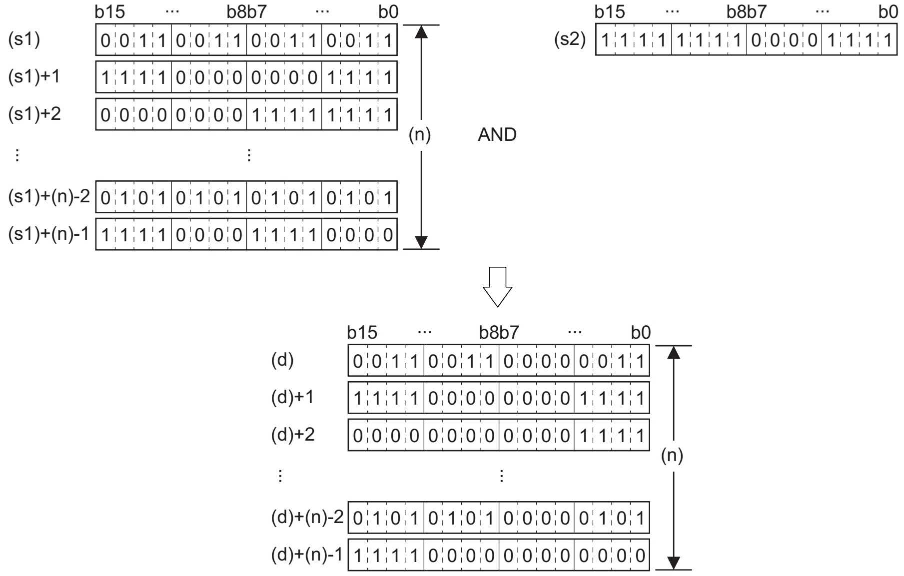
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an AND operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



Operation error

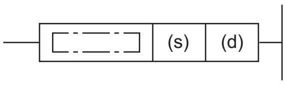
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an OR operation on 16-bit data

WOR(P) [when two operands are set]

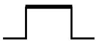
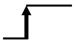
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an OR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 312 WOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 312 WOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WOR	
WORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

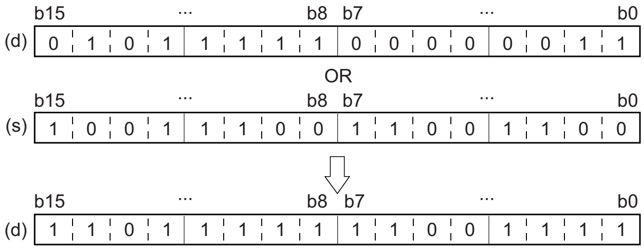
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the 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.

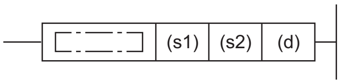
Operation error

There is no operation error.

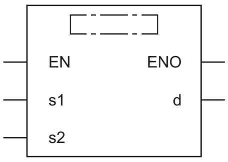
WOR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

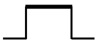
These instructions perform an OR operation on the two sets of 16-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
WOR	
WORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

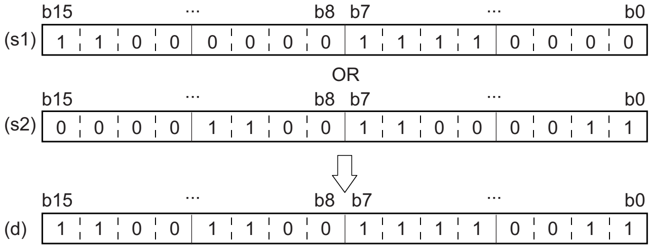
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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
(s2)	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1	1
(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

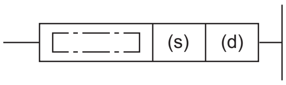
There is no operation error.

Performing an OR operation on 32-bit data

DOR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an OR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 316 DOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 316 DOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DOR	
DORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

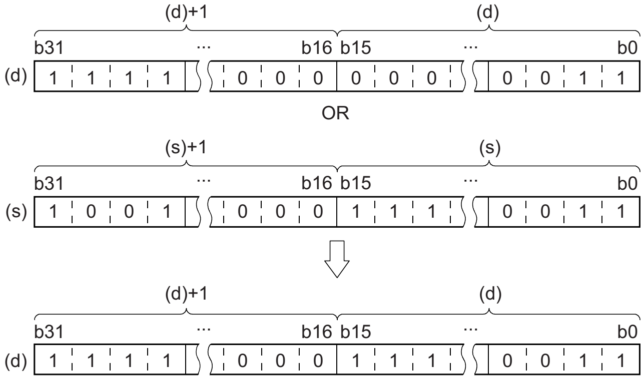
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the 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.

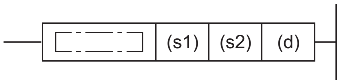
Operation error

There is no operation error.

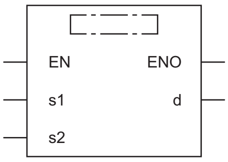
DOR(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an OR operation on the two sets of 32-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DOR	
DORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

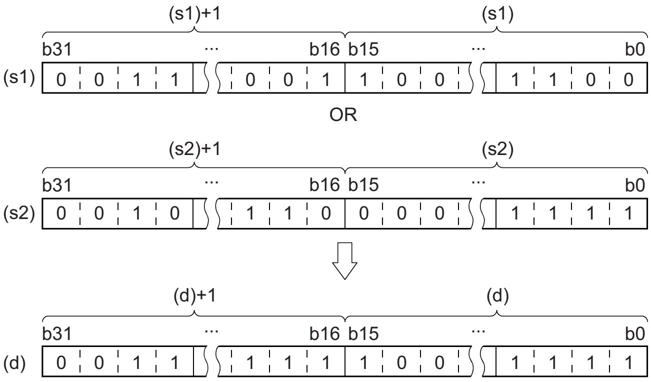
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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

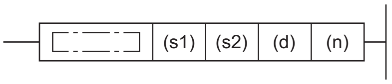
There is no operation error.

Performing an OR operation on 16-bit block data

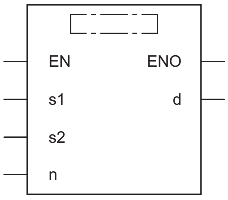
BKOR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

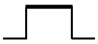
These instructions perform an OR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKOR(EN,s1,s2,n,d); ENO:=BKORP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKOR	
BKORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the start device where logical OR data is stored	—	16-bit signed binary	ANY16[*1]
(s2)	Logical OR data or the start device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16[*1]
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

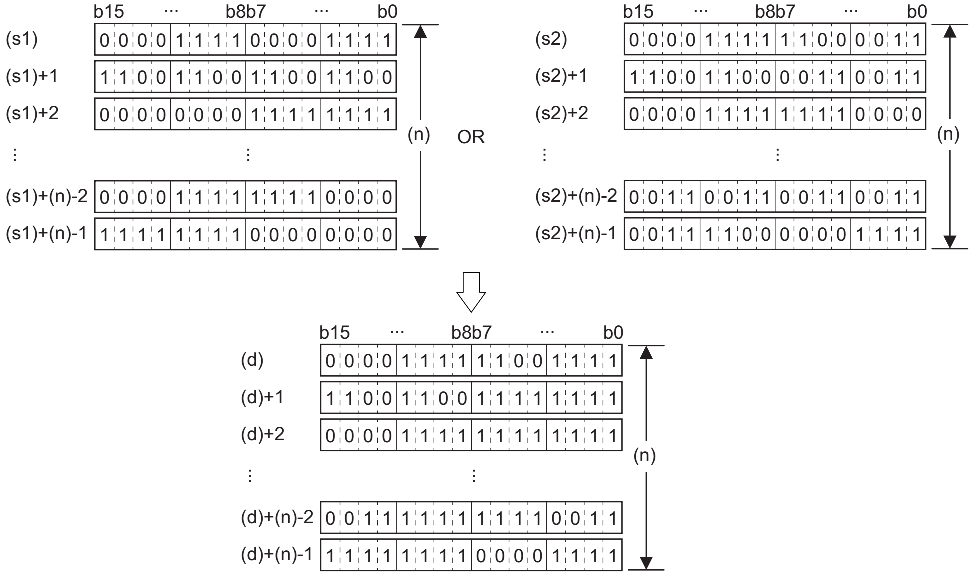
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(s2)[*1]	—	—	○	—	—	—	—	○	○	—	—	—
(d)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

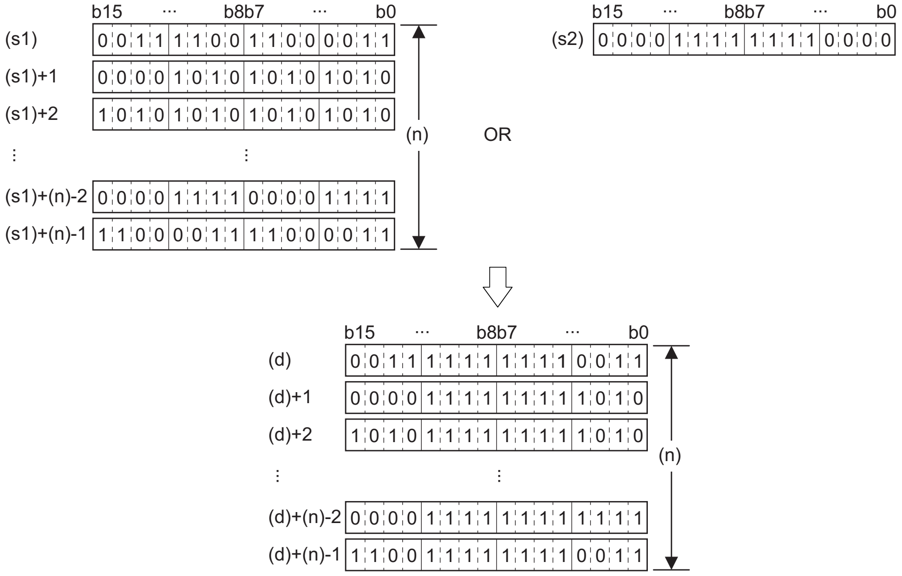
[*1] The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an OR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



Operation error

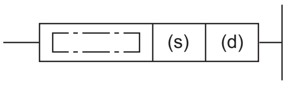
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XOR operation on 16-bit data

WXOR(P) [when two operands are set]

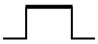
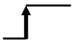
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 322 WXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 322 WXOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WXOR	
WXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

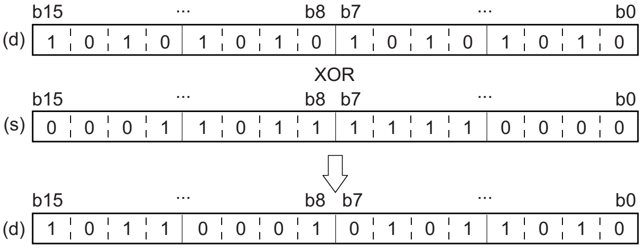
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the 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.

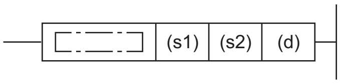
Operation error

There is no operation error.

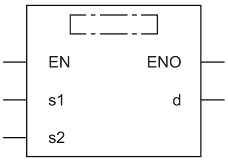
WXOR(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XOR operation on the two sets of 16-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
WXOR	
WXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

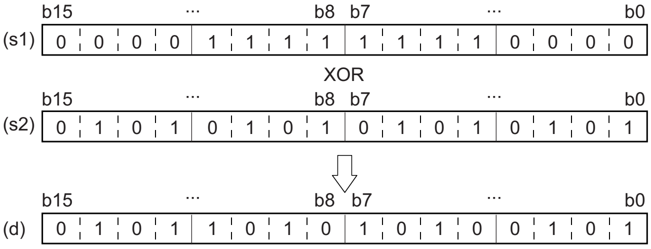
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store the operation result in the device specified by (d).



(s1) 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0

(s2) 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

(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

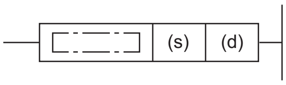
There is no operation error.

Performing an XOR operation on 32-bit data

DXOR(P) [when two operands are set]

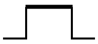
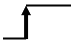
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 326 DXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 326 DXOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DXOR	
DXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

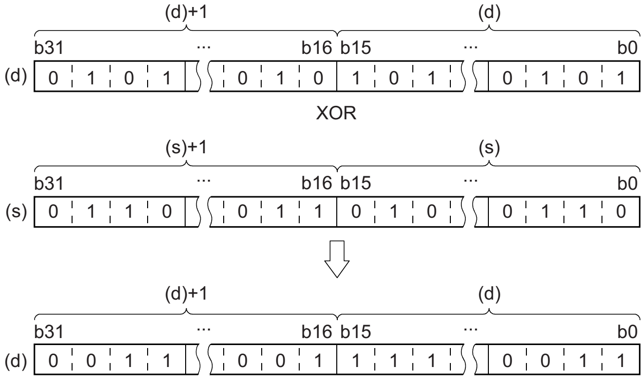
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the 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.

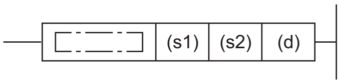
Operation error

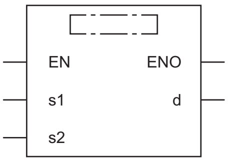
There is no operation error.

DXOR(P) [when three operands are set]

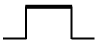
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XOR operation on the two sets of 32-bit binary data specified.

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

FBD/LD


■Execution condition

Instruction	Execution condition
DXOR	
DXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

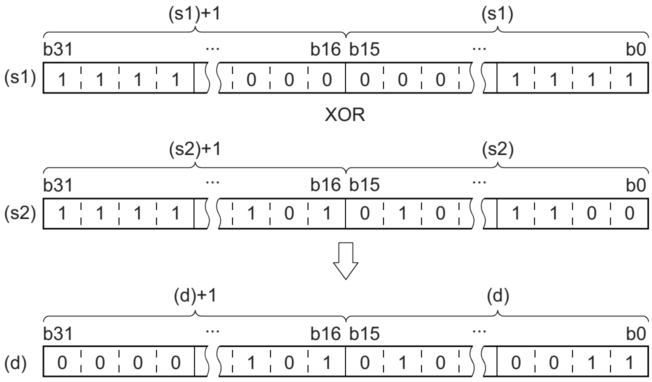
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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

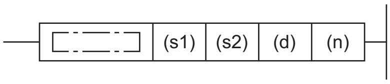
There is no operation error.

Performing an XOR operation on 16-bit block data

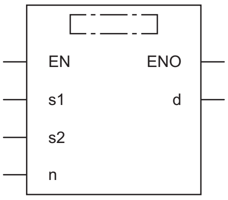
BKXOR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

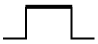
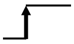
These instructions perform an XOR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKXOR(EN,s1,s2,n,d); ENO:=BKXORP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKXOR	
BKXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the logical operation data is stored	—	16-bit signed binary	ANY16*1
(s2)	Logical operation data or the start device where the logical operation data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16*1
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

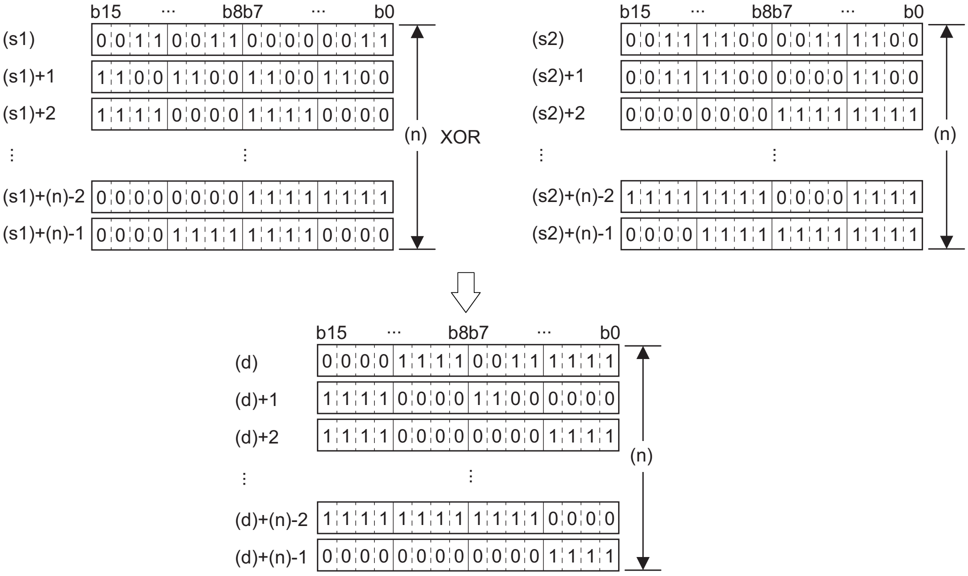
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

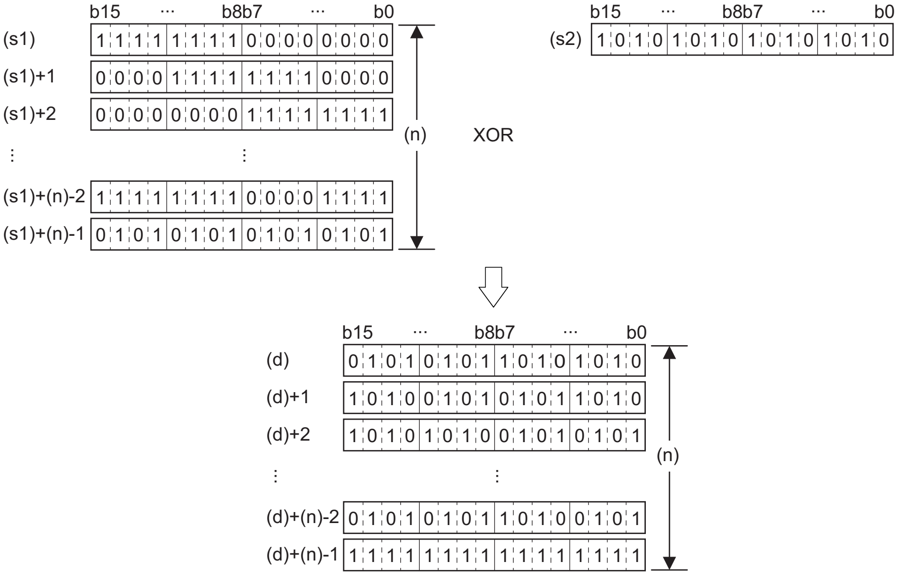
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an XOR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



Operation error

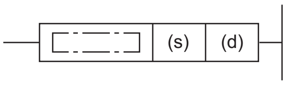
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XNOR operation on 16-bit data

WXNR(P) [when two operands are set]

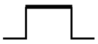
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XNOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 332 WXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 332 WXNR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WXNR	
WXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

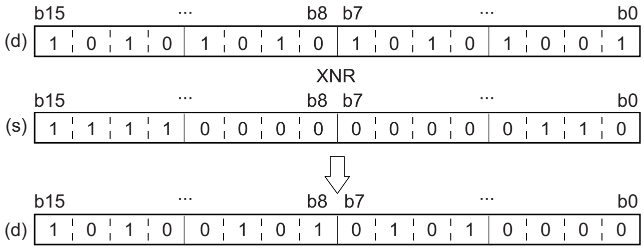
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an XNOR operation on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the 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.

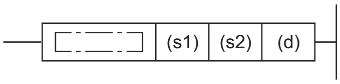
Operation error

There is no operation error.

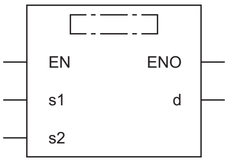
WXNR(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

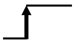
These instructions perform an XNOR operation on the two sets of 16-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
WXNR	
WXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1), (s2)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

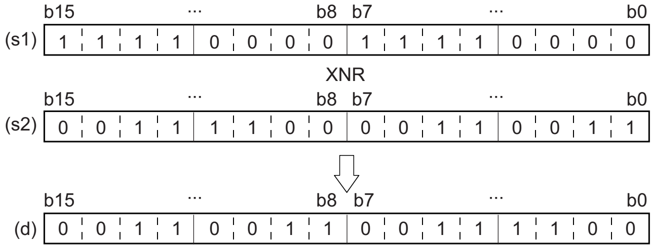
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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
(s2)	0	0	1	1	1	1	0	0	0	0	1	1	0	0	1	1
(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

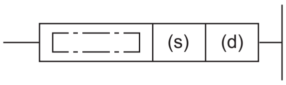
There is no operation error.

Performing an XNOR operation on 32-bit data

DXNR(P) [when two operands are set]

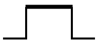
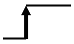
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions perform an XNOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 336 DXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 336 DXNR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DXNR	
DXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

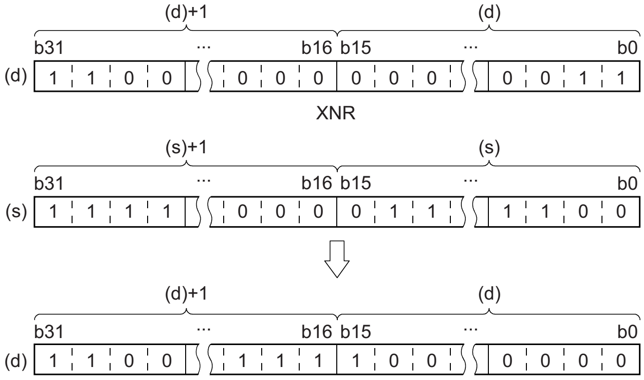
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform an XNOR operation on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the 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.

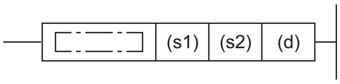
Operation error

There is no operation error.

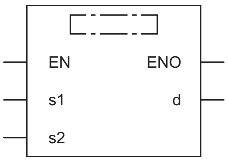
DXNR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

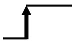
These instructions perform an XNOR operation on the two sets of 32-bit binary data specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DXNR	
DXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1), (s2)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

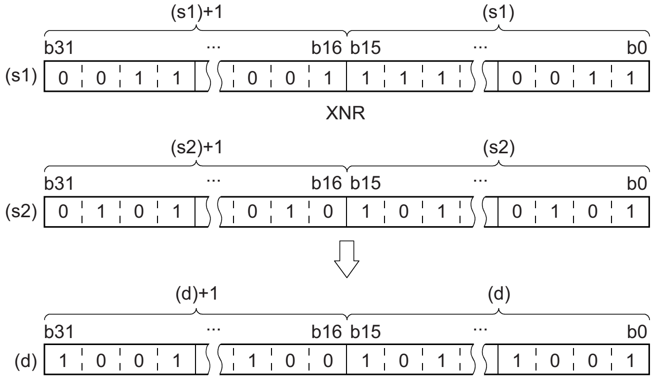
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

- These instructions perform 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 store 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

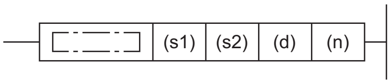
There is no operation error.

Performing an XNOR operation on 16-bit block data

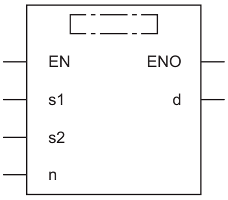
BKXNR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

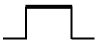
These instructions perform an XNOR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKXNR(EN,s1,s2,n,d); ENO:=BKXNRP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKXNR	
BKXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the logical operation data is stored	—	16-bit signed binary	ANY16[*1]
(s2)	Logical operation data or the start device where the logical operation data is stored	-32768 to 32767	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16[*1]
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

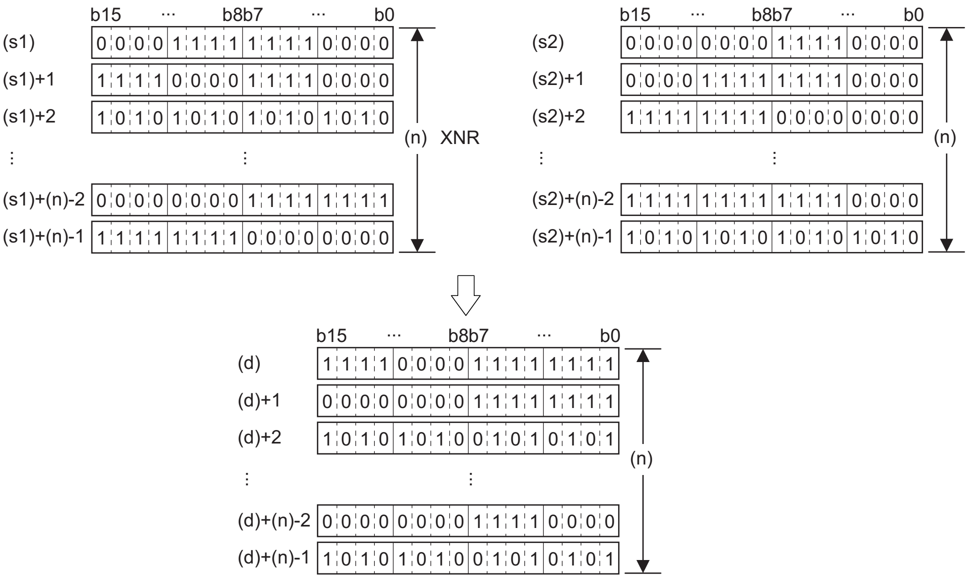
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(s2)[*1]	—	—	○	—	—	—	—	○	○	—	—	—
(d)[*1]	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an exclusive NOR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.

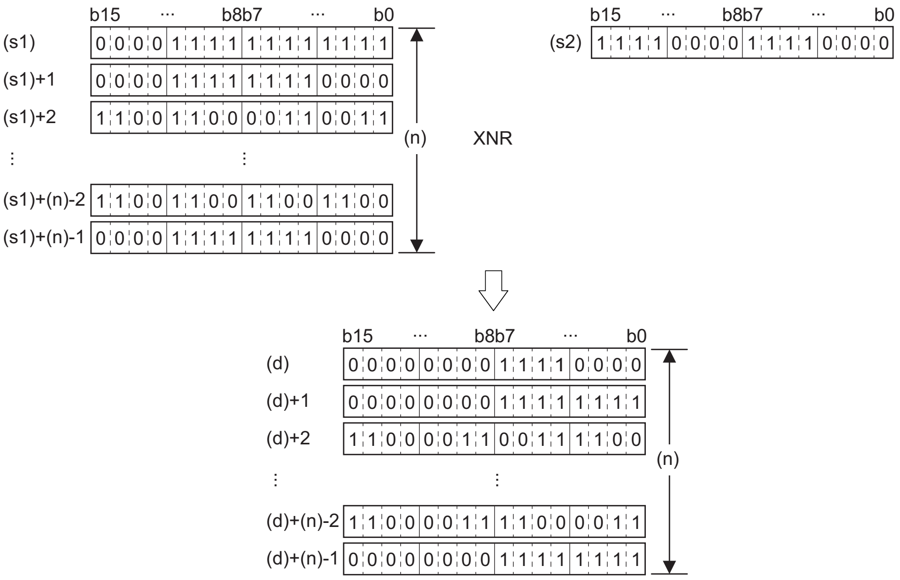


	b15 ··· b8b7 ··· b0
(s1)	0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
(s1)+1	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
(s1)+2	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
⋮	⋮
(s1)+(n)-2	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
(s1)+(n)-1	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

	b15 ··· b8b7 ··· b0
(s2)	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
(s2)+1	0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
(s2)+2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
⋮	⋮
(s2)+(n)-2	1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
(s2)+(n)-1	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

	b15 ··· b8b7 ··· b0
(d)	1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1
(d)+1	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
(d)+2	1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1
⋮	⋮
(d)+(n)-2	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
(d)+(n)-1	1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1

- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



	b15 ··· b8b7 ··· b0
(s1)	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
(s1)+1	0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
(s1)+2	1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1
⋮	⋮
(s1)+(n)-2	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
(s1)+(n)-1	0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0

	b15 ··· b8b7 ··· b0
(s2)	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0

	b15 ··· b8b7 ··· b0
(d)	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
(d)+1	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
(d)+2	1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0
⋮	⋮
(d)+(n)-2	1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
(d)+(n)-1	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

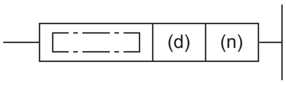
6.4 Bit Processing Instructions

Setting a bit in the word device

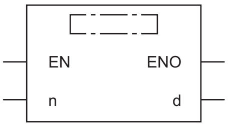
BSET(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

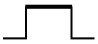
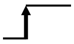
These instructions set the 'n'th bit in the specified word device to 1.

Ladder	ST
	ENO:=BSET(EN,n,d); ENO:=BSETP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSET	
BSETP	

Setting data

■Description, range, data type

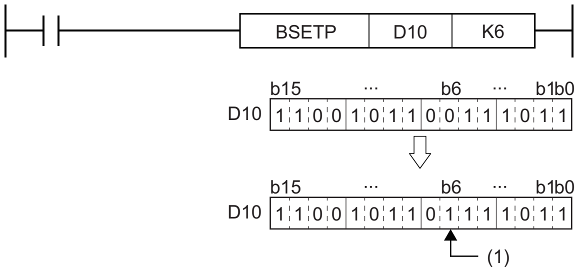
Operand	Description	Range	Data type	Data type (label)
(d)	Set target device	—	16-bit signed binary	ANY16
(n)	Set target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions set the 'n'th bit in the word device specified by (d) to 1.



(1) Set b6 of D10 to 1.

- If (n) exceeds 15, the instruction sets lower 4 bits of data.

Operation error

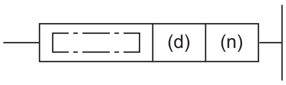
There is no operation error.

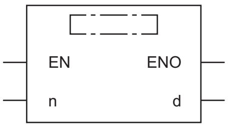
Resetting a bit in the word device

BRST(P)

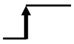
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions reset the 'n'th bit in the specified word device to 0.

Ladder	ST
	ENO:=BRST(EN,n,d); ENO:=BRSTP(EN,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BRST	
BRSTP	

Setting data

■Description, range, data type

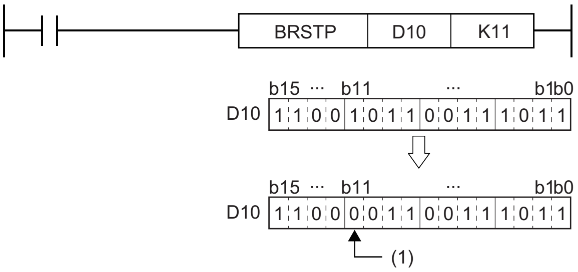
Operand	Description	Range	Data type	Data type (label)
(d)	Reset target device	—	16-bit signed binary	ANY16
(n)	Reset target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions reset the 'n'th bit in the word device specified by (d) to 0.



(1) Reset the b11 of D10 to 0.

- If (n) exceeds 15, the instruction sets lower 4 bits of data.

Operation error

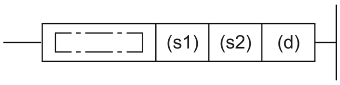
There is no operation error.

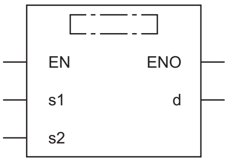
Performing a 16-bit test

TEST(P)

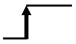
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

These instructions extract the 'n'th bit in the specified word device.

Ladder	ST
	ENO:=TEST(EN,s1,s2,d); ENO:=TESTP(EN,s1,s2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
TEST	
TESTP	

Setting data

■Description, range, data type

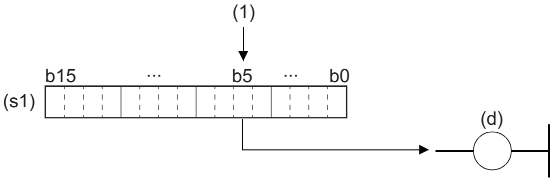
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	16-bit signed binary	ANY16
(s2)	Extract target bit position	0 to 15	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	○	○	○	○	—	—	○	—	—	—	—
(s2)	—	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—

Processing details

- These instructions extract the bit data at the position specified by (s2) of the word device specified by (s1), and write it to the bit device specified by (d).



(1) (s2) bit (When (s2)=5)

- The bit device specified by (d) turns off when the extracted bit data is 0 and turns on when the bit data is 1.
- Specify the bit position (0 to 15) of the word data in (s2). When 16 or a greater value is specified in (s2), the remainder of (s2)÷16 becomes the bit position.

Ex.

When (s2)=18: The remainder of 18÷16 is 2, and therefore the data in bit 2 will be extracted.

Operation error

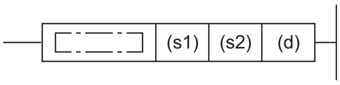
There is no operation error.

Performing a 32-bit test

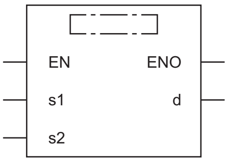
DTEST(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

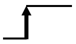
These instructions extract the 'n'th bit in the specified double-word device.

Ladder	ST
	ENO:=DTEST(EN,s1,s2,d); ENO:=DTESTP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DTEST	
DTESTP	

Setting data

■Description, range, data type

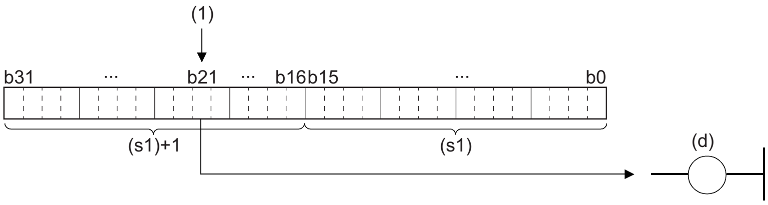
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	32-bit signed binary	ANY32
(s2)	Extract target bit position	0 to 31	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	○	○	○	○	○	○	○	—	—	—	—
(s2)	—	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—

Processing details

- These instructions extract the bit data at the position specified by (s2) of the double-word device specified by (s1), and write it to the bit device specified by (d).



(1) (s2) bit (When (s2)=21)

- The bit device specified by (d) turns off when the extracted bit data is 0 and turns on when the bit data is 1.
- Specify the bit position (0 to 31) of the double-word data in (s2). When 32 or a greater value is specified in (s2), the remainder of (s2)÷32 becomes the bit position.

Ex.

When (s2)=34: The remainder of 34÷32 is 2, and therefore the data in bit 2 will be extracted.

Operation error

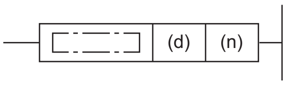
There is no operation error.

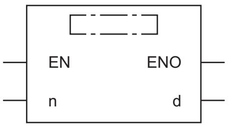
Batch-resetting bit devices

BKRST(P)

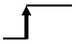
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions reset the (n) points of bit devices starting from the bit device specified.

Ladder	ST
	ENO:=BKRST(EN,n,d); ENO:=BKRSTP(EN,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BKRST	
BKRSTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to be reset	—	Bit	ANY_BOOL
(n)	Number of reset target devices	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	—	○	—	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions reset the (n) points of bit devices starting from the bit device specified by (d).
- The following table lists the reset status of the bit devices.

Device	Status
Annunciator (F)	• The (n) points of data starting from the annunciator (F) number in the device specified by (d) turn off. • The annunciator numbers that turned off are deleted from SD64 to SD79, and the remaining data are compressed forward. • The number of annunciators stored in SD64 to SD79 is stored in SD63.
Timer (T), retentive timer (ST), counter (C), long timer (LT), long retentive timer (LST), long counter (LC)	• The (n) points, starting from the timer (T), retentive timer (ST), counter (C), long timer (LT), long retentive timer (LST), or long counter (LC) number in the device specified by (d), of current values are reset to 0, and the (n) points of coils and contacts are turned off.
Bit devices other than the above	• The (n) points, starting from the device specified by (d), of coils and contacts are turned off.

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

Operation error

There is no operation error.

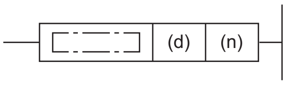
6.5 Shift Instructions

Shifting 16-bit binary data to the right by n bit(s)

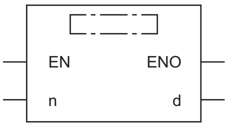
SFR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

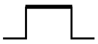
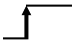
These instructions shift the 16-bit binary data in the specified device to the right.

Ladder	ST
	ENO:=SFR(EN,n,d); ENO:=SFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFR	
SFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	16-bit signed binary	ANY16
(n)	Number of shifts	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

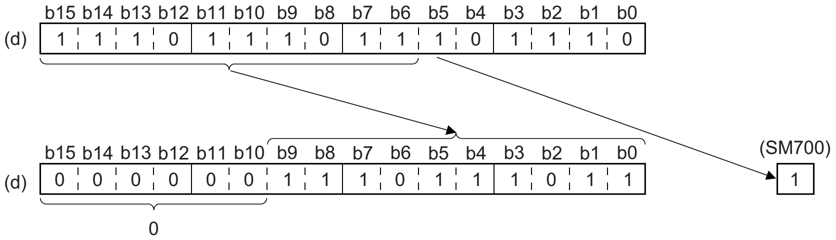
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions shift the 16-bit binary data in the device specified by (d) to the right by (n) bit(s). The (n) bit(s) from the most significant bit is/are filled with 0(s).

Ex.

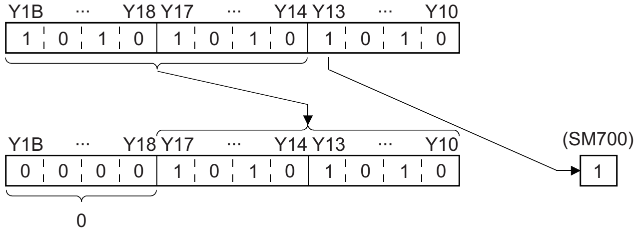
When (n)=6



- When (d) is a bit device, bits are shifted to the right within the device range specified by digit specification.

Ex.

When (n)=4



- The number of bits actually to be shifted is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 8, 7 bits are shifted because 15 divided by 8 equals 1 with a remainder of 7.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are shifted by the remainder value of n÷16. For example, when (n) is 18, 2 bits are shifted to the right because 18 divided by 16 equals 1 with a remainder of 2.

Operation error

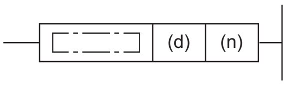
There is no operation error.

Shifting 16-bit binary data to the left by n bit(s)

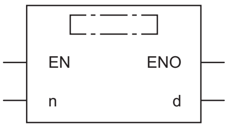
SFL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions shift the 16-bit binary data in the specified device to the left.

Ladder	ST
	ENO:=SFL(EN,n,d); ENO:=SFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFL	
SFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	16-bit signed binary	ANY16
(n)	Number of shifts	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

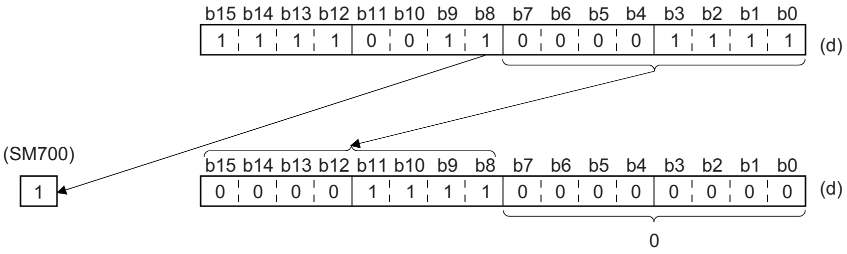
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- This instruction shifts the 16-bit binary data in the device specified by (d) to the left by (n) bit(s). The (n) bit(s) from the least significant bit is/are filled with 0(s).

Ex.

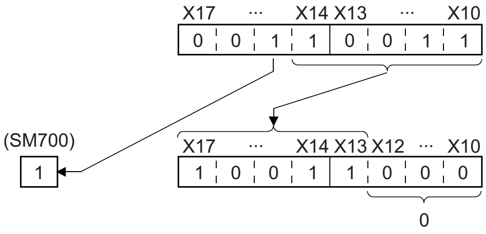
When (n)=8



- When (d) is a bit device, bits are shifted to the left within the device range specified by digit specification.

Ex.

When (n)=5



- The number of bits actually to be shifted is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 8, 7 bits are shifted because 15 divided by 8 equals 1 with a remainder of 7.
- Specify any value between 0 and 15 for (n). If a value 16 or larger is specified, the value is shifted by the remainder value of n÷16 to the left. For example, when (n) is 18, 2 bits are shifted to the left because 18 divided by 16 equals 1 with a remainder of 2.

Operation error

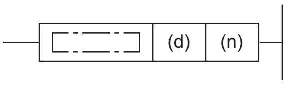
There is no operation error.

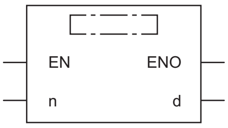
Shifting n-bit data to the right by one bit

BSFR(P)

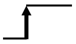
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions shift the n points of data starting from the specified device to the right by one bit.

Ladder	ST
	ENO:=BSFR(EN,n,d); ENO:=BSFRP(EN,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BSFR	
BSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n)	Number of bits to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

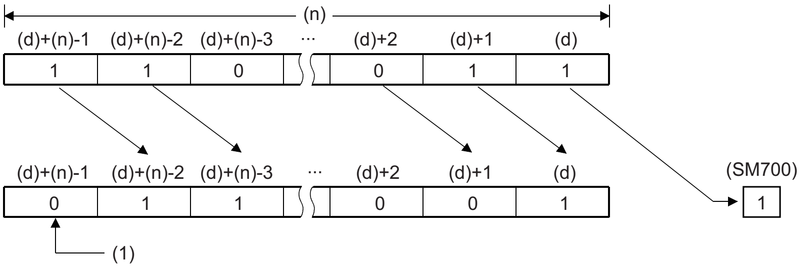
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[1]	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

[1] T, C, and ST cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the right by one bit.



(1) Filled with 0.

Operation error

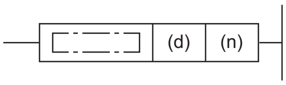
There is no operation error.

Shifting n-bit data to the left by one bit

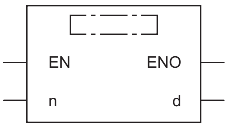
BSFL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions shift the n points of data starting from the specified device to the left by one bit.

Ladder	ST
	ENO:=BSFL(EN,n,d); ENO:=BSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSFL	
BSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n)	Number of bits to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

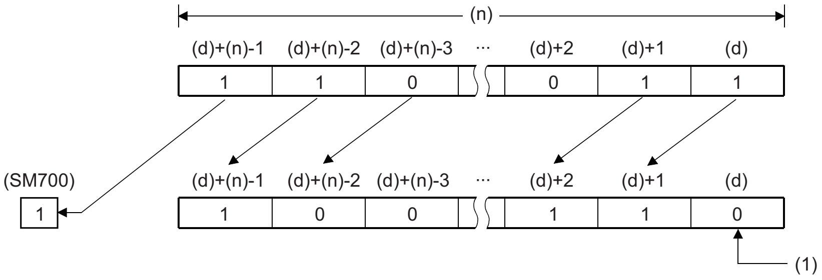
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 T, C, and ST cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the left by one bit.



(1) Filled with 0

Operation error

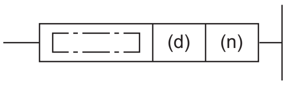
There is no operation error.

Shifting n-word data to the right by one word

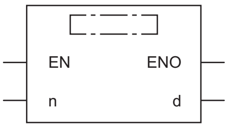
DSFR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions shift the n points of data starting from the specified device to the right by one word.

Ladder	ST
	ENO:=DSFR(EN,n,d); ENO:=DSFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSFR	
DSFRP	

Setting data

■Description, range, data type

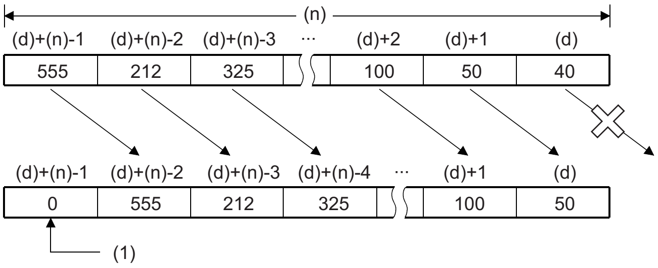
Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n)	Number of devices to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the right by one word.



(1) Filled with 0.

Operation error

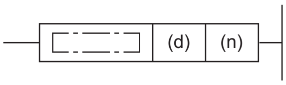
There is no operation error.

Shifting n-word data to the left by one word

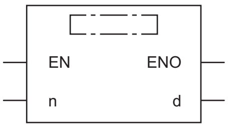
DSFL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions shift the n points of data starting from the specified device to the left by one word.

Ladder	ST
	ENO:=DSFL(EN,n,d); ENO:=DSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSFL	
DSFLP	

Setting data

■Description, range, data type

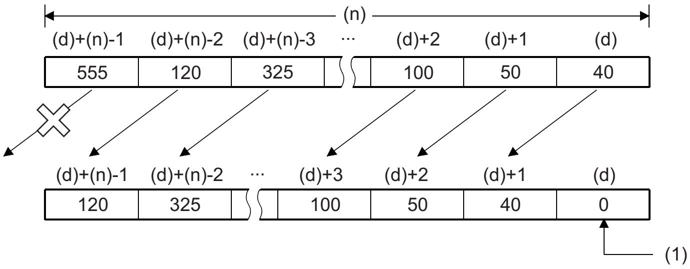
Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n)	Number of devices to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the left by one word.



(1) Filled with 0.

Operation error

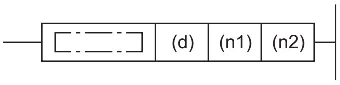
There is no operation error.

Shifting n-bit data to the right by n bit(s)

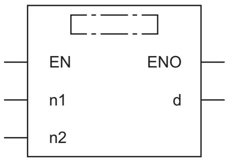
SFTBR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions shift the n-bit data starting from the specified device to the right by n bit(s).

Ladder	ST
	ENO:=SFTBR(EN,n1,n2,d); ENO:=SFTBRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTBR	
SFTBRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n1)	Number of bits to be shifted	0 to 64	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[*1]	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—

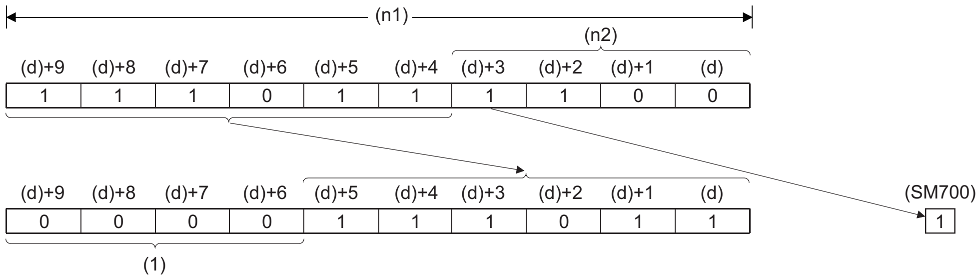
[*1] T, C, and ST cannot be used.

Processing details

- These instructions shift the (n1) bit(s) of data starting from the specified device to the right by (n2) bit(s).

Ex.

When (n1)=10 and (n2)=4



(1) Filled with 0s.

- Specify (n1) and (n2) so that the following condition is satisfied: (n1)>(n2). In the case of (n1)≤(n2), data is shifted by the value of the remainder of (n2)÷(n1). However, if the remainder value is 0, no processing is performed.
- Specify (n1) within the range of 1 to 64.
- The (n2) bit(s) from the most significant bit is/are filled with 0(s). In the case of (n1)<(n2), the bits are filled with 0s by the value of the remainder of (n2)÷(n1).
- If (n1) or (n2) is 0, no processing is performed.

Operation error

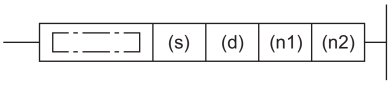
Error code (SD0)	Description
3405H	The value specified by (n1) is out of the range, 0 to 64.

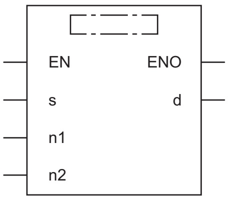
SFTR(P)

RnCPU RnENCPU ✕RnPCPU (Process) ✕RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

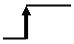
- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

These instructions shift the (n2) bit(s) of area to the right within the (n1) bits of data area starting from the specified device.

Ladder	ST
	ENO:=SFTR(EN,s,n1,n2,d); ENO:=SFTRP(EN,s,n1,n2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
SFTR	
SFTRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Bit	ANY_BOOL
(d)	Shift target start device	—	Bit	ANY_BOOL
(n1)[1]	Data length of shift data	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)[1]	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○[1]	—	—	—	—	○	○[2]	—	—	—
(d)	○	—	○[1]	—	—	—	—	○	—	—	—	—
(n1)	○[3]	○	○	○	○	—	—	○	○	—	—	—
(n2)	○[3]	○	○	○	○	—	—	○	○	—	—	—

[1] T, ST, C, and FD cannot be used.

[2] Only 0 or 1 can be used.

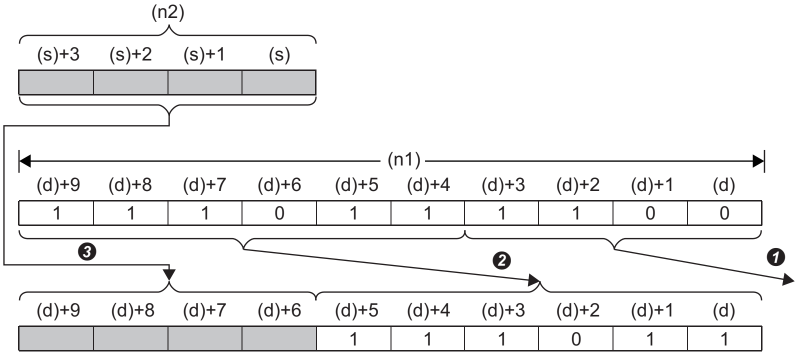
[3] FX and FY cannot be used.

Processing details

- These instructions shift the (n2) bit(s) of area to the right within the (n1) bits of data area starting from the device specified by (d). After the shift, (n2) points of area from (s) are set into (n2) points of area from (d)+(n1-n2).
- When constant 0 is specified for (s), 0s are stored in (n2) bits from (d)+(n1-n2) after the shift.
- When constant 1 is specified for (s), 1s are stored in (n2) bits from (d)+(n1-n2) after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=10 and (n2)=4



❶Overflow (data to be deleted)

❷(n2)-bit shift to the right

❸Copy

Operation error

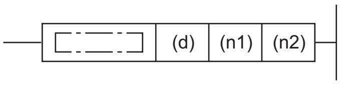
Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	A constant other than 0 or 1 is specified when the constant (s) is specified.
	The values specified in (n1) and (n2) are such that (n1)<(n2).

Shifting n-bit data to the left by n bit(s)

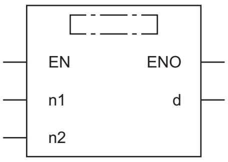
SFTBL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

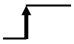
These instructions shift the n-bit data starting from the specified device to the left by n bit(s).

Ladder	ST
	ENO:=SFTBL(EN,n1,n2,d); ENO:=SFTBLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTBL	
SFTBLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n1)	Number of bits to be shifted	0 to 64	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○[*1]	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—

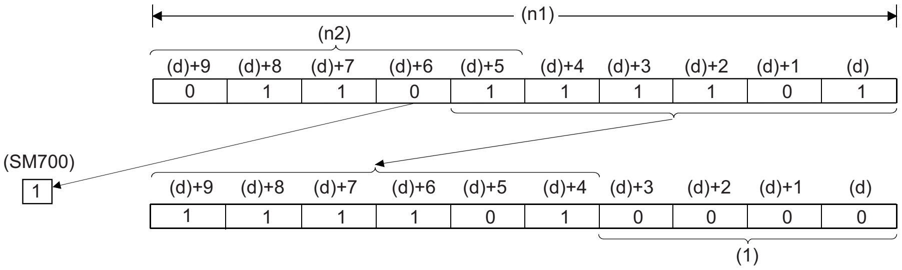
*1 T, C, and ST cannot be used.

Processing details

- These instructions shift the (n1) bit(s) of data starting from the specified device to the left by (n2) bit(s).

Ex.

When (n1)=10 and (n2)=4



(1) Filled with 0s.

- Specify (n1) and (n2) so that the following condition is satisfied: (n1)>(n2). In the case of (n1)≤(n2), data is shifted by the value of the remainder of (n2)÷(n1). However, if the remainder value is 0, no processing is performed.
- Specify (n1) within the range of 1 to 64.
- The (n2) bit(s) from the least significant bit is/are filled with 0(s). In the case of (n1)<(n2), the bits are filled with 0s by the value of the remainder of (n2)÷(n1).
- If (n1) or (n2) is 0, no processing is performed.

Operation error

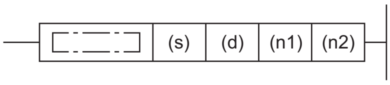
Error code (SD0)	Description
3405H	The value specified by (n1) is out of the range, 0 to 64.

SFTL(P)

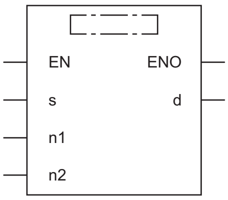
RnCPU **RnENCPU** ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

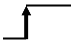
These instructions shift the (n2) bit(s) of area to the left within the (n1) bits of data area starting from the specified device.

Ladder	ST
	ENO:=SFTL(EN,s,n1,n2,d); ENO:=SFTLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTL	
SFTLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Bit	ANY_BOOL
(d)	Shift target start device	—	Bit	ANY_BOOL
(n1)*1	Data length of shift data	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○*1	—	—	—	—	○	○*2	—	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—
(n1)	○*3	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*3	○	○	○	○	—	—	○	○	—	—	—

*1 T, ST, C, and FD cannot be used.

*2 Only 0 or 1 can be used.

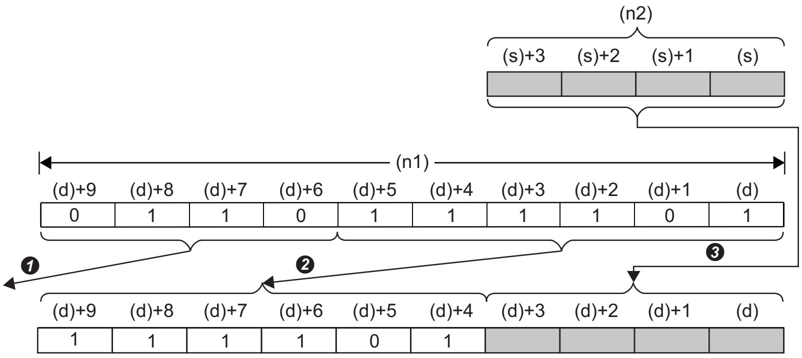
*3 FX and FY cannot be used.

Processing details

- These instructions shift the (n2) bit(s) of area to the left within the (n1) bits of data area starting from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d).
- When constant 0 is specified for (s), 0s are stored in (n2) bits from (d) after the shift.
- When constant 1 is specified for (s), 1s are stored in (n2) bits from (d) after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=10 and (n2)=4



❶Overflow (data to be deleted)

❷(n2)-bit shift to the left

❸Copy

Operation error

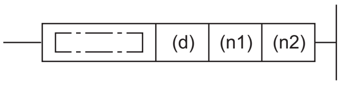
Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	A constant other than 0 or 1 is specified when the constant (s) is specified.
	The values specified in (n1) and (n2) are such that (n1)<(n2).

Shifting n-word data to the right by n word(s)

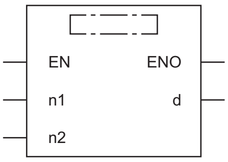
SFTWR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

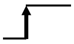
These instructions shift the n-word data starting from the specified device to the right by n word(s).

Ladder	ST
	ENO:=SFTWR(EN,n1,n2,d); ENO:=SFTWRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTWR	
SFTWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n1)	Number of devices to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

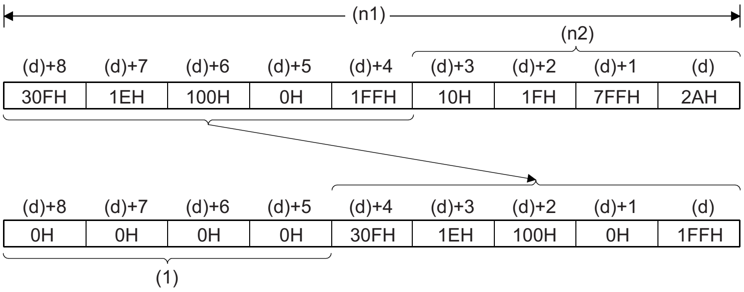
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions shift the (n1) word(s) of data starting from the specified device to the right by (n2) word(s).

Ex.

When (n1)=9 and (n2)=4



(d)+8	(d)+7	(d)+6	(d)+5	(d)+4	(d)+3	(d)+2	(d)+1	(d)
30FH	1EH	100H	0H	1FFH	10H	1FH	7FFH	2AH

(d)+8	(d)+7	(d)+6	(d)+5	(d)+4	(d)+3	(d)+2	(d)+1	(d)
0H	0H	0H	0H	30FH	1EH	100H	0H	1FFH

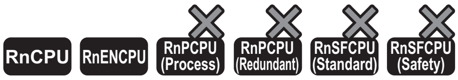
(1) Filled with 0Hs.

- The (n2) word(s) from the most significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0H, no processing is performed.
- In the case of (n1)≤(n2), (n1) words of data starting from the device specified by (d) become all 0Hs.

Operation error

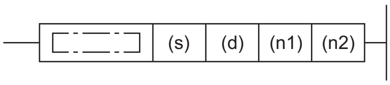
There is no operation error.

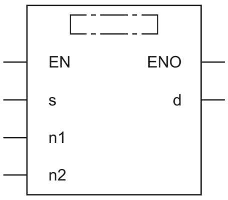
WSFR(P)



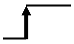
- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

These instructions shift the (n2) word(s) of area to the right within the (n1) words of data area starting from the specified device.

Ladder	ST
	ENO:=WSFR(EN,s,n1,n2,d); ENO:=WSFRP(EN,s,n1,n2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
WSFR	
WSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	16-bit unsigned binary	ANY16
(d)	Shift target start device	—	Word	ANY16
(n1)[*1]	Data length of shift data	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)[*1]	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[*1]	—	○	—	—	—	—	○	○	—	—	—
(d)	○[*1]	—	○	—	—	—	—	○	—	—	—	—
(n1)	○[*1]	○	○	○	○	—	—	○	○	—	—	—
(n2)	○[*1]	○	○	○	○	—	—	○	○	—	—	—

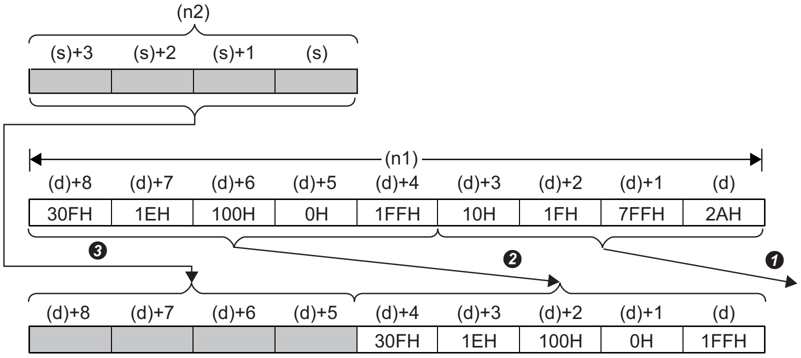
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n2) word(s) of area to the right within the (n1) words of data area starting from the device specified by (d). After the shift, (n2) points of area from (s) are set into (n2) points of area from (d)+(n1-n2).
- When a constant is specified for (s), the specified value is stored in (n2) point(s) of area in the device from (d) + (n1-n2) after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶Overflow (data to be deleted)

❷(n2)-word shift to the right

❸Copy

Operation error

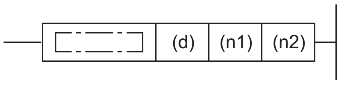
Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

Shifting n-word data to the left by n word(s)

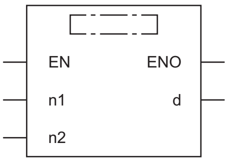
SFTWL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

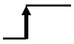
These instructions shift the n-word data starting from the specified device to the left by n word(s).

Ladder	ST
	ENO:=SFTWL(EN,n1,n2,d); ENO:=SFTWLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTWL	
SFTWLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n1)	Number of devices to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

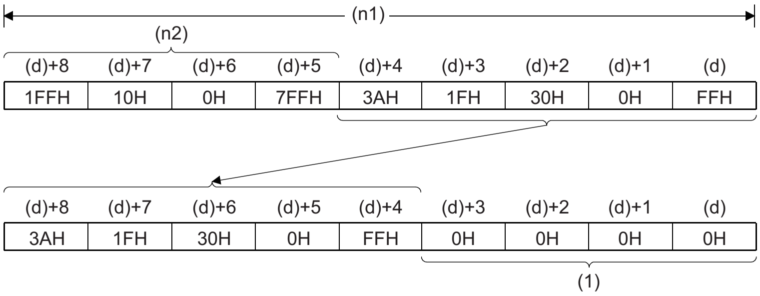
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions shift the (n1) word(s) of data starting from the specified device to the left by (n2) word(s).

Ex.

When (n1)=9 and (n2)=4



(1) Filled with 0Hs.

- The (n2) word(s) from the least significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0H, no processing is performed.
- In the case of (n1)≤(n2), (n1) words of data starting from the device specified by (d) become all 0Hs.

Operation error

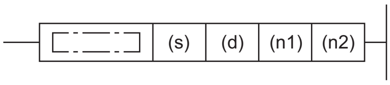
There is no operation error.

WSFL(P)

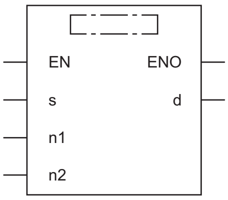
RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

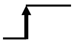
These instructions shift the (n2) word(s) of area to the left within the (n1) words of data area starting from the specified device.

Ladder	ST
	ENO:=WSFL(EN,s,n1,n2,d); ENO:=WSFLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WSFL	
WSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	16-bit unsigned binary	ANY16
(d)	Shift target start device	—	Word	ANY16
(n1)*1	Data length of shift data	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○*1	—	○	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

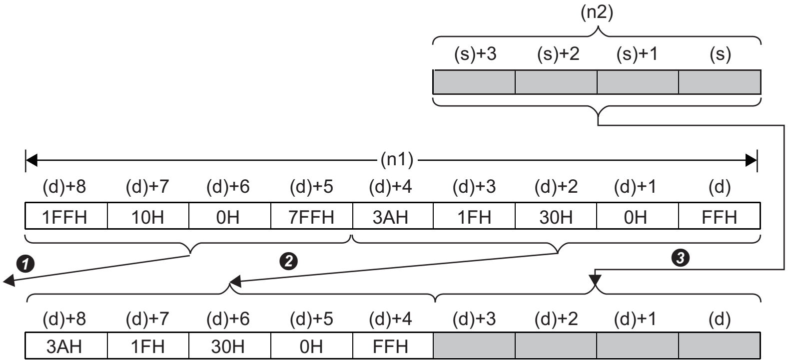
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n2) word(s) of area to the left within the (n1) words of data area starting from the device specified by (d). After the shift, (n2) points from (s) are set into (n2) points from (d).
- When a constant is specified for (s), the specified value is stored in (n2) point(s) of area in the device from (d) after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶Overflow (data to be deleted)

❷(n2)-word shift to the left

❸Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

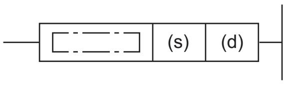
6.6 Data Conversion Instructions

Converting binary data to BCD 4-digit data

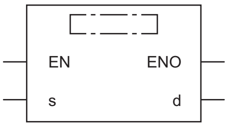
BCD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

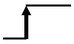
These instructions convert the specified 16-bit binary data to BCD 4-digit data.

Ladder	ST
	ENO:=BCD(EN,s,d); ENO:=BCDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BCD	
BCDP	

Setting data

■Description, range, data type

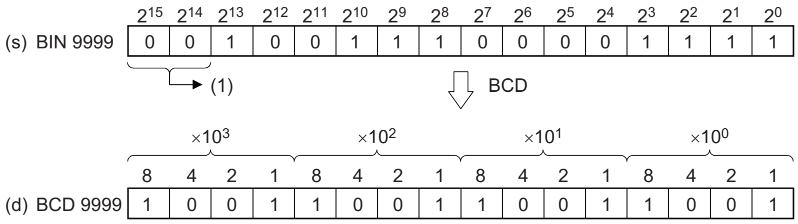
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 9999	16-bit signed binary	ANY16
(d)	Device for storing the converted BCD data	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	○	○	○	○	—	—	○	○	—	—	—
(d)	—	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary data (0 to 9999) in the device specified by (s) to BCD 4-digit data, and store the converted data in the device specified by (d).



(1) Set 0s.

Operation error

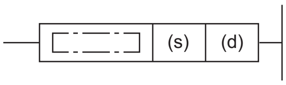
Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 9999.

Converting binary data to BCD 8-digit data

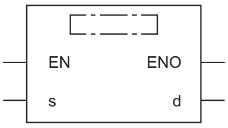
DBCD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

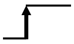
These instructions convert the specified 32-bit binary data to BCD 8-digit data.

Ladder	ST
	ENO:=DBCD(EN,s,d); ENO:=DBCDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCD	
DBCDP	

Setting data

■Description, range, data type

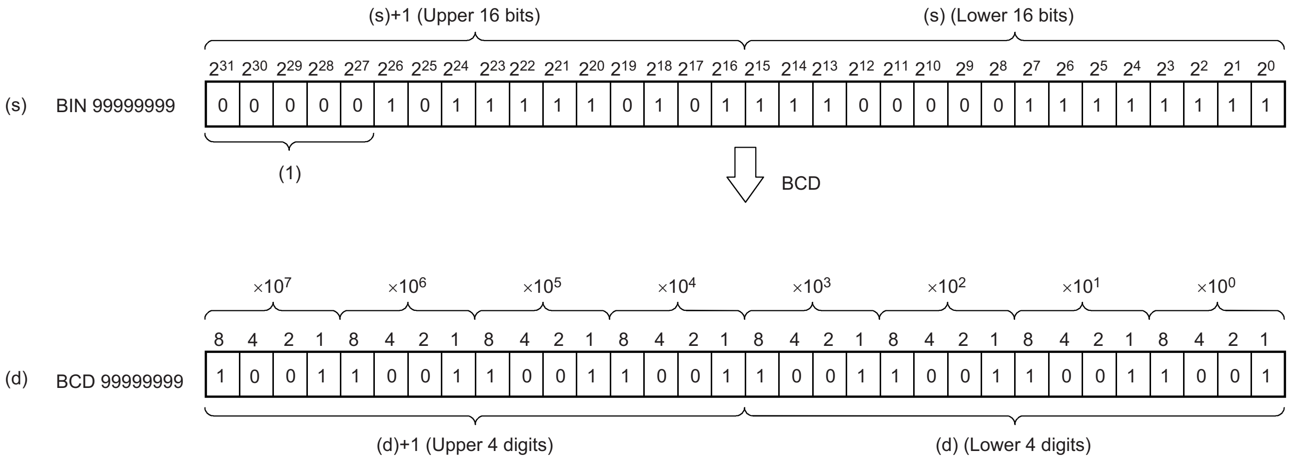
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 99999999	32-bit signed binary	ANY32
(d)	Start device for storing the converted BCD data	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary data (0 to 99999999) in the device specified by (s) to BCD 8-digit data, and store the converted data in the device specified by (d).



(1) Set 0s to the upper 5 bits.

Operation error

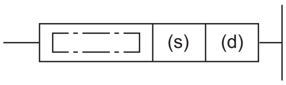
Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 99999999.

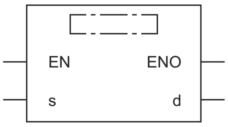
Converting BCD 4-digit data to 16-bit binary data

BIN(P)

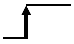
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✕

These instructions convert the specified BCD 4-digit data to 16-bit binary data.

Ladder	ST
	ENO:=BIN(EN,s,d); ENO:=BINP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BIN	
BINP	

Setting data

■Description, range, data type

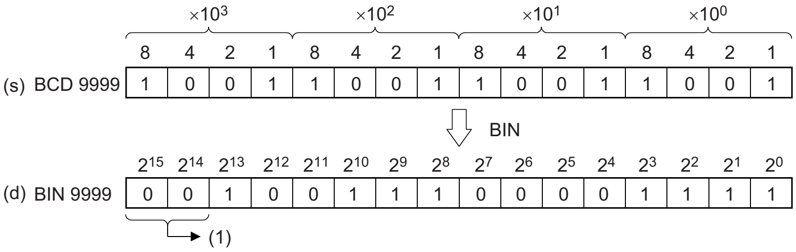
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the device where the BCD data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the BCD 4-digit data (0 to 9999) in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).



(1) Filled with 0s.

Operation error

Error code (SD0)	Description
3401H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).[*1]

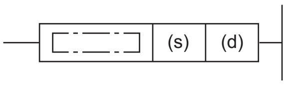
[*1] Turning on SM754 can prevent this error from being detected.
If the specified value is out of the valid range, the BIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The BIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

Converting BCD 8-digit data to 32-bit binary data

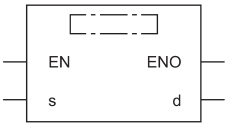
DBIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

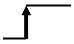
These instructions convert the specified BCD 8-digit data to 32-bit binary data.

Ladder	ST
	ENO:=DBIN(EN,s,d); ENO:=DBINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBIN	
DBINP	

Setting data

■Description, range, data type

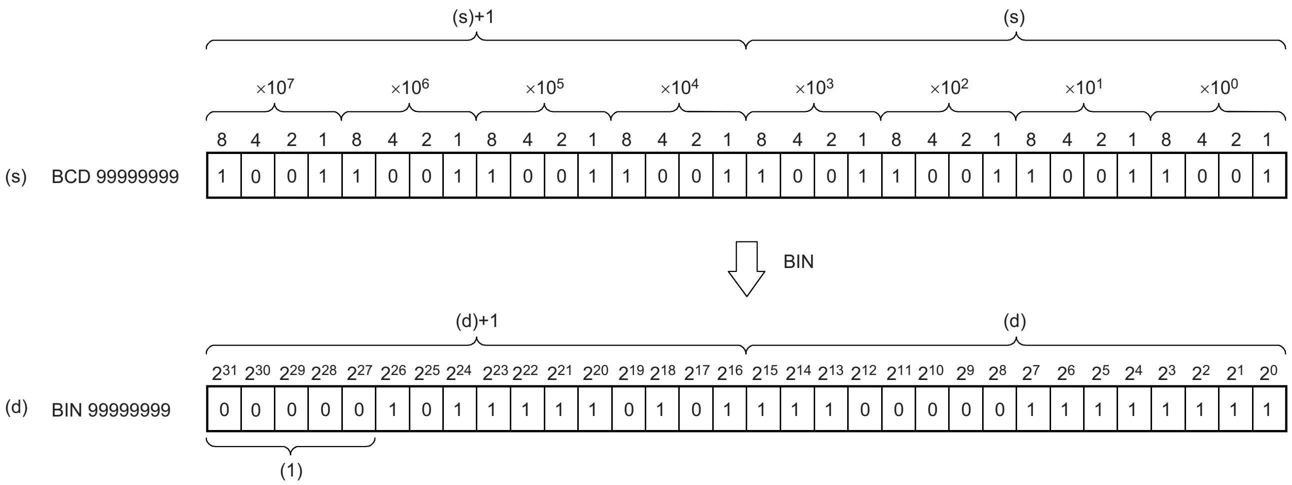
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the start device where the BCD data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the BCD 8-digit data (0 to 99999999) in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).



(1) Filled with 0s.

Operation error

Error code (SD0)	Description
3401H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).*1

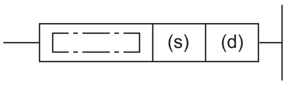
*1 Turning on SM754 can prevent this error from being detected.
If the specified value is out of the valid range, the DBIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The DBIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

Converting single-precision real number to 16-bit signed binary data

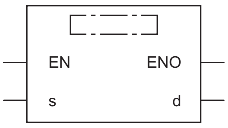
FLT2INT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✕

These instructions convert the specified single-precision real number to 16-bit signed binary data.

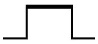
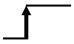
Ladder	ST[*1]
	ENO:=FLT2INT(EN,s,d); ENO:=FLT2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2INT	
FLT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	-32768 to 32767	Single-precision real number	ANYREAL_32
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

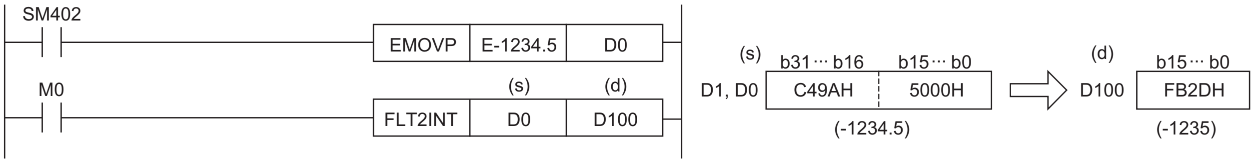
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



Operation error

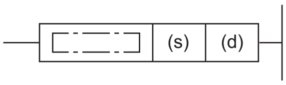
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: 0, $2^{-126} \leq \|(s)\| < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

Converting single-precision real number to 16-bit unsigned binary data

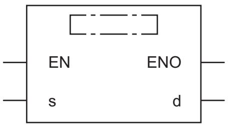
FLT2UINT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions convert the specified single-precision real number to 16-bit unsigned binary data.

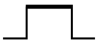
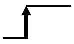
Ladder	ST[*1]
	ENO:=FLT2UINT(EN,s,d); ENO:=FLT2UINTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2UINT	
FLT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	0 to 65535	Single-precision real number	ANYREAL_32
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

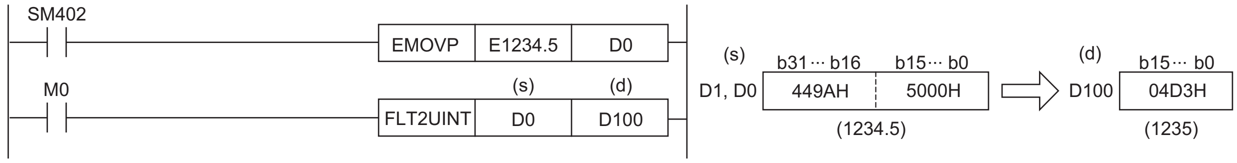
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.



Operation error

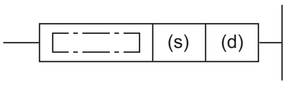
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: 0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

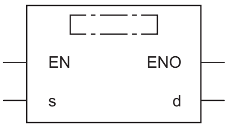
Converting single-precision real number to 32-bit signed binary data

FLT2DINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions convert the specified single-precision real number to 32-bit signed binary data.

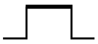
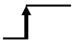
Ladder	ST[1]
	ENO:=FLT2DINT(EN,s,d); ENO:=FLT2DINTP(EN,s,d);

FBD/LD


---

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2DINT	
FLT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	-2147483648 to 2147483647	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

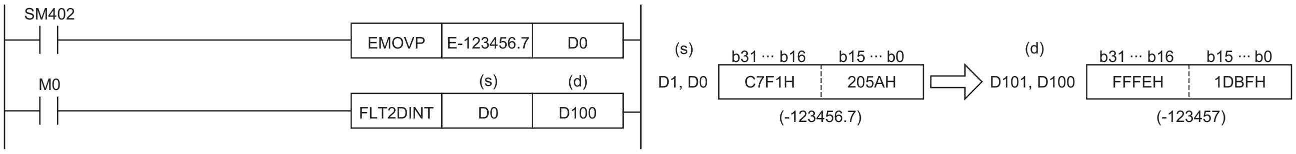
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 32-bit signed binary data, and stores the converted data in D100 and D101.



Operation error

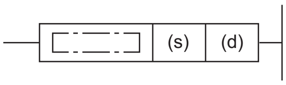
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: 0, $2^{-126} \leq \|(s)\| < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

Converting single-precision real number to 32-bit unsigned binary data

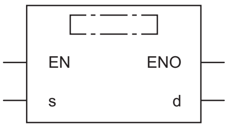
FLT2UDINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert the specified single-precision real number to 32-bit unsigned binary data.

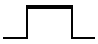
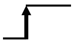
Ladder	ST[*1]
	ENO:=FLT2UDINT(EN,s,d); ENO:=FLT2UDINTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2UDINT	
FLT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	0 to 4294967295	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

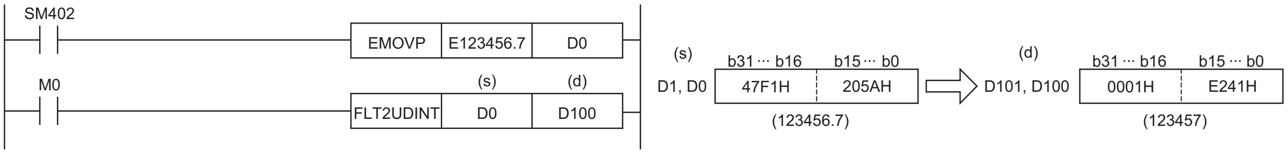
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



Operation error

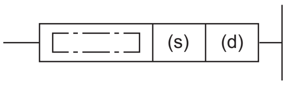
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

Converting double-precision real number to 16-bit signed binary data

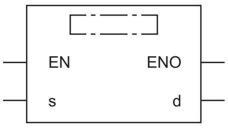
DBL2INT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕RnSFCPU (Safety)

These instructions convert the specified double-precision real number to 16-bit signed binary data.

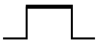
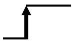
Ladder	ST[1]
	ENO:=DBL2INT(EN,s,d); ENO:=DBL2INTP(EN,s,d);

FBD/LD



[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2INT	
DBL2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	-32768 to 32767	Double-precision real number	ANYREAL_64
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

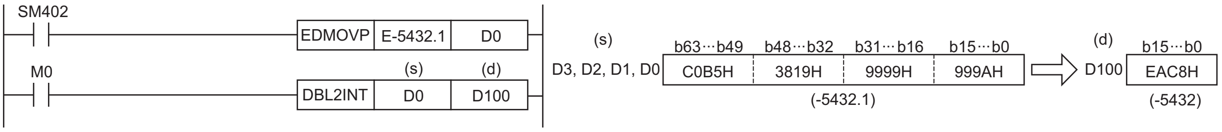
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 16-bit signed binary data, and stores the converted data in D100.



Operation error

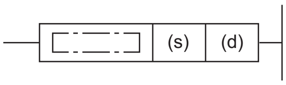
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

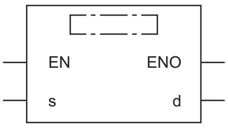
Converting double-precision real number to 16-bit unsigned binary data

DBL2UINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕RnSFCPU (Safety)

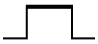
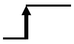
These instructions convert the specified double-precision real number to 16-bit unsigned binary data.

Ladder	ST[*1]
 (s) (d)	ENO:=DBL2UINT(EN,s,d); ENO:=DBL2UINTP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2UINT	
DBL2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	0 to 65535	Double-precision real number	ANYREAL_64
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

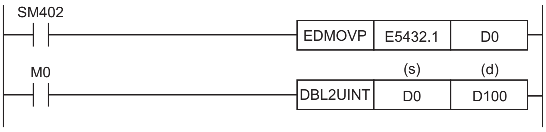
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

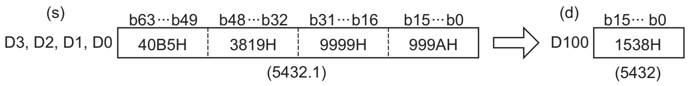
Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 16-bit unsigned binary data, and stores the converted data in D100.





Operation error

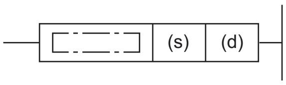
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq \|(s)\| < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

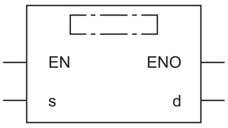
Converting double-precision real number to 32-bit signed binary data

DBL2DINT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

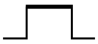
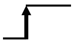
These instructions convert the specified double-precision real number to 32-bit signed binary data.

Ladder	ST[*1]
	ENO:=DBL2DINT(EN,s,d); ENO:=DBL2DINTP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2DINT	
DBL2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	-2147483648 to 2147483647	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

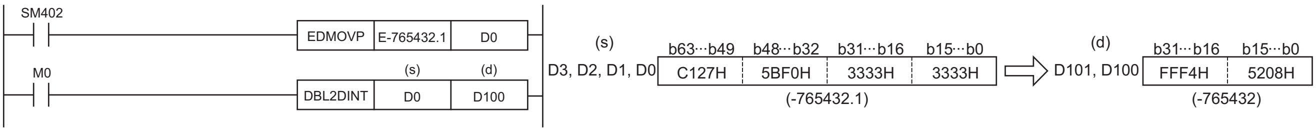
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 32-bit signed binary data, and stores the converted data in D100 and D101.

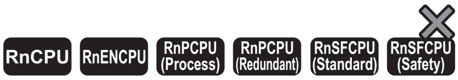


Operation error

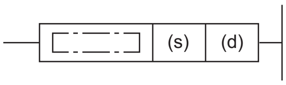
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq \lvert (s) \rvert < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

Converting double-precision real number to 32-bit unsigned binary data

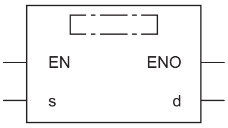
DBL2UDINT(P)



These instructions convert the specified double-precision real number to 32-bit unsigned binary data.

Ladder	ST[1]
	ENO:=DBL2UDINT(EN,s,d); ENO:=DBL2UDINTP(EN,s,d);

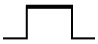
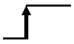
FBD/LD



---

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2UDINT	
DBL2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	0 to 4294967295	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

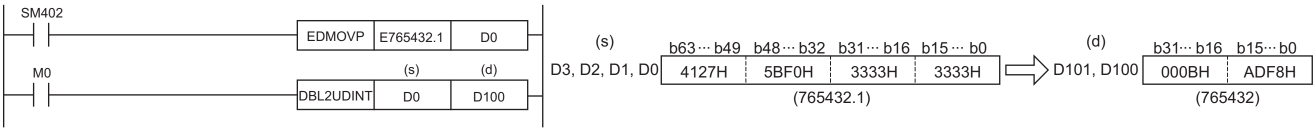
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 49 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



Operation error

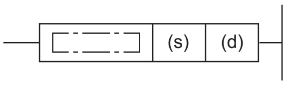
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq \|(s)\| < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

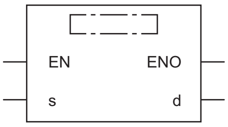
Converting 16-bit signed binary data to 16-bit unsigned binary data

INT2UINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

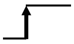
These instructions convert the specified 16-bit signed binary data to 16-bit unsigned binary data.

Ladder	ST[*1]
	ENO:=INT2UINT(EN,s,d); ENO:=INT2UINTP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2UINT	
INT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Label for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Point

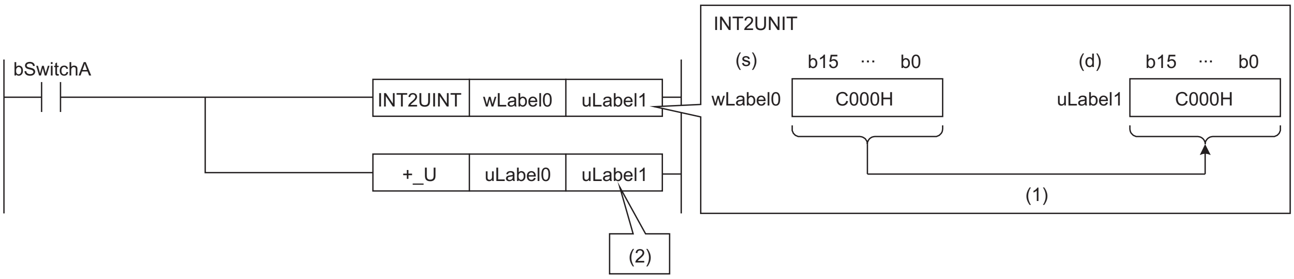
The INT2UINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand. In programming using devices, use of the INT2UINT(P) instruction is not required.

Processing details

- These instructions convert the 16-bit signed binary data (ANY16_S) in the label specified by (s) to 16-bit unsigned binary data (ANY16_U), and store the converted data in the label specified by (d).
- The following figure shows a program example using the INT2UINT(P) instruction.

Ex.

The +_U instruction requires ANY16_U to be specified by the operand, and therefore, before the +_U instruction is executed, the INT2UINT instruction is used to convert wLabel0 of ANY16_S to uLabel1 of ANY16_U.
The value in wLabel0 is stored in uLabel1 as is.



bSwitchA: Bit
wLabel0: Word [signed]
uLabel0, uLabel1: Word [unsigned]/bit string [16 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the +_U instruction, and the operation starts.

Operation error

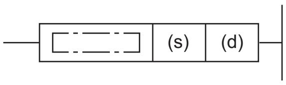
There is no operation error.

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

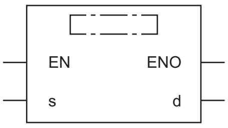
INT2DINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert the specified 16-bit signed binary data to 32-bit signed binary data.

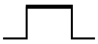
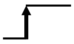
Ladder	ST[*1]
	ENO:=INT2DINT(EN,s,d); ENO:=INT2DINTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2DINT	
INT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

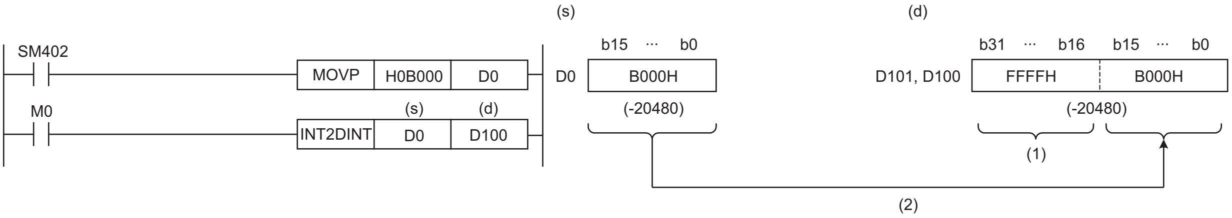
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 16-bit signed binary data stored in D0 to 32-bit signed binary data, and stores the converted data in D100 and D101.



(1) The most significant bit of data before conversion is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

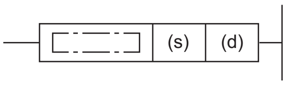
There is no operation error.

Converting 16-bit signed binary data to 32-bit unsigned binary data

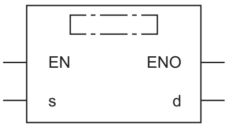
INT2UDINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert the specified 16-bit signed binary data to 32-bit unsigned binary data.

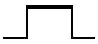
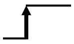
Ladder	ST[*1]
	ENO:=INT2UDINT(EN,s,d); ENO:=INT2UDINTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2UDINT	
INT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

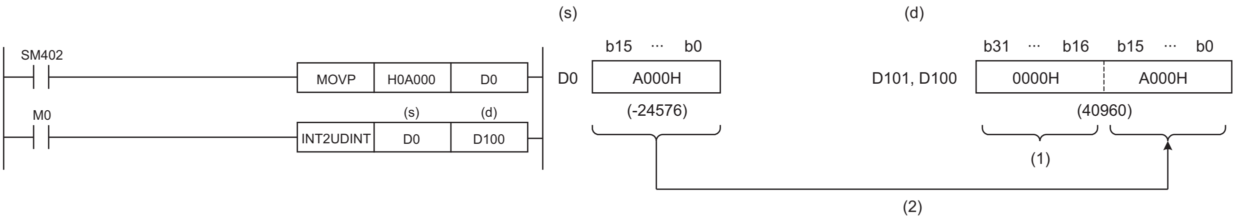
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 16-bit signed binary data stored in D0 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

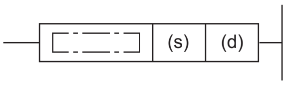
There is no operation error.

Converting 16-bit unsigned binary data to 16-bit signed binary data

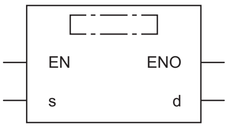
UINT2INT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions convert the specified 16-bit unsigned binary data to 16-bit signed binary data.

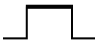
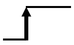
Ladder	ST[1]
	ENO:=UINT2INT(EN,s,d); ENO:=UINT2INTP(EN,s,d);

FBD/LD



[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2INT	
UINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Label for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Point

The UINT2INT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand. In programming using devices, use of the UINT2INT(P) instruction is not required.

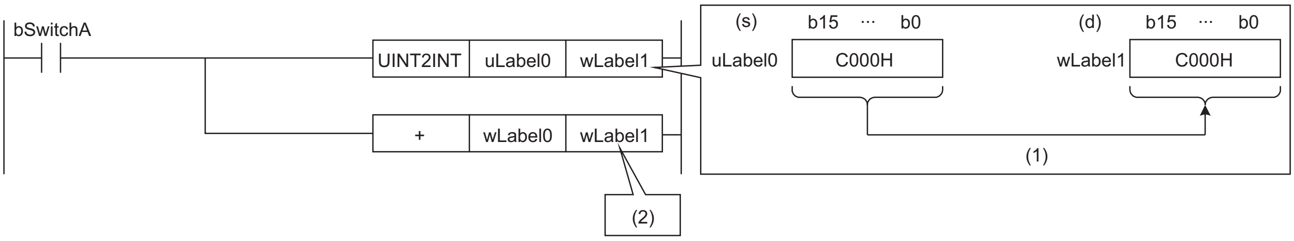
Processing details

- These instructions convert the 16-bit signed binary data (ANY16_U) in the label specified by (s) to 16-bit unsigned binary data (ANY16_S), and store the converted data in the label specified by (d).
- The following figure shows a program example using the UINT2INT(P) instruction.

Ex.

The + instruction requires ANY16_S to be specified by the operand, and therefore, before the + instruction is executed, the UINT2INT instruction is used to convert uLabel0 of ANY16_U to wLabel1 of ANY16_S.

The value in uLabel0 is stored in wLabel1 as is.



bSwitchA: Bit
wLabel0, wLabel1: Word [signed]
uLabel0: Word [unsigned]/bit string [16 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the + instruction, and the operation starts.

Operation error

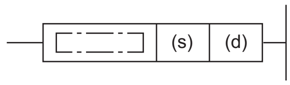
There is no operation error.

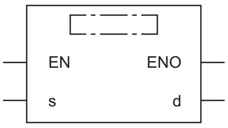
Converting 16-bit unsigned binary data to 32-bit signed binary data

UINT2DINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

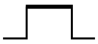
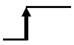
These instructions convert the specified 16-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST[*1]
	ENO:=UINT2DINT(EN,s,d); ENO:=UINT2DINTP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2DINT	
UINT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

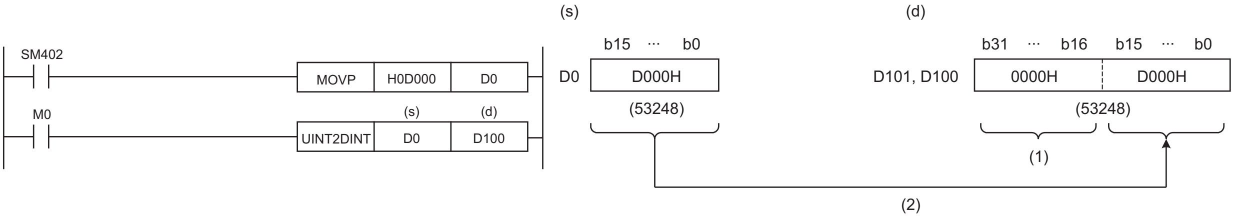
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 16-bit unsigned binary data stored in D0 to 32-bit signed binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

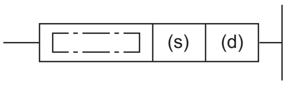
There is no operation error.

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

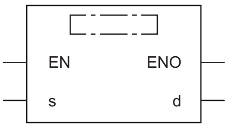
UINT2UDINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert the specified 16-bit unsigned binary data to 32-bit unsigned binary data.

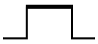
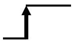
Ladder	ST[*1]
	ENO:=UINT2UDINT(EN,s,d); ENO:=UINT2UDINTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2UDINT	
UINT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

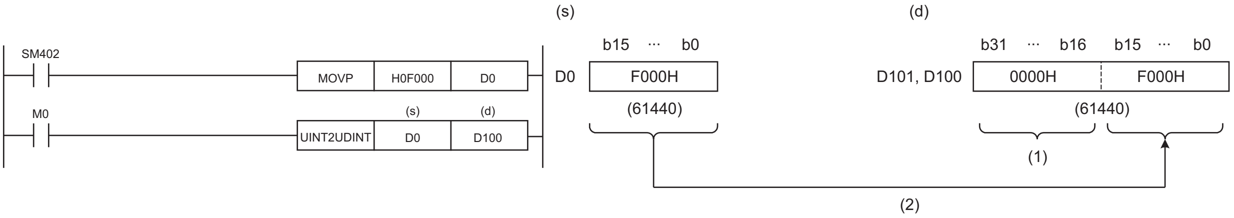
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 16-bit unsigned binary data stored in D0 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

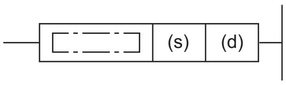
There is no operation error.

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

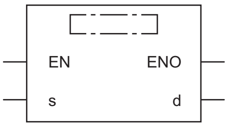
DINT2INT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions convert the specified 32-bit signed binary data to 16-bit signed binary data.

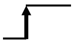
Ladder	ST[*1]
	ENO:=DINT2INT(EN,s,d); ENO:=DINT2INTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2INT	
DINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	-32768 to 32767	32-bit signed binary	ANY32_S
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

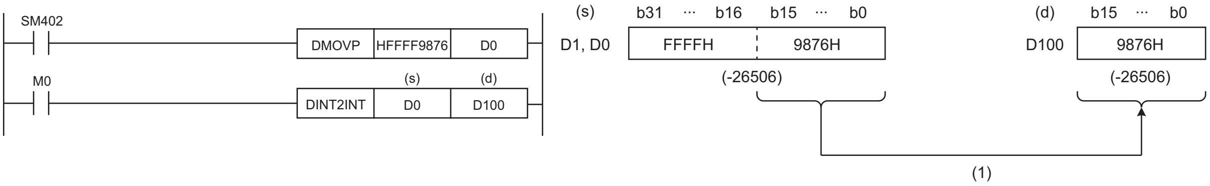
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 32-bit signed binary data stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

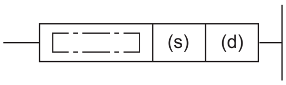
Error code (SD0)	Description
3401H	The 32-bit signed binary data in the device specified by (s) is out of the range, -32768 to 32767.

Converting 32-bit signed binary data to 16-bit unsigned binary data

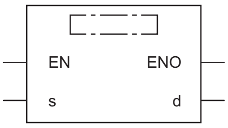
DINT2UINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert the specified 32-bit signed binary data to 16-bit unsigned binary data.

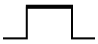
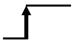
Ladder	ST[1]
	ENO:=DINT2UINT(EN,s,d); ENO:=DINT2UINTP(EN,s,d);

FBD/LD



[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2UINT	
DINT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

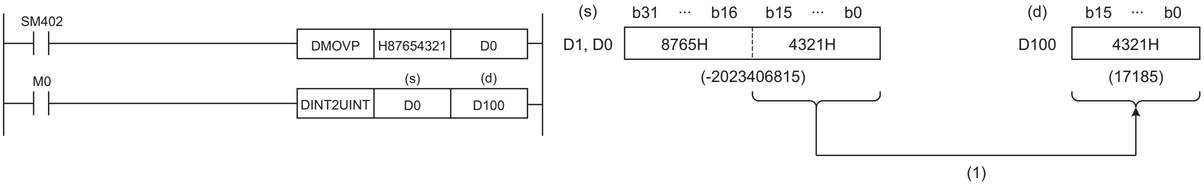
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 32-bit signed binary data stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

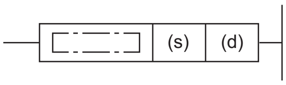
There is no operation error.

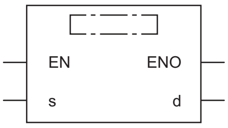
Converting 32-bit signed binary data to 32-bit unsigned binary data

DINT2UDINT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

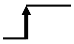
These instructions convert the specified 32-bit signed binary data to 32-bit unsigned binary data.

Ladder	ST[1]
	ENO:=DINT2UDINT(EN,s,d); ENO:=DINT2UDINTP(EN,s,d);

FBD/LD


[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2UDINT	
DINT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Label for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Point

The DINT2UDINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand.
In programming using devices, use of the DINT2UDINT(P) instruction is not required.

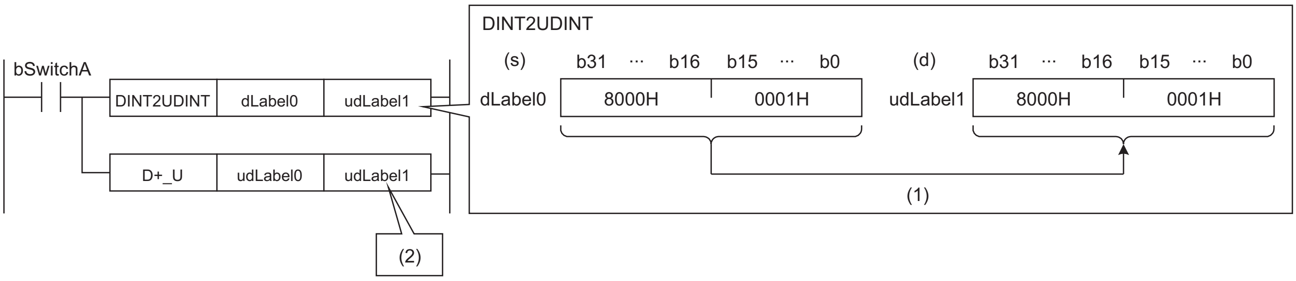
Processing details

- These instructions convert the 32-bit signed binary data (ANY32_S) in the label specified by (s) to 32-bit unsigned binary data (ANY32_U), and store the converted data in the label specified by (d).
- The following figure shows a program example using the DINT2UDINT(P) instruction.

Ex.

The D+_U instruction requires ANY32_U to be specified by the operand, and therefore, before the D+_U instruction is executed, the DINT2UDINT instruction is used to convert dLabel0 of ANY32_S to udLabel1 of ANY32_U.

The value in dLabel0 is stored in udLabel1 as is.



bSwitchA: Bit
dLabel0: Double word [signed]
udLabel0, udLabel1: Double word [unsigned]/bit string [32 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the D+_U instruction, and the operation starts.

Operation error

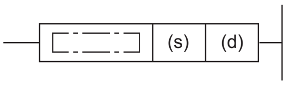
There is no operation error.

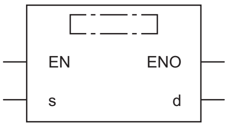
Converting 32-bit unsigned binary data to 16-bit signed binary data

UDINT2INT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

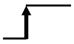
These instructions convert the specified 32-bit unsigned binary data to 16-bit signed binary data.

Ladder	ST[1]
	ENO:=UDINT2INT(EN,s,d); ENO:=UDINT2INTP(EN,s,d);

FBD/LD


[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2INT	
UDINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

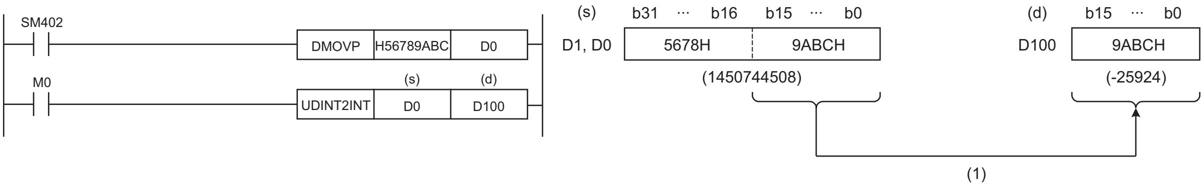
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

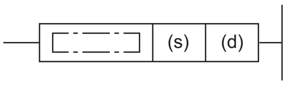
There is no operation error.

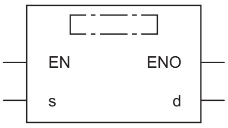
Converting 32-bit unsigned binary data to 16-bit unsigned binary data

UDINT2UINT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

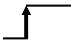
These instructions convert the specified 32-bit unsigned binary data to 16-bit unsigned binary data.

Ladder	ST[*1]
	ENO:=UDINT2UINT(EN,s,d); ENO:=UDINT2UINTP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2UINT	
UDINT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

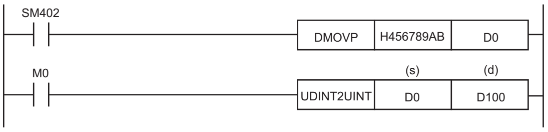
■Applicable devices

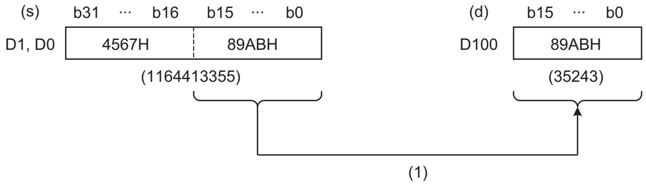
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

The following program example converts, when M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.





(1) Data before conversion is stored in the lower 16 bits.

Operation error

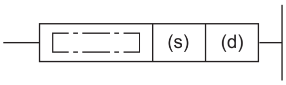
There is no operation error.

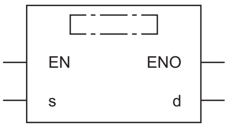
Converting 32-bit unsigned binary data to 32-bit signed binary data

UDINT2DINT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

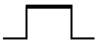
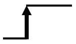
These instructions convert the specified 32-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST[*1]
	ENO:=UDINT2DINT(EN,s,d); ENO:=UDINT2DINTP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2DINT	
UDINT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Label for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Point

The UDINT2DINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand.

In programming using devices, use of the UDINT2DINT(P) instruction is not required.

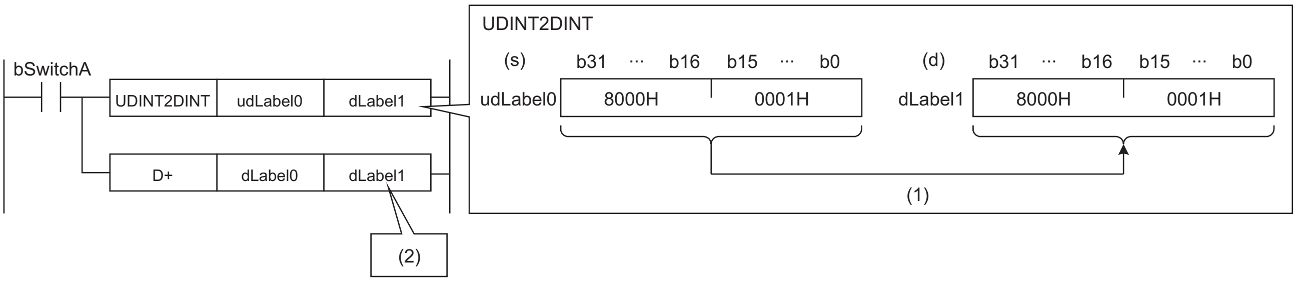
Processing details

- These instructions convert the 32-bit signed binary data (ANY32_U) in the label specified by (s) to 32-bit unsigned binary data (ANY32_S), and store the converted data in the label specified by (d).
- The following figure shows a program example using the UDINT2DINT(P) instruction.

Ex.

The D+ instruction requires ANY32_S to be specified by the operand, and therefore, before the D+ instruction is executed, the UDINT2DINT instruction is used to convert udLabel0 of ANY32_U to dLabel1 of ANY32_S.

The value in udLabel0 is stored in dLabel1 as is.



bSwitchA: Bit

dLabel0, dLabel1: Double word [signed]

udLabel0: Double word [unsigned]/bit string [32 bits]

(1) The value is stored as is.

(2) The data type of the value is converted to the one of the operand in the D+ instruction, and the operation starts.

Operation error

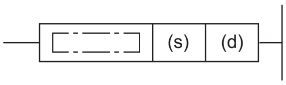
There is no operation error.

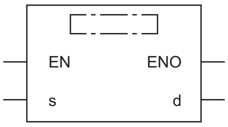
Converting 16-bit binary data to Gray code data

GRY(P)(_U)

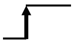
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✗

These instructions convert the specified 16-bit binary data to 16-bit binary Gray code data.

Ladder	ST	
	ENO:=GRY(EN,s,d); ENO:=GRYP(EN,s,d);	ENO:=GRY_U(EN,s,d); ENO:=GRYP_U(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
GRY GRY_U	
GRYP GRYP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	GRY(P)	Binary data or the device where the binary data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GRY(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GRY(P)	Device for storing the converted Gray code data	—	16-bit signed binary	ANY16_S
	GRY(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary data in the device specified by (s) to 16-bit binary Gray code data, and store the converted data in the device specified by (d).

		b15							···								b0
(s) BIN	1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

⇩

		b15							···								b0
(d)	1234	0	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1

Operation error

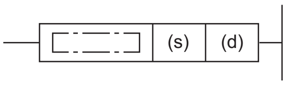
Error code (SD0)	Description
3401H	When the GRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 32767.

Converting 32-bit binary data to Gray code data

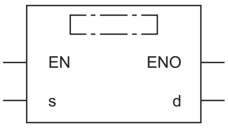
DGRY(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

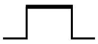
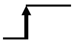
These instructions convert the specified 32-bit binary data to 32-bit binary Gray code data.

Ladder	ST	
	ENO:=DGRY(EN,s,d); ENO:=DGRYP(EN,s,d);	ENO:=DGRY_U(EN,s,d); ENO:=DGRYP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DGRY DGRY_U	
DGRYP DGRYP_U	

Setting data

■Description, range, data type

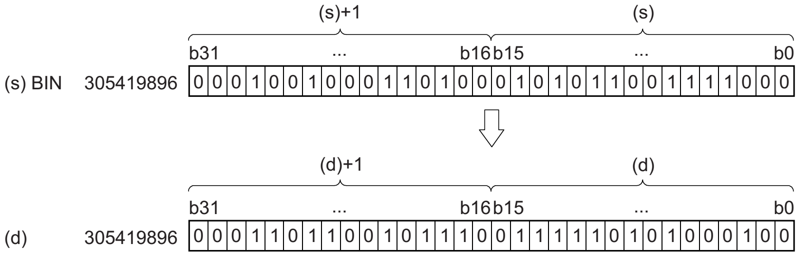
Operand		Description	Range	Data type	Data type (label)
(s)	DGRY(P)	Binary data or the start device where the binary data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGRY(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGRY(P)	Start device for storing the converted Gray code data	—	32-bit signed binary	ANY32_S
	DGRY(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary data in the device specified by (s) to 32-bit binary Gray code data, and store the converted data in the device specified by (d).



(s)+1: Upper 16 bits

(s): Lower 16 bits

Operation error

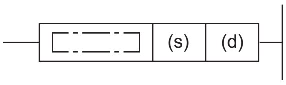
Error code (SD0)	Description
3401H	When the DGRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

Converting 16-bit binary Gray code data to 16-bit binary data

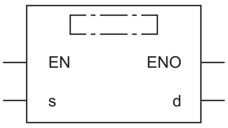
GBIN(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✕

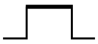
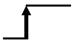
These instructions convert the specified 16-bit binary Gray code data to 16-bit binary data.

Ladder	ST	
	ENO:=GBIN(EN,s,d); ENO:=GBINP(EN,s,d);	ENO:=GBIN_U(EN,s,d); ENO:=GBINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GBIN GBIN_U	
GBINP GBINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	GBIN(P)	Gray code data or the device where the Gray code data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GBIN(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GBIN(P)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
	GBIN(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary Gray code data in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).

		b15							···								b0
(s)	1234	0	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1

⇩

		b15							···								b0
(d) BIN	1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

Operation error

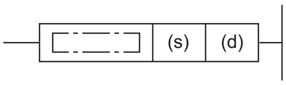
Error code (SD0)	Description
3401H	When the GBIN(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 32767.

Converting 32-bit binary Gray code data to 32-bit binary data

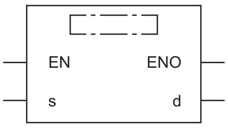
DGBIN(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

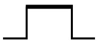
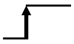
These instructions convert the specified 32-bit binary Gray code data to 32-bit binary data.

Ladder	ST	
	ENO:=DGBIN(EN,s,d); ENO:=DGBINP(EN,s,d);	ENO:=DGBIN_U(EN,s,d); ENO:=DGBINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DGBIN DGBIN_U	
DGBINP DGBINP_U	

Setting data

■Description, range, data type

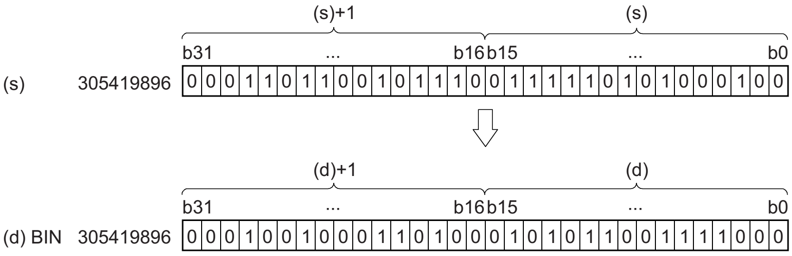
Operand		Description	Range	Data type	Data type (label)
(s)	DGBIN(P)	Gray code data or the start device where the Gray code data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGBIN(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGBIN(P)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
	DGBIN(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary Gray code data in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).



(s)+1: Upper 16 bits

(s): Lower 16 bits

Operation error

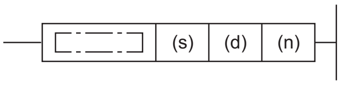
Error code (SD0)	Description
3401H	When the DGBIN(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

Converting 16-bit binary data block to BCD 4-digit data block

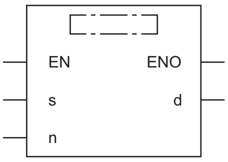
BKBCD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

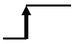
These instructions convert the n points of binary data (0 to 9999) starting from the specified device to BCD data.

Ladder	ST
	ENO:=BKBCD(EN,s,n,d); ENO:=BKBCDP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKBCD	
BKBCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the binary data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the converted BCD data	—	BCD 4-digit	ANY16[*1]
(n)	Number of variables	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

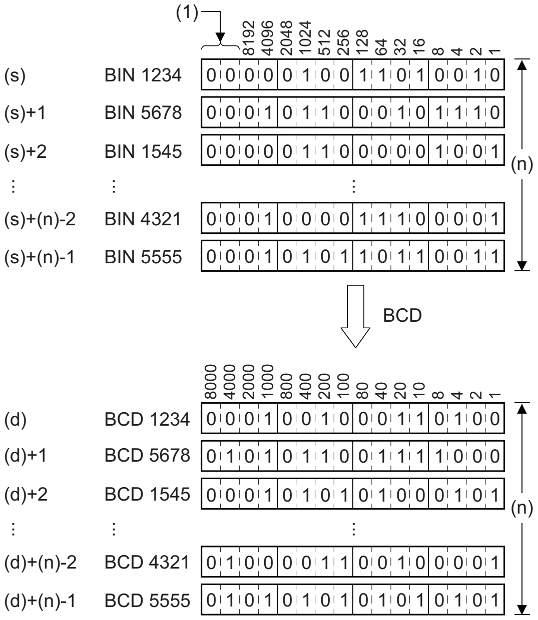
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the (n) points of 16-bit binary data (0 to 9999) starting from the device specified by (s) to BCD data, and store the converted data in the device specified by (d) and later.



(1) Set 0s.

Operation error

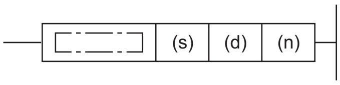
Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3401H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

Converting BCD 4-digit block data to 16-bit binary block data

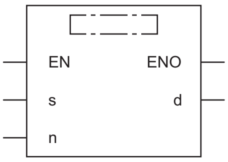
BKBIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

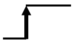
These instructions convert the n points of BCD data (0 to 9999) starting from the specified device to binary data.

Ladder	ST
	ENO:=BKBIN(EN,s,n,d); ENO:=BKBINP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKBIN	
BKBINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the BCD data is stored	—	BCD 4-digit	ANY16[*1]
(d)	Start device for storing the converted binary data	—	16-bit signed binary	ANY16[*1]
(n)	Number of variables	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

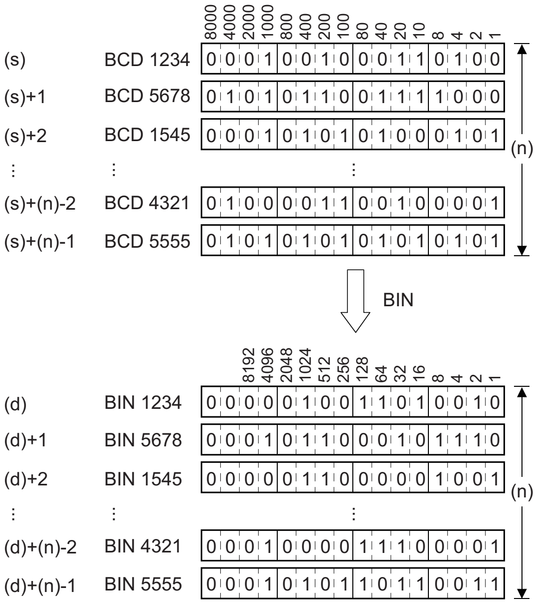
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the (n) points of BCD data (0 to 9999) starting from the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d) and later.



Operation error

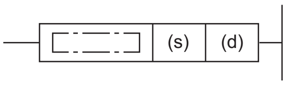
Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3401H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

Converting decimal ASCII data to 16-bit binary data

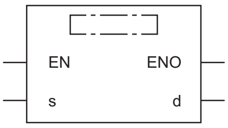
DABIN(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

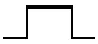
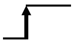
These instructions convert decimal ASCII data to 16-bit binary data.

Ladder	ST	
	ENO:=DABIN(EN,s,d); ENO:=DABINP(EN,s,d);	ENO:=DABIN_U(EN,s,d); ENO:=DABINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DABIN DABIN_U	
DABINP DABINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)		ASCII data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	DABIN(P)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
	DABIN(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

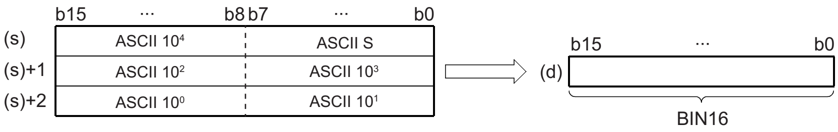
- These instructions convert the decimal ASCII data in the device specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (a sign + 5 digits in the numeric part).	Page 439 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: a sign + 5 digits in the numeric part).	Page 440 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

Set decimal ASCII data with the fixed number of digits in (s) to (s)+2.



ASCII S: Sign data of ASCII code

ASCII 10^4: Ten-thousands place of ASCII code

ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

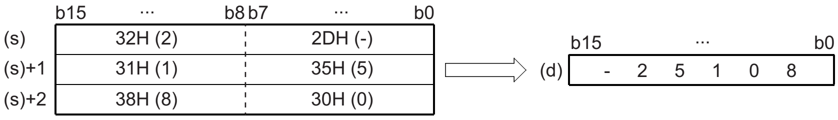
ASCII 10^1: Tens place of ASCII code

ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+2 is within the range from -32768 to 32767 for the DABIN(P) instruction, and it is within the range from 0 to 65535 for the DABIN(P)_U instruction.
- The data of (s)+3 or later is ignored.
- As sign data, set 20H (space) when the ASCII data is positive, and set 2DH (-) when the data is negative. (If a value other than 20H and 2DH is set, the data will be processed as positive data.)
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

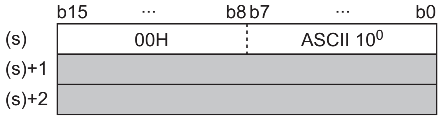
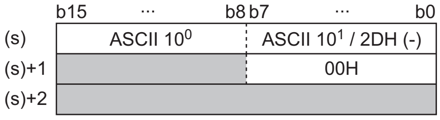
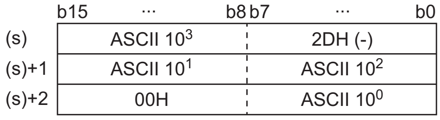
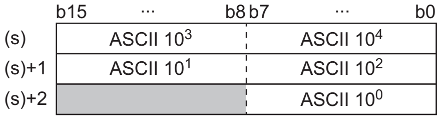
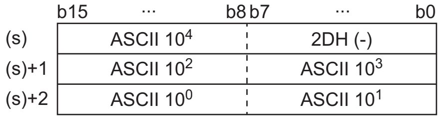
"-25108" is set in (s) when the DABIN(P) instruction is used



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set if the integral part has the maximum number of digits (5 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+2	Value to be set in (s)	Data of (s) to (s)+2
• 0 • Positive value (1 digit in numeric part)	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. 	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
⋮			
• Negative value (4 digits in numeric part)	• Set 00H in the upper byte of (s)+2. 	• Positive value (5 digits in numeric part)	• The data of the upper byte of (s)+2 or later is ignored. Since the number of digits is the maximum, 00H is not required to be set. 
• Negative value (5 digits in numeric part)	• The data of (s)+3 or later is ignored. Since the number of digits is the maximum, 00H is not required to be set. 	ASCII 10^0: Ones place of ASCII code ASCII 10^1: Tens place of ASCII code ⋮ ASCII 10^4: Ten-thousands place of ASCII code	

- The ASCII data in the device specified by (s) to (s)+2 is within the range from -32768 to 32767 for the DABIN(P) instruction, and it is within the range from 0 to 65535 for the DABIN(P)_U instruction.
- Set 2DH (-) to lower byte of (s)+0 as sign data when the ASCII data is negative. Set an ASCII code of the uppermost digit instead of setting sign data when the ASCII data is 0 or positive.
- A value from 30H to 39H can be set in each place of ASCII code.
- If the value is positive and the numeric part has 5 digits, the data of the upper byte of (s)+2 or later is ignored. If the value is negative and the numeric part has 5 digits, the data of (s)+3 or later is ignored.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.
- In the following cases, 0 is stored.
- The first character is 00H (NULL).
- The first character is 2DH (-) and the second character is 00H (NULL).

Operation error

Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+2. • The ASCII code of the first character is other than 2DH, 30H to 39H, 20H, and 00H.[*1] • The ASCII code of the second character or later is other than 30H to 39H, 20H, and 00H. • When the DABIN(P) instruction is used, ASCII data is out of the range from -32768 to 32767. • When the DABIN(P)_U instruction is used, ASCII data is out of the range from 0 to 65535.

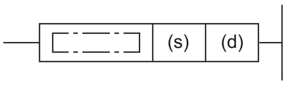
[*1] When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

Converting decimal ASCII data to 32-bit binary data

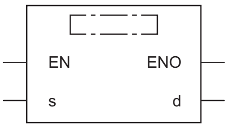
DDABIN(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

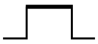
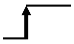
These instructions convert decimal ASCII data to 32-bit binary data.

Ladder	ST	
	ENO:=DDABIN(EN,s,d); ENO:=DDABINP(EN,s,d);	ENO:=DDABIN_U(EN,s,d); ENO:=DDABINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDABIN DDABIN_U	
DDABINP DDABINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)		ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	DDABIN(P)	Start device for storing the conversion result	—	32-bit signed binary	ANY32_S
	DDABIN(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

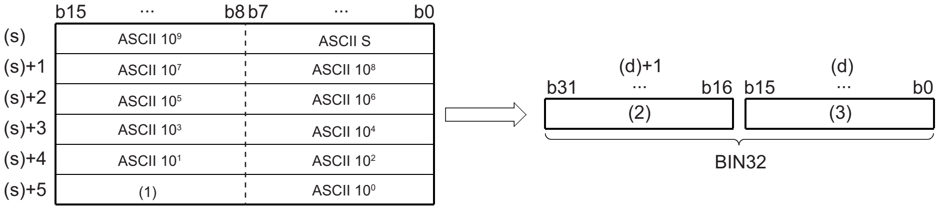
- These instructions convert the decimal ASCII data in the device areas specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (a sign + 10 digits in the numeric part).	Page 442 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: a sign + 10 digits in the numeric part).	Page 443 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

[*1] For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

Set decimal ASCII data with the fixed number of digits in (s) to (s)+5.

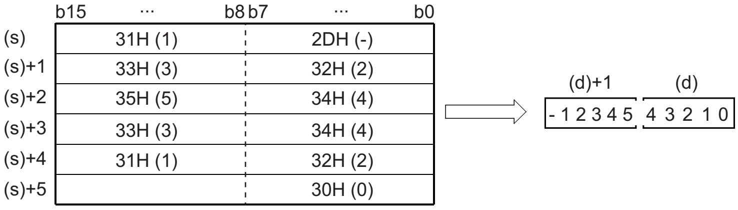


ASCII S: Sign data of ASCII code
ASCII 10^0: Ones place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^8: Hundred-millions place of ASCII code
ASCII 10^9: Billions place of ASCII code
(1) Ignored.
(2) Upper 16 bits
(3) Lower 16 bits

- The ASCII data in the device specified by (s) to (s)+5 is within the range from -2147483648 to 2147483647 for the DDABIN(P) instruction, and it is within the range from 0 to 4294967295 for the DDABIN(P)_U instruction. Any data stored in the upper bytes in the device specified by (s)+5 and data in the device specified by (s)+6 and later are ignored.
- As sign data, set 20H if the ASCII data is positive, and set 2DH if the data is negative. (If a value other than 20H and 2DH is set, the data will be processed as positive data.)
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

"-1234543210" is set in (s) when the DDABIN(P) instruction is used



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set if the integral part has the maximum number of digits (10 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+5</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+5</th></tr>
<tr>
<td>• 0
• Positive value (1 digit in numeric part)</td>
<td>• Set 00H in the upper byte of (s).
• The data of (s)+1 or later is ignored.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>00H</td><td>ASCII 10^{0}</td></tr>
<tr><td>(s)+1</td><td colspan="2"></td></tr>
<tr><td>(s)+2</td><td colspan="2"></td></tr>
<tr><td>(s)+3</td><td colspan="2"></td></tr>
<tr><td>(s)+4</td><td colspan="2"></td></tr>
<tr><td>(s)+5</td><td colspan="2"></td></tr>
</table></td>
<td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td>
<td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>ASCII 10^{0}</td><td>ASCII 10^{1} / 2DH (-)</td></tr>
<tr><td>(s)+1</td><td></td><td>00H</td></tr>
<tr><td>(s)+2</td><td colspan="2"></td></tr>
<tr><td>(s)+3</td><td colspan="2"></td></tr>
<tr><td>(s)+4</td><td colspan="2"></td></tr>
<tr><td>(s)+5</td><td colspan="2"></td></tr>
</table></td>
</tr>
<tr><td colspan="4">⋮</td></tr>
<tr>
<td>• Positive value (9 digits in numeric part)
• Negative value (8 digits in numeric part)</td>
<td>• Set 00H in the upper byte of (s)+4.
• (s)+5 is ignored.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>ASCII 10^{7}</td><td>ASCII 10^{8} / 2DH (-)</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^{5}</td><td>ASCII 10^{6}</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^{3}</td><td>ASCII 10^{4}</td></tr>
<tr><td>(s)+3</td><td>ASCII 10^{1}</td><td>ASCII 10^{2}</td></tr>
<tr><td>(s)+4</td><td>00H</td><td>ASCII 10^{0}</td></tr>
<tr><td>(s)+5</td><td colspan="2"></td></tr>
</table></td>
<td>• Negative value (9 digits in numeric part)</td>
<td>• Set 00H in the lower byte of (s)+5.
• The data of the upper byte of (s)+5 is ignored.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>ASCII 10^{8}</td><td>2DH (-)</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^{6}</td><td>ASCII 10^{7}</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^{4}</td><td>ASCII 10^{5}</td></tr>
<tr><td>(s)+3</td><td>ASCII 10^{2}</td><td>ASCII 10^{3}</td></tr>
<tr><td>(s)+4</td><td>ASCII 10^{0}</td><td>ASCII 10^{1}</td></tr>
<tr><td>(s)+5</td><td></td><td>00H</td></tr>
</table></td>
</tr>
<tr>
<td>• Positive value (10 digits in numeric part)</td>
<td>• (s)+5 is ignored. Since the number of digits is the maximum, 00H is not required to be set.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>ASCII 10^{8}</td><td>ASCII 10^{9}</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^{6}</td><td>ASCII 10^{7}</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^{4}</td><td>ASCII 10^{5}</td></tr>
<tr><td>(s)+3</td><td>ASCII 10^{2}</td><td>ASCII 10^{3}</td></tr>
<tr><td>(s)+4</td><td>ASCII 10^{0}</td><td>ASCII 10^{1}</td></tr>
<tr><td>(s)+5</td><td></td><td></td></tr>
</table></td>
<td>• Negative value (10 digits in numeric part)</td>
<td>• The data of the upper byte of (s)+5 is ignored. Since the number of digits is the maximum, 00H is not required to be set.
<table>
<tr><td></td><td>b15 ··· b8</td><td>b7 ··· b0</td></tr>
<tr><td>(s)</td><td>ASCII 10^{9}</td><td>2DH (-)</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^{7}</td><td>ASCII 10^{8}</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^{5}</td><td>ASCII 10^{6}</td></tr>
<tr><td>(s)+3</td><td>ASCII 10^{3}</td><td>ASCII 10^{4}</td></tr>
<tr><td>(s)+4</td><td>ASCII 10^{1}</td><td>ASCII 10^{2}</td></tr>
<tr><td>(s)+5</td><td></td><td>ASCII 10^{0}</td></tr>
</table></td>
</tr>
<tr><td colspan="4">ASCII 10^{0}: Ones place of ASCII code
ASCII 10^{1}: Tens place of ASCII code
⋮
ASCII 10^{9}: Billions place of ASCII code</td></tr>
</table>

- The ASCII data in the device specified by (s) to (s)+5 is within the range from -2147483648 to 2147483647 for the DDABIN(P) instruction, and it is within the range from 0 to 4294967295 for the DDABIN(P)_U instruction.
- Set 2DH (-) to lower byte of (s)+0 as sign data when the ASCII data is negative. Set an ASCII code of the uppermost digit instead of setting sign data when the ASCII data is 0 or positive.
- A value from 30H to 39H can be set in each place of ASCII code.
- If the value is positive and the numeric part has 10 digits, the data stored in (s)+5 or later is ignored. If the value is negative and the numeric part has 10 digits, the data stored in the upper byte of (s)+5 or later is ignored.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.
- In the following cases, "0" is stored in (d).
  - The first character is 00H (NULL).
  - The first character is 2DH (-) and the second character is 00H (NULL).

Operation error

Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+5. • The ASCII code of the first character is other than 2DH, 30H to 39H, 20H, and 00H.[1] • The ASCII code of the second character or later is other than 30H to 39H, 20H, and 00H. • When the DDABIN(P) instruction is used, ASCII data is out of the range from -2147483648 to 2147483647. • When the DDABIN(P)_U instruction is used, ASCII data is out of the range from 0 to 4294967295.

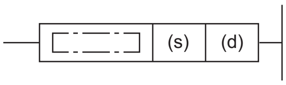
[1] When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

Converting hexadecimal ASCII data to 16-bit binary data

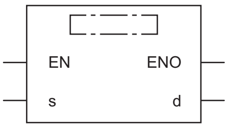
HABIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

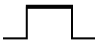
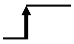
These instructions convert hexadecimal ASCII data to 16-bit binary data.

Ladder	ST
	ENO:=HABIN(EN,s,d); ENO:=HABINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
HABIN	
HABINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Device for storing the conversion result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

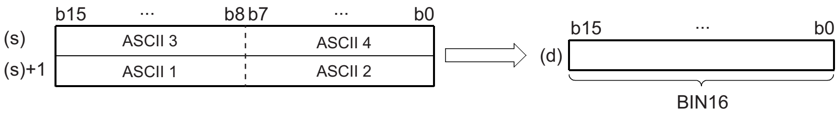
- These instructions convert the hexadecimal ASCII data stored in the device areas specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (4 digits).	Page 446 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: 4 digits).	Page 446 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

[*1] For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set hexadecimal ASCII data with 4 digits (fixed) in (s) to (s)+1.

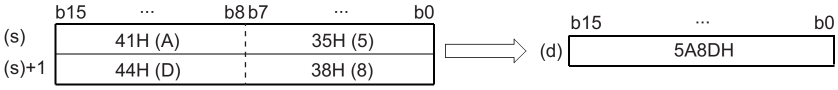


ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0000H to FFFFH.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.

Ex.

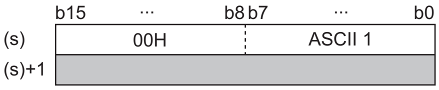
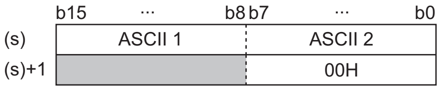
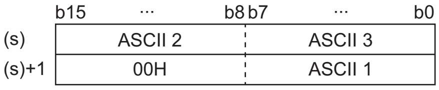
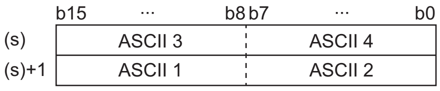
When 5A8DH is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set hexadecimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (4 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+1	Value to be set in (s)	Data of (s) to (s)+1
• 0H to FH	• Set 00H in the upper byte of (s)+0. • The data of (s)+1 or later is ignored.  	• 10H to FFH	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
• 100H to FFFH	• Set 00H in the upper byte of (s)+1. • The data of (s)+2 or later is ignored.  	• 1000H to FFFFH	• The data of (s)+2 or later is ignored. 

ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0000H to FFFFH.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.
- If a value 00H is set in each place of ASCII code, the value will be processed as the end of the hexadecimal ASCII data.

Operation error

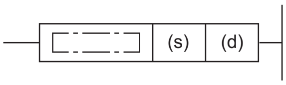
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+1. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

Converting hexadecimal ASCII data to 32-bit binary data

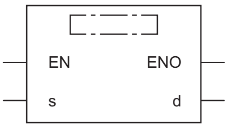
DHABIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

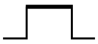
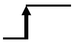
These instructions convert hexadecimal ASCII data to 32-bit binary data.

Ladder	ST
	ENO:=DHABIN(EN,s,d); ENO:=DHABINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DHABIN	
DHABINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the conversion result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

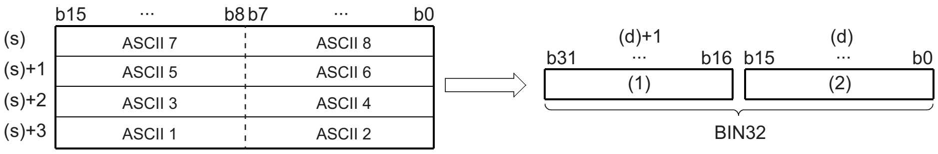
- These instructions convert the hexadecimal ASCII data stored in the device specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (8 digits).	Page 449 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: 8 digits).	Page 450 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set hexadecimal ASCII data with 8 digits (fixed) in (s) to (s)+3.



ASCII □: ASCII code (□th digit)

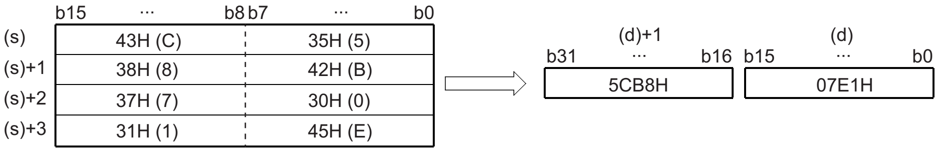
(1) Upper 16 bits

(2) Lower 16 bits

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 00000000H to FFFFFFFFH.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.

Ex.

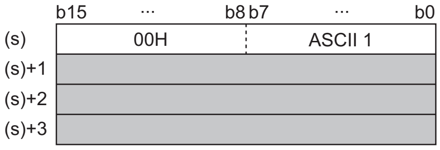
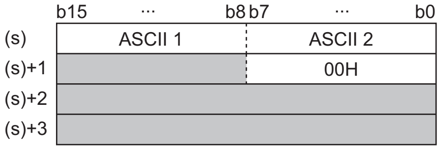
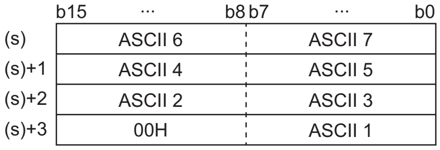
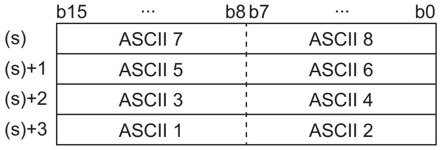
When 5CB807E1H is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set hexadecimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (8 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+3	Value to be set in (s)	Data of (s) to (s)+3
• 0H to FH	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. 	• 10H to FFH	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
⋮			
• 1000000H to FFFFFFFH	• Set 00H in the upper byte of (s)+3. • The data of (s)+4 or later is ignored. 	• 10000000H to FFFFFFFFH	• The data of (s)+4 or later is ignored. 

ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 00000000H to FFFFFFFFH.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.
- If a value 00H is set in each place of ASCII code, the value will be processed as the end of the hexadecimal ASCII data.

Operation error

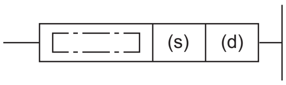
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+3. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

Converting decimal ASCII data to BCD 4-digit data

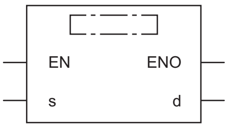
DABCD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

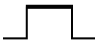
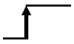
These instructions convert decimal ASCII data to BCD 4-digit data.

Ladder	ST
	ENO:=DABCD(EN,s,d); ENO:=DABCDP(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
DABCD	
DABCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to BCD data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Device for storing the conversion result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

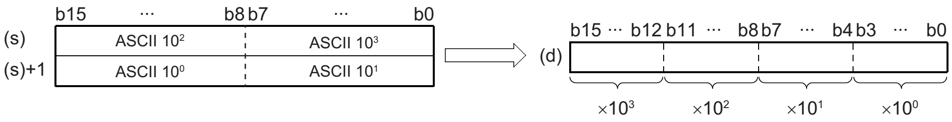
- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 4-digit data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (4 digits).	Page 452 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: 4 digits).	Page 453 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

[1] For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set decimal ASCII data the 4 digits (fixed) in (s) to (s)+1.

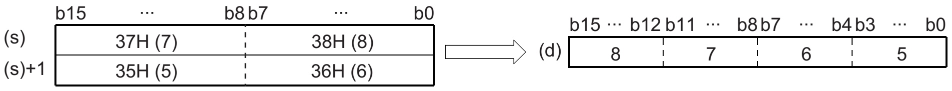


ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0 to 9999.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

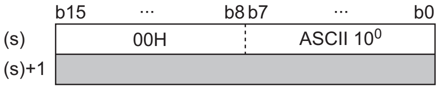
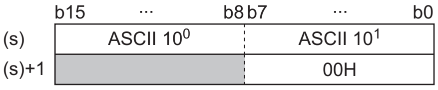
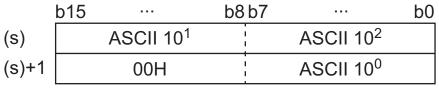
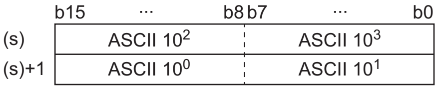
When 8765 is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (4 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+1	Value to be set in (s)	Data of (s) to (s)+1
• 0 to 9	• Set 00H in the upper byte of (s)+0. • The data of (s)+1 or later is ignored. 	• 10 to 99	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
• 100 to 999	• Set 00H in the upper byte of (s)+1. • The data of (s)+2 or later is ignored. 	• 1000 to 9999	• The data of (s)+2 or later is ignored. 

ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0 to 9999.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.

Operation error

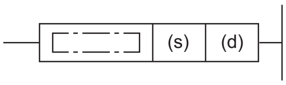
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

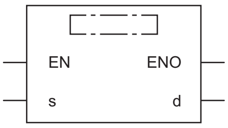
Converting decimal ASCII data to BCD 8-digit data

DDABCD(P)

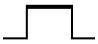
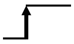
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions convert decimal ASCII data to BCD 8-digit data.

Ladder	ST
	ENO:=DDABCD(EN,s,d); ENO:=DDABCDP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DDABCD	
DDABCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to BCD data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the conversion result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

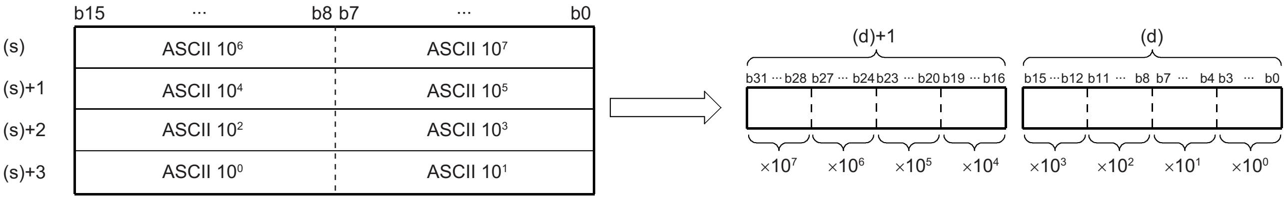
- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 8-digit data, and store the converted data in the device number specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (8 digits).	Page 455 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
ON	Set (s) with a desired number of digits (maximum: 8 digits).	Page 456 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set decimal ASCII data with 8 digits (fixed) in (s) to (s)+3.

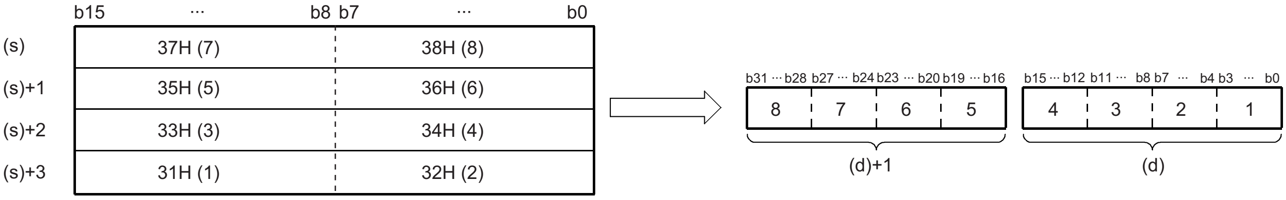


ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 0 to 99999999.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

When 87654321 is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (8 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+3</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+3</th></tr>
<tr>
<td>• 0 to 9</td>
<td>• Set 00H in the upper byte of (s).
• The data of (s)+1 or later is ignored.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(s)</td><td>00H</td><td>ASCII 10^0</td></tr>
<tr><td>(s)+1</td><td colspan="2">▨</td></tr>
<tr><td>(s)+2</td><td colspan="2">▨</td></tr>
<tr><td>(s)+3</td><td colspan="2">▨</td></tr>
</table></td>
<td>• 10 to 99</td>
<td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(s)</td><td>ASCII 10^0</td><td>ASCII 10^1</td></tr>
<tr><td>(s)+1</td><td>▨</td><td>00H</td></tr>
<tr><td>(s)+2</td><td colspan="2">▨</td></tr>
<tr><td>(s)+3</td><td colspan="2">▨</td></tr>
</table></td>
</tr>
<tr><td colspan="4">⋮</td></tr>
<tr>
<td>• 1000000 to 9999999</td>
<td>• Set 00H in the upper byte of (s)+3.
• The data of (s)+4 or later is ignored.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(s)</td><td>ASCII 10^5</td><td>ASCII 10^6</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^3</td><td>ASCII 10^4</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^1</td><td>ASCII 10^2</td></tr>
<tr><td>(s)+3</td><td>00H</td><td>ASCII 10^0</td></tr>
</table></td>
<td>• 10000000 to 99999999</td>
<td>• The data of (s)+4 or later is ignored.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(s)</td><td>ASCII 10^6</td><td>ASCII 10^7</td></tr>
<tr><td>(s)+1</td><td>ASCII 10^4</td><td>ASCII 10^5</td></tr>
<tr><td>(s)+2</td><td>ASCII 10^2</td><td>ASCII 10^3</td></tr>
<tr><td>(s)+3</td><td>ASCII 10^0</td><td>ASCII 10^1</td></tr>
</table></td>
</tr>
</table>

ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 0 to 99999999.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.

Operation error

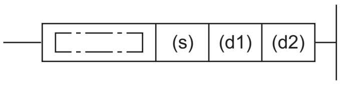
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

Converting decimal string data to 16-bit binary data

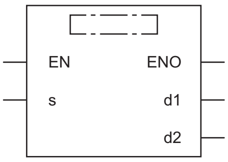
VAL(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

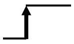
These instructions convert character strings to 16-bit binary data.

Ladder	ST	
	ENO:=VAL(EN,s,d1,d2); ENO:=VALP(EN,s,d1,d2);	ENO:=VAL_U(EN,s,d1,d2); ENO:=VALP_U(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
VAL VAL_U	
VALP VALP_U	

Setting data

■Description, range, data type

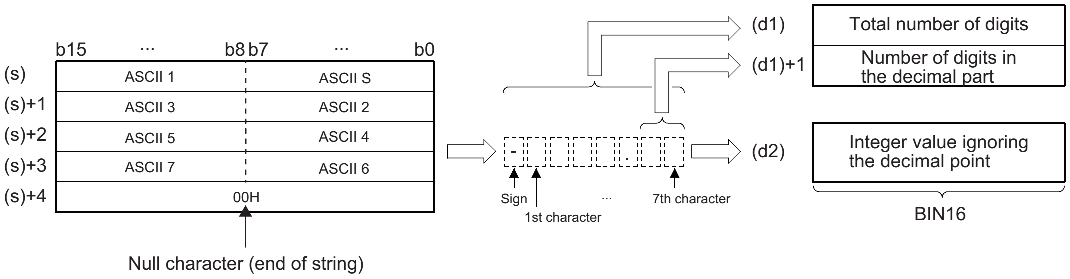
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	VAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	VAL(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(d2)	VAL(P)	Device for storing the binary data after conversion	—	16-bit signed binary	ANY16_S
	VAL(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

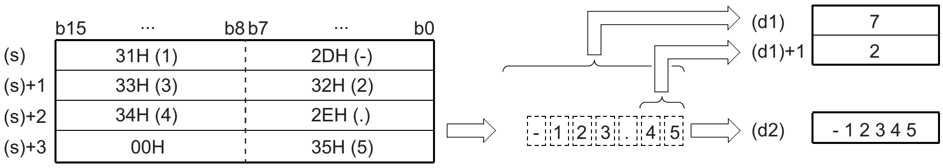
- These instructions convert the character strings stored in the device numbers specified by (s) and later to 16-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of character strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as character strings.



ASCII S: ASCII code (sign data)
ASCII □: ASCII code (□th character)

Ex.

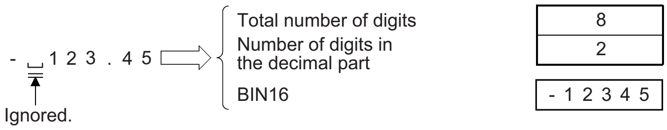
When a string "-123.45" (signed) is stored in the device specified by (s) or later



- The total number of characters of the character string stored in the device specified by (s) is 2 to 8.
- Of the character string stored in the device specified by (s), the number of characters in the decimal part is 0 to 5. Note, however, that the number must not exceed the total number of digits minus 3.
- A numerical character string that can be converted to binary data ranges from -32768 to 32767 when a signed value is specified ignoring the decimal point or from 0 to 65535 when an unsigned value is specified. Numerical character strings excluding signs and decimal points can be specified only within the range from 30H to 39H. (A value ignoring the decimal point···"-12345.6" for example becomes "-123456".)
- For the sign, "20H" can be set to indicate a positive numerical value, or "2DH" can be set to indicate a negative numerical value.
- "2EH" is set for the decimal point.
- The total number of digits stored in the device specified by (d1) includes all characters (including signs and decimal points) that represent a numerical value. The number of digits in the decimal part to be stored in the device specified by (d1)+1 represents the decimal part after 2EH(.) For the 16-bit binary data to be stored in the device specified by (d2), the character string is converted to binary data by ignoring the decimal point and stored.
- If "20H" (space) or "30H" (0) exists between the sign and the first numerical value other than 0 in the character string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

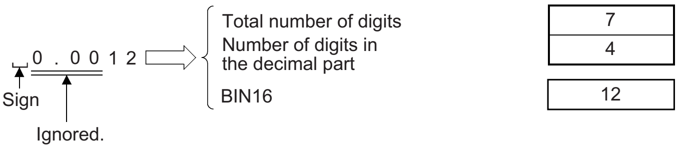
Ex.

When "20H" exists between the sign and the first numerical value other than 0 (when a signed value is specified)



Ex.

When "30H" exists between the sign and the first numerical value other than 0



Operation error

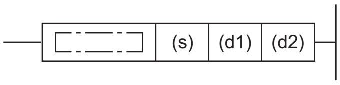
Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3401H	Invalid data that cannot be converted are input in (s). • The number of characters is not between 2 and 8. • The number of characters in the decimal part is not between 0 and 5. • The relationship between the total number of characters and the number of characters in the decimal part is not in the following range. Total number of characters - 3≥number of characters in the decimal part • When the VAL(P) instruction is used, an ASCII code other than "20H" and "2DH" is set as a sign. • When the VAL(P)_U instruction is used, an ASCII code other than "20H" is set as a sign. • An ASCII code other than "30H" to "39H" and "2EH" (decimal point) is set as a digit of individual numbers. • More than one decimal point are set. The converted binary value exceeds the range in which each instruction can implement conversion.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

Converting decimal string data to 32-bit binary data

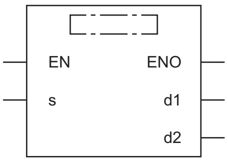
DVAL(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

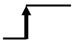
These instructions convert character strings to 32-bit binary data.

Ladder	ST	
	ENO:=DVAL(EN,s,d1,d2); ENO:=DVALP(EN,s,d1,d2);	ENO:=DVAL_U(EN,s,d1,d2); ENO:=DVALP_U(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DVAL DVAL_U	
DVALP DVALP_U	

Setting data

■Description, range, data type

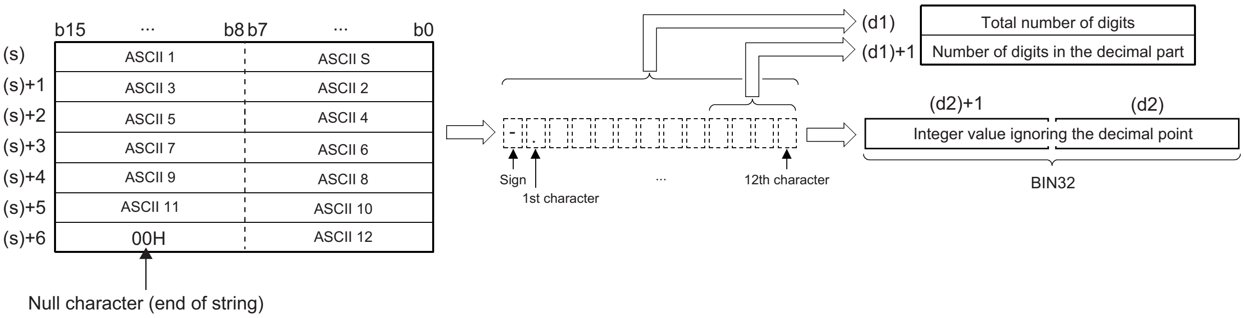
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	DVAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	DVAL(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(d2)	DVAL(P)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
	DVAL(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

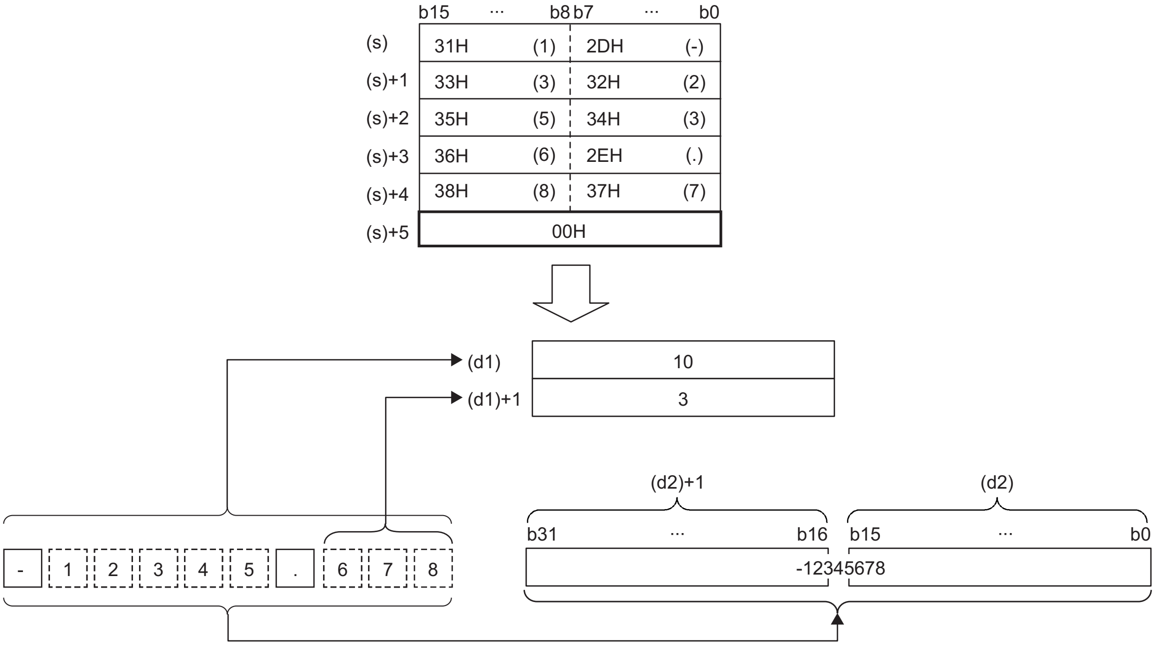
- These instructions convert the character strings stored in the device numbers specified by (s) and later to 32-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of character strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as character strings.



ASCII S: ASCII code (sign data)
ASCII □: ASCII code (□th character)

Ex.

When a string "-12345.678" (signed) is stored in the device specified by (s) or later

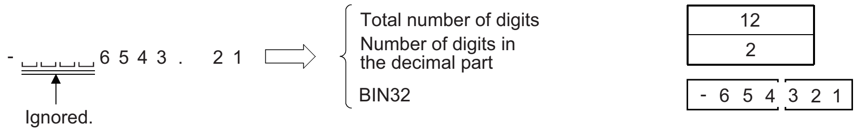


- The total number of characters of the character string stored in the device specified by (s) is 2 to 13.
- Of the character string stored in the device specified by (s), the number of characters in the decimal part is 0 to 10. Note, however, that the number must not exceed the total number of digits minus 3.
- The range of numerical character strings that can be converted to binary is as follows. Numerical character strings excluding signs and decimal points can be specified only within the range from 30H to 39H. (A value ignoring the decimal point···"-12345.6" for example becomes "-123456".)
  - When a signed value ignoring the decimal point is specified: -2147483648 to 2147483647
  - When an unsigned value ignoring the decimal point is specified: 0 to 4294967295
- For the sign, "20H" can be set to indicate a positive numerical value, or "2DH" can be set to indicate a negative numerical value.
- "2EH" is set for the decimal point.
- The total number of digits stored in the device specified by (d1) includes all characters (including signs and decimal points) that represent a numerical value. The number of digits in the decimal part to be stored in the device specified by (d1)+1 represents the decimal part after 2EH(.) For the 32-bit binary data to be stored in the device specified by (d2), the character string is converted to binary data by ignoring the decimal point and stored.

- If "20H" (space) or "30H" (0) exists between the sign and the first numerical value other than 0 in the character string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

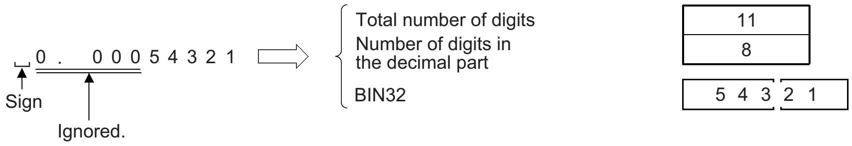
Ex.

When "20H" exists between the sign and the first numerical value other than 0 (when a signed value is specified)



Ex.

When "30H" exists between the sign and the first numerical value other than 0



Operation error

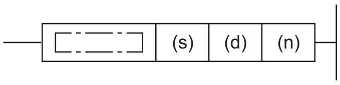
Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3401H	Invalid data that cannot be converted are input in (s). • The number of characters of the character string is not between 2 and 13. • The number of characters in the decimal part of the character string is not between 0 and 10. • The relationship between the total number of characters and the number of characters in the decimal part is not in the following range. Total number of characters - 3≥number of characters in the decimal part • When the DVAL(P) instruction is used, an ASCII code other than "20H" and "2DH" is set as a sign. • When the DVAL(P)_U instruction is used, an ASCII code other than "20H" is set as a sign. • An ASCII code other than "30H" to "39H" and "2EH" (decimal point) is set as a digit of individual numbers. • More than one decimal point are set.
	The converted binary value exceeds the range in which each instruction can implement conversion.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

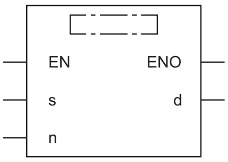
Converting hexadecimal ASCII to hexadecimal binary data

ASC2INT(P)

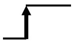
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions convert hexadecimal ASCII data to binary data.

Ladder	ST
	ENO:=ASC2INT(EN,s,n,d); ENO:=ASC2INTP(EN,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
ASC2INT	
ASC2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the string data to be converted to binary data	—	String	ANYSTRING_SINGLE*1
(d)	Start device for storing the converted binary data	—	16-bit signed binary	ANY16*1
(n)	Number of characters to be stored	0 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

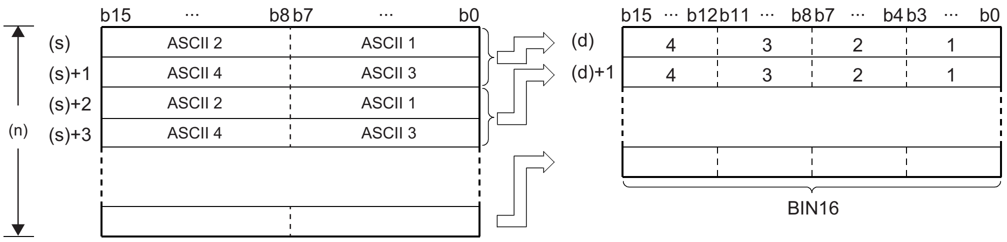
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the hexadecimal ASCII data stored in the device by the number of characters specified by (n) after the device number specified by (s) and later to binary data, and store the converted data in the device number specified by (d) and later.

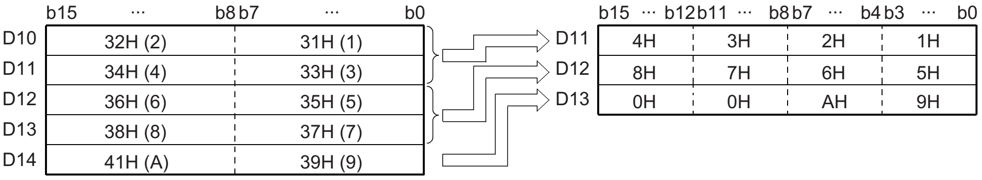


ASCII □: ASCII code (□th digit)

1 to 4: 1st to 4th digit

(n): Number of characters specified by (n)

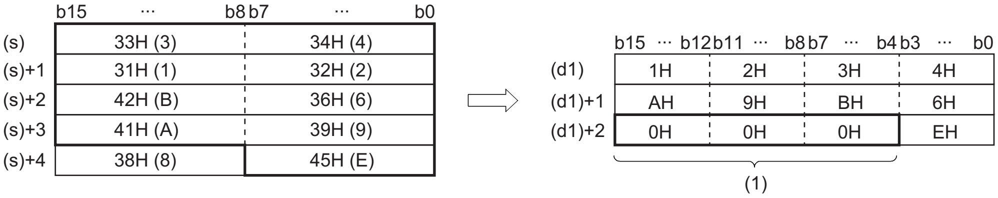
- Setting the number of characters for (n) automatically determines the range of the character string in the device specified by (s) and the device range in which the binary data in the device specified by (d) is stored.
- Processing is performed normally even if the device range in which the ASCII data to be converted and the device range for storing the converted binary data overlap.



- If the number of characters in the device specified by (n) is not a multiple of 4, "0H" is automatically stored after the specified number of the last device number among device numbers for storing the converted binary data.

Ex.

When the number of characters in (n) is 9



(1) 0H is automatically stored.

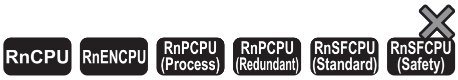
- If the number of characters in the device specified by (n) is 0, no processing is performed.
- The ASCII code that can be specified by (s) must be in the range from "30H" to "39H" or "41H to 46H".

Operation error

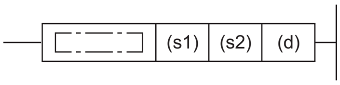
Error code (SD0)	Description
3401H	A character other than hexadecimal numerical character string (an ASCII code other than "30H" to "39H" and "41H" to "46H") is set in the device specified by (s).
3405H	Out-of-range data is set in the device specified by (n). • The specified number of characters is not between 0 and 16383.

Converting single-precision real number to BCD format data

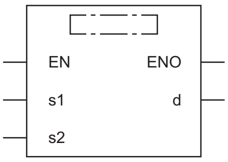
EMOD(P)



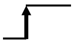
These instructions convert single-precision real number data to the BCD floating point format data.

Ladder	ST
	ENO:=EMOD(EN,s1,s2,d); ENO:=EMODP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EMOD	
EMODP	

Setting data

■Description, range, data type

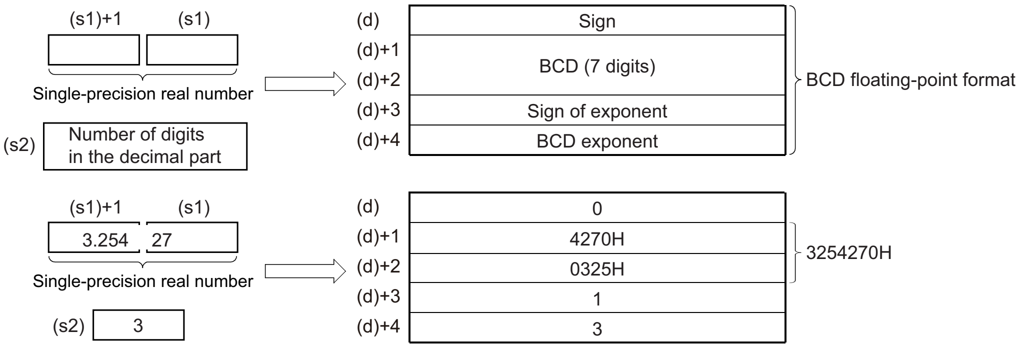
Operand	Description	Range	Data type	Data type (label)
(s1)	Single-precision real number data or the start device where the single-precision real number data is stored	$0, 2^{-126} \leq \|(s1)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Decimal part digit data	0 to 7	16-bit signed binary	ANY16
(d)	Start device for storing the BCD format data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

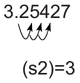
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

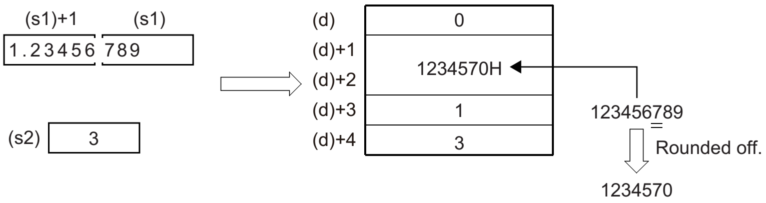
- These instructions convert the single-precision real number data stored in the device specified by (s1) to the BCD floating point format based on the number of decimal part digits stored in the device specified by (s2), and store the converted data in the device number specified by (d) and later.



- For the sign in (d) and the exponent sign in (d)+3, 0 is set for positive and 1 is set for negative.
- For the BCD exponent in (d)+4, a value between 0 and 38 is stored.
- The number of decimal part digits of the single-precision real number data in the device specified by (s1) is stored in the device specified by (S2). The example in the above figure shows the following.



- Six-digit BCD data, determined by rounding off the seventh digit, is stored in (d)+1 and (d)+2.



- A value of 0 to 7 can be set for the number of decimal part digits in the device specified by (s2).
- 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 49 Precautions

Operation error

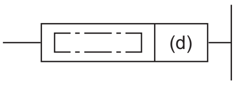
Error code (SD0)	**Description**
3401H	The number of decimal part digits in the device specified by (s2) is out of the range from 0 to 7.
3402H	The value set to a device or label in (s1) is -0, a subnormal number, NaN (not a number), or ±∞.

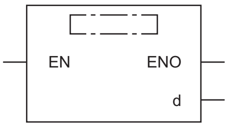
Two's complement of 16-bit binary data (sign inversion)

NEG(P)

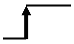
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

Invert the sign of 16-bit binary device.

Ladder	ST
	ENO:=NEG(EN,d); ENO:=NEGP(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
NEG	
NEGP	

Setting data

■Description, range, data type

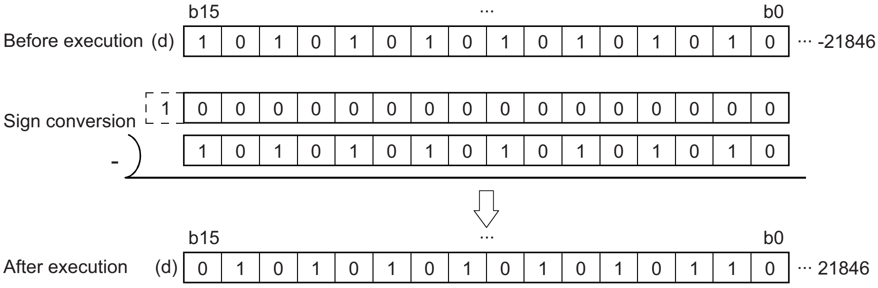
Operand	Description	Range	Data type	Data type (label)
(d)	Device where the data subjected to two's complement is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions invert the sign of the 16-bit binary 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

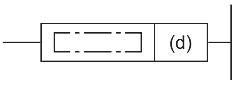
There is no operation error.

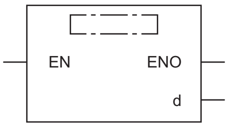
Two's complement of 32-bit binary data (sign inversion)

DNEG(P)

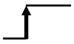
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions invert the sign of 32-bit binary device.

Ladder	ST
	ENO:=DNEG(EN,d); ENO:=DNEGP(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DNEG	
DNEGP	

Setting data

■Description, range, data type

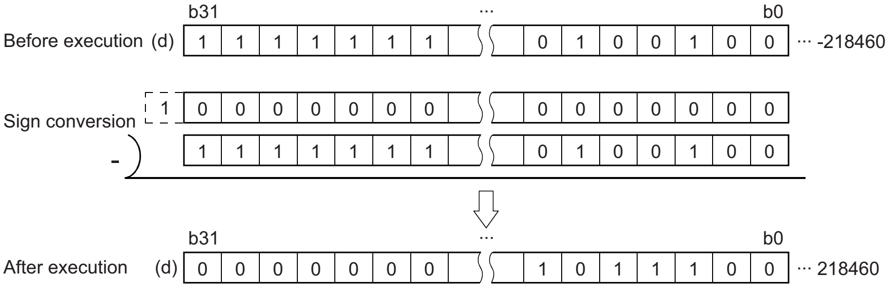
Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the data subjected to two's complement is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions invert the sign of the 32-bit binary 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

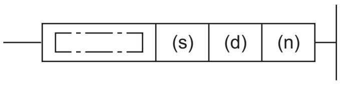
There is no operation error.

Decoding 8-bit data to 256-bit data

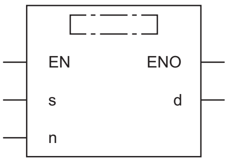
DECO(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

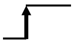
These instructions decode the lower (n) bits of the specified device.

Ladder	ST
	ENO:=DECO(EN,s,n,d); ENO:=DECOP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DECO	
DECOP	

Setting data

■Description, range, data type

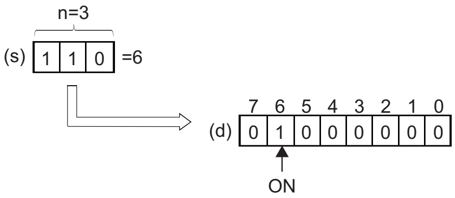
Operand	Description	Range	Data type	Data type (label)
(s)	Decode data or the start number of the device where the decode data is stored	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(d)	Device for storing the decoded data	—	Bit/Word	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions turn on the bit, corresponding to the binary value specified by the lower (n) bits in the device specified by (s), in the device specified by (d)



- Specify a value 1 to 8 for (n).
- When (n)=0, no processing is performed and the values in the device specified by (d) remain unchanged.
- A bit device is treated as 1 bit, and a word device is treated as 16 bits.

Operation error

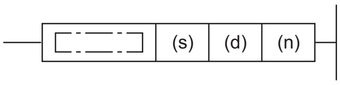
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 8.

Encoding 256-bit data to 8-bit data

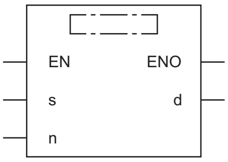
ENCO(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

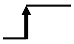
These instructions encode the bit data of 'n'th power of 2.

Ladder	ST
	ENO:=ENCO(EN,s,n,d); ENO:=ENCOP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ENCO	
ENCOP	

Setting data

■Description, range, data type

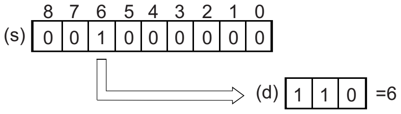
Operand	Description	Range	Data type	Data type (label)
(s)	Device where the encode data is stored	—	Bit/Word	ANY_ELEMENTARY
(d)	Start number of the device for storing the encoded data	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the binary value, corresponding to the bit which is set to 1 in the $2^{(n)}$ bit data in the device specified by (s), in the device specified by (d).



- Specify a value 1 to 8 for (n).
- When (n)=0, no processing is performed and the values in the device specified by (d) remain unchanged.
- A bit device is treated as 1 bit, and a word device is treated as 16 bits.
- When two or more bits are 1, the upper bit position is used for processing.

Operation error

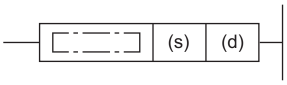
Error code (SD0)	**Description**
3401H	The value specified by (n) is out of the range, 0 to 8.
	The bits in the $2^{(n)}$ bit data in the device specified by (s) are all 0s.

Decoding data to seven-segment display data

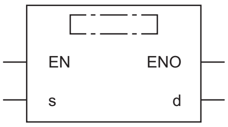
SEG(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

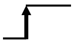
These instructions decode the data consisting of 0 to F specified by the lower 4 bits of the device to seven-segment display data.

Ladder	ST
	ENO:=SEG(EN,s,d); ENO:=SEGP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SEG	
SEGP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Decode data or the device where the decode data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the decoded data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

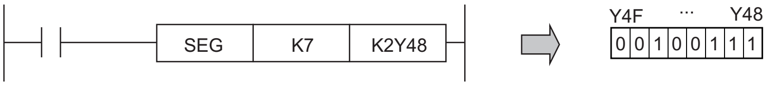
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions decode the data consisting of 0 to F specified by the lower 4 bits in the device specified by (s) to seven-segment display data, and store the decoded data in the device specified by (d).
- In the case of a bit device, (d) indicates the start device for storing 7-segment display data. In the case of a word device, it indicates the device number for storing the data.

Ex.

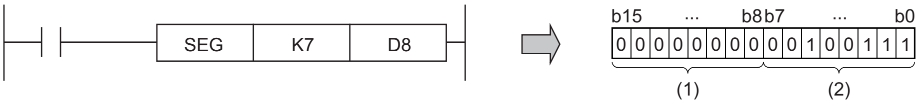
Bit device



The data in Y48 to Y4F does not change until the next data is output.

Ex.

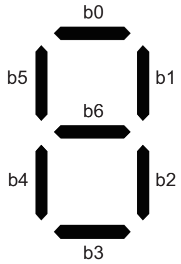
Word device



(1) The upper 8 bits are filled with 0s.

(2) The seven-segment display data are stored in the lower 8 bits.

- The following is the truth table for the seven-segment display.

(s)		Seven-segment display	(d)								Display data
Hexadecimal	**Bit pattern**		**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	
0	0000		0	0	1	1	1	1	1	1	
1	0001		0	0	0	0	0	1	1	0	
2	0010		0	1	0	1	1	0	1	1	
3	0011		0	1	0	0	1	1	1	1	
4	0100		0	1	1	0	0	1	1	0	
5	0101		0	1	1	0	1	1	0	1	
6	0110		0	1	1	1	1	1	0	1	
7	0111		0	0	1	0	0	1	1	1	
8	1000		0	1	1	1	1	1	1	1	
9	1001		0	1	1	0	1	1	1	1	
A	1010		0	1	1	1	0	1	1	1	
B	1011		0	1	1	1	1	1	0	0	
C	1100		0	0	1	1	1	0	0	1	
D	1101		0	1	0	1	1	1	1	0	
E	1110		0	1	1	1	1	0	0	1	
F	1111		0	1	1	1	0	0	0	1	

Operation error

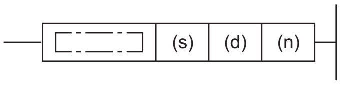
There is no operation error.

Separating data in units of 4 bits

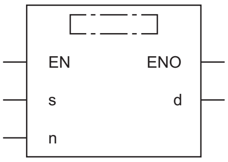
DIS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

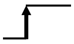
These instructions store the lower (n) nibble(s) of 16-bit binary data in another device range specified.

Ladder	ST
	ENO:=DIS(EN,s,n,d); ENO:=DISP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DIS	
DISP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the separation target data is stored	—	16-bit signed binary	ANY16
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16[*1]
(n)	Number of separation units	1 to 4	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

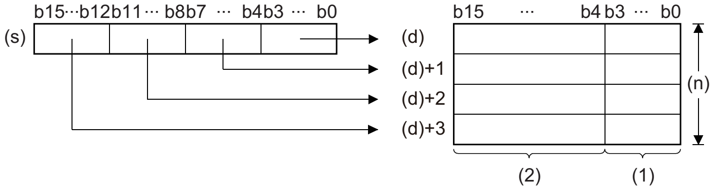
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions separate the lower (n) nibble(s) (4 bits/nibble) from the 16-bit binary data in the device specified by (s), and store each of the separated data in the lower 4 bits of the (n) points of data in the device specified by (d).



(1) Data storage area
(2) Filled with 0s.

- The upper 12 bits of the (n) points of data in the device specified by (d) are filled with 0s.
- Specify a value 1 to 4 for (n).
- When (n)=0, no processing is performed and the (n) points of data starting from the device specified by (d) remain unchanged.

Operation error

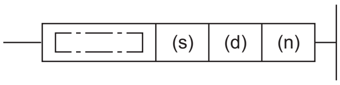
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 4.

Combining data in units of 4 bits

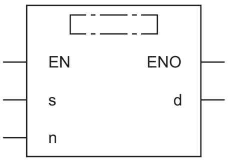
UNI(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

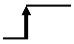
These instructions store the lower 4 bits of the (n) points of 16-bit binary data in another 16-bit device.

Ladder	ST
	ENO:=UNI(EN,s,n,d); ENO:=UNIP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UNI	
UNIP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Device for storing the combined data	—	16-bit signed binary	ANY16
(n)	Number of combination units	1 to 4	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

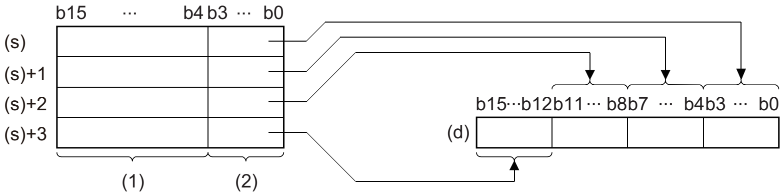
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the lower 4 bits of the (n) points of 16-bit binary data in the device specified by (s) in the 16-bit device specified by (d).



(1) Ignored.

(2) Data to be connected

- The upper bits (bits in the (4-n) nibble(s)) of data in the device specified by (d) are filled with 0s.
- Specify a value 1 to 4 for (n).
- When (n)=0, no processing is performed and the data in the device specified by (d) remain unchanged.

Operation error

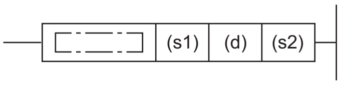
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 4.

Separating data in units of bits

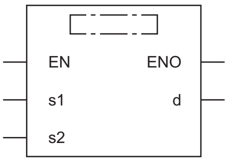
NDIS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

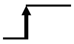
These instructions separate the data in units of bits. (The number of bits can be specified as desired.)

Ladder	ST
	ENO:=NDIS(EN,s1,s2,d); ENO:=NDISP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
NDIS	
NDISP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the separation target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16[*1]
(s2)	Start device for storing the separation unit	—	16-bit signed binary	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

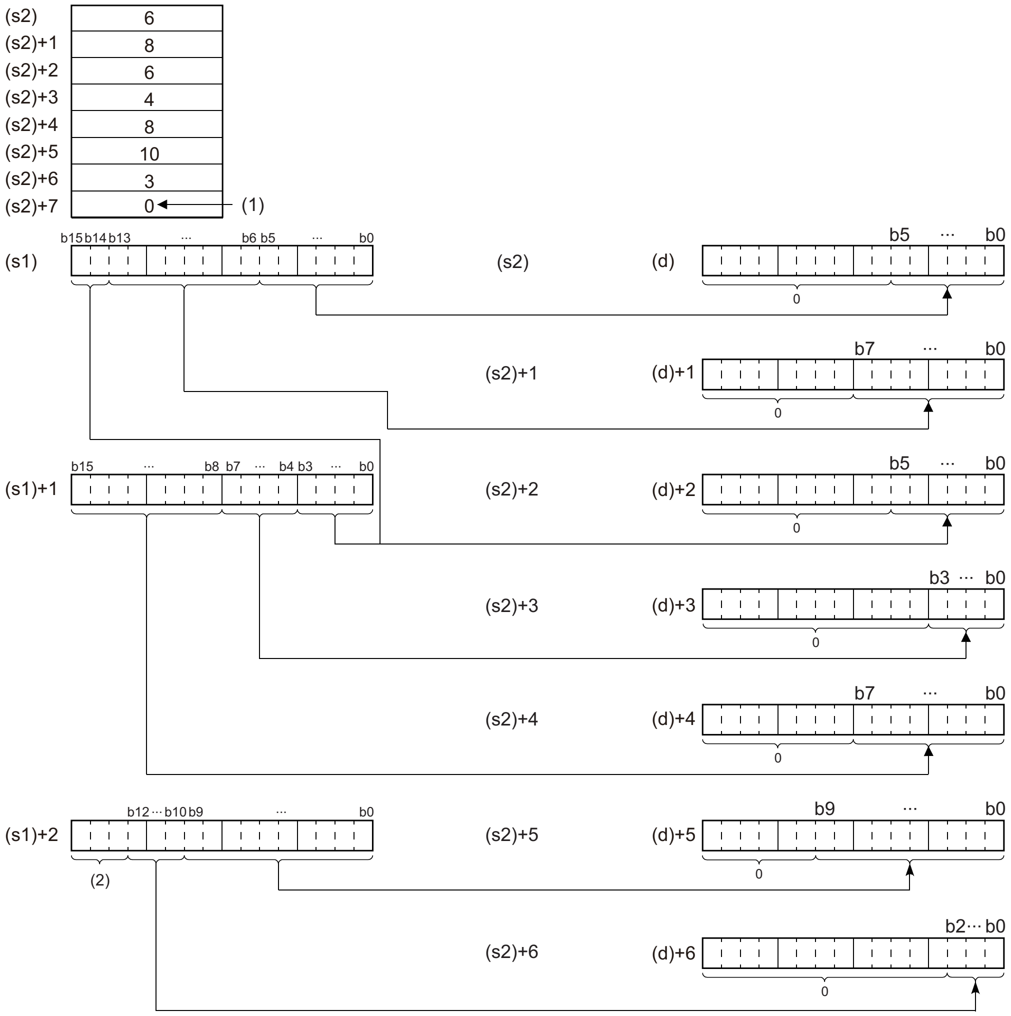
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions separate the bits of data in the device specified by (s1) and later in units of bits specified by (s2), and store the separated data in the device range specified by (d) and later.



(s2) to (s2)+6: Number of bits specified by (s2) to (s2)+6

(1) The value, 0, indicates the end of setting.

(2) Ignored.

- Specify the value 1 to 16 for (s2).
- The device areas from the one specified by (s2) to the one storing "0" are processed.
- Specify the devices so that the range of the device where the separation target data is stored ((s1) and later) and the range of the device for storing the separated data ((d) and later) do not overlap. If they overlap, a correct operation result may not be obtained.
- Do not overlap the device numbers that are specified by (s1), (s2), and (d).

Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3401H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 in the label or device area (between the specified device number and the last device number).

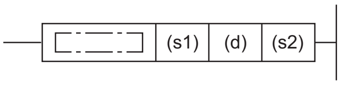
Combining data in units of bits

NUNI(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions combine the data in units of bits. (The number of bits can be specified as desired.)

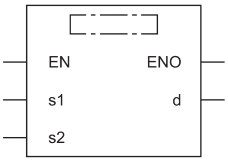
Ladder



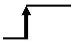
ST

ENO:=NUNI(EN,s1,s2,d);
ENO:=NUNIP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
NUNI	
NUNIP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the combined data	—	16-bit signed binary	ANY16[*1]
(s2)	Start device for storing the combination unit	—	16-bit signed binary	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

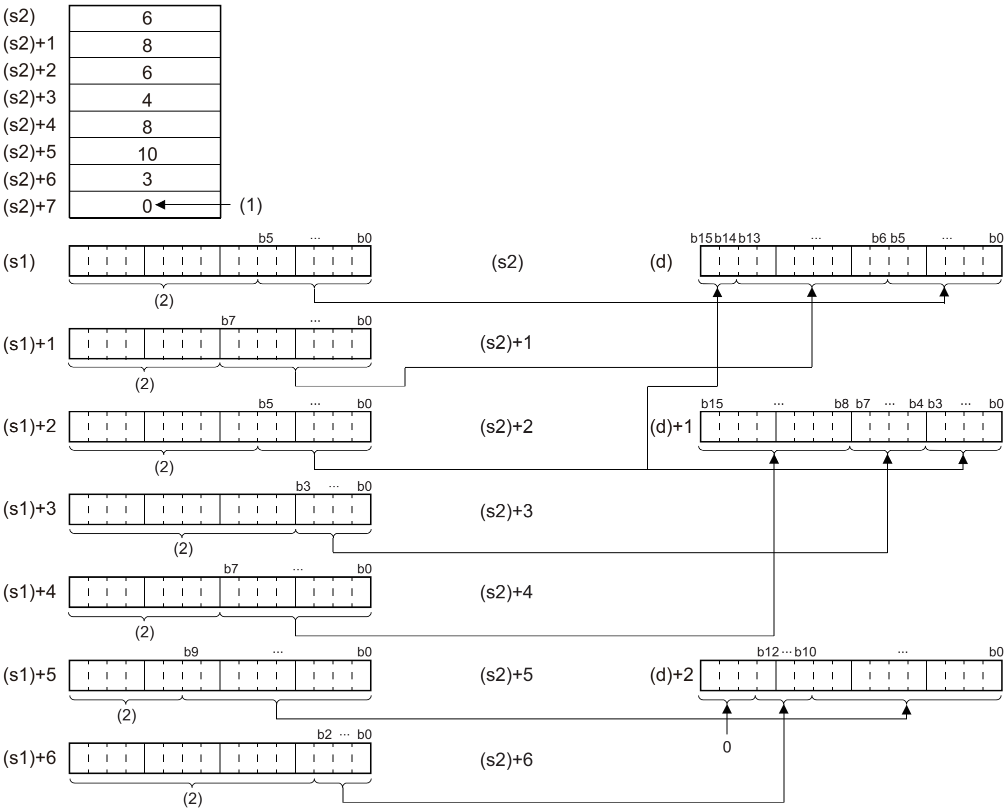
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions combine the bits of data in the device specified by (s1) and later in units of bits specified by (s2), and store the combined data in the device specified by (d) and later.



(s2) to (s2)+6: Number of bits specified by (s2) to (s2)+6

(1) The value, 0, indicates the end of setting.

(2) Ignored.

- Specify the value 1 to 16 for (s2).
- The device areas from the one specified by (s2) to the one storing "0" are processed.
- Specify the devices so that the range of the device where the combination target data is stored ((s1) and later) and the range of the device for storing the combined data ((d) and later) do not overlap. If they overlap, a correct operation result may not be obtained.
- Do not overlap the device numbers that are specified by (s1), (s2), and (d).

Operation error

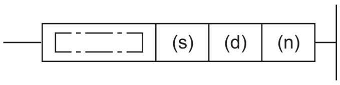
Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3401H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 in the label or device area (between the specified device number and the last device number).

Separating data in units of bytes

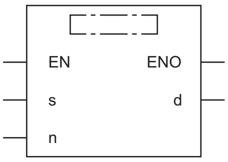
WTOB(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

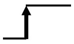
These instructions separate 16-bit binary data into (n) bytes.

Ladder	ST
	ENO:=WTOB(EN,s,n,d); ENO:=WTOBP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WTOB	
WTOBP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the separation target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16[*1]
(n)	Number of data bytes to be separated	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

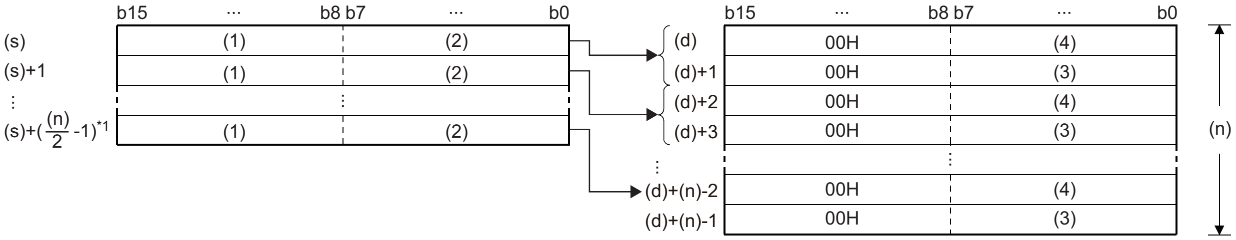
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions separate the 16-bit binary data in the device specified by (s) and later into (n) bytes, and store the separated data in the device specified by (d) and later.



(1) Upper byte

(2) Lower byte

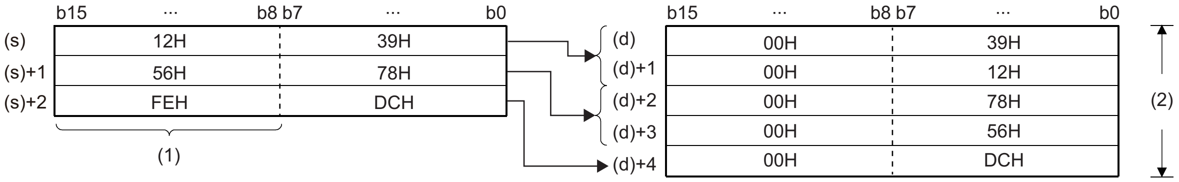
(3) Upper byte data

(4) Lower byte data

*1 Values after the decimal point are rounded up.

Ex.

When (n) is 5, the data in the device specified by (s) (upper 8 bits) to (s)+2 (lower 8 bits) are stored in the device specified by (d) to (d)+4.



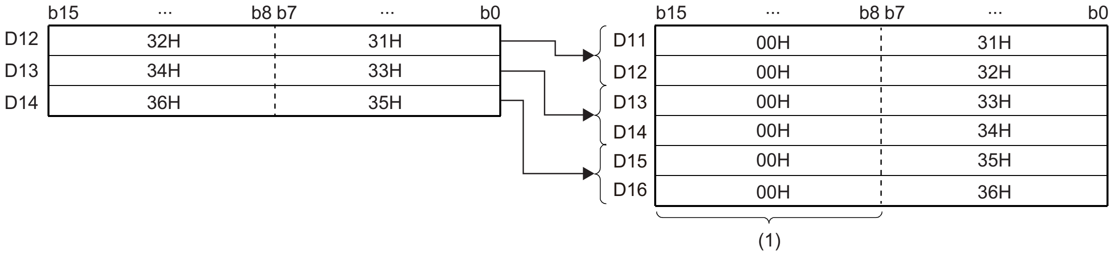
(1) The data, FEH, is ignored when (n) is 5.

(2) (n)=5

- Setting the number of bytes for (n) automatically determines the range of 16-bit binary data specified by (s) and the range of the device specified by (d) for storing the separated data.
- If (n) is 0, no processing is performed.
- The upper 8 bits of the device specified by (d) are automatically filled with 00Hs.

Ex.

When the byte data in D12 to D14 are stored in the lower 8 bits of D11 to D16



(1) 00H is automatically stored.

- Even when the ranges of the device where the separation target data is stored and the device for storing the separated data overlap, the processing is performed normally.

Range of the device where the separation target data is stored	Range of the device for storing the separated data
(s) to (s)+($\frac{(n)}{2}$-1)	(d)+0 to (d)+(n)-1

Operation error

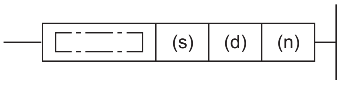
There is no operation error.

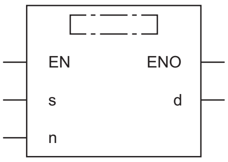
Combining data in units of bytes

BTOW(P)

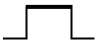
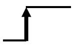
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions combine the lower 8 bits of 16-bit binary data in units of words.

Ladder	ST
	ENO:=BTOW(EN,s,n,d); ENO:=BTOWP(EN,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BTOW	
BTOWP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the combined data	—	16-bit signed binary	ANY16[*1]
(n)	Number of data bytes to be combined	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

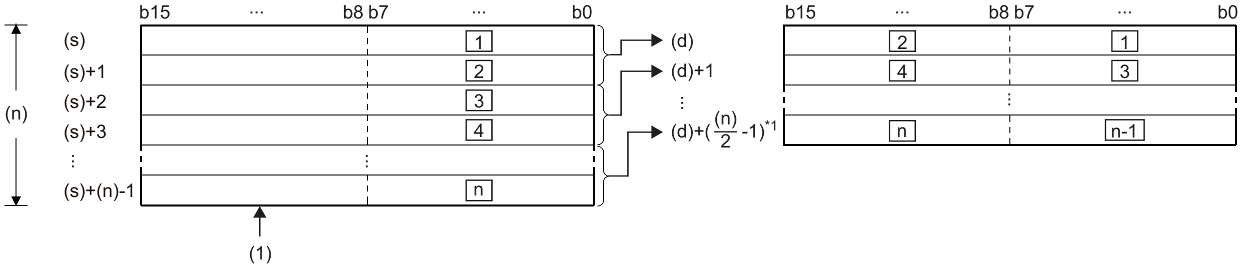
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions combine the (n) bytes of lower 8 bits of 16-bit binary data in the device specified by (s) and later in units of words, and store the combined data in the device specified by (d) and later.
- The (n) bytes of upper 8 bits of 16-bit binary data in the device specified by (s) and later are ignored. When (n) is an odd number, the upper 8 bits of the device where 'n'th-byte data is stored are filled with 0s.



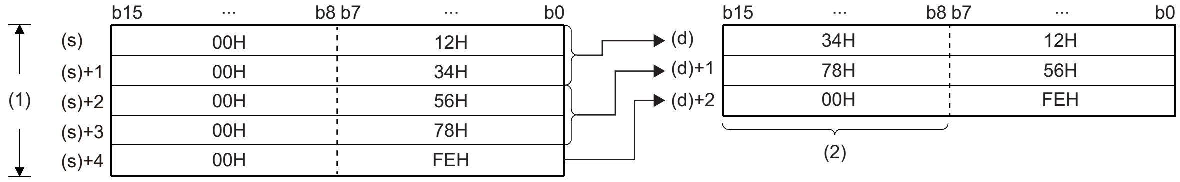
☐: ☐th byte data

(1) The upper byte data are ignored.

[1] Values after the decimal point are rounded up.

Ex.

When (n) is 5, the lower 8 bits of the data in the device specified by (s) to (s)+4 are combined and stored in the device specified by (d) to (d)+2.



(1) (n)=5

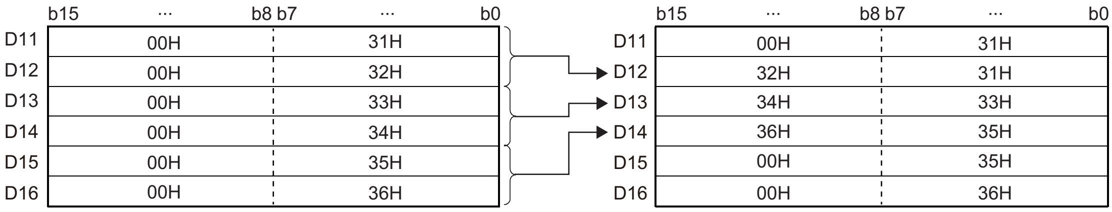
(2) Filled with 00H.

- Setting the number of bytes for (n) automatically determines the range of byte data in the device specified by (s) and the range of the device specified by (d) for storing the combined data.
- If (n) is 0, no processing is performed.
- The upper 8 bits in the device specified by (s) and later are ignored, and only the lower 8 bits are processed.
- Even when the ranges of the device where the combination target data is stored and the device for storing the combined data overlap, the processing is performed normally.

Range of the device where the combination target data is stored	Range of the device for storing the combined data
(s)+0 to (s)+(n)-1	(d) to $(d)+(\frac{(n)}{2}-1)$

Ex.

When the lower 8 bits of D11 to D16 are stored in D12 to D14



Operation error

There is no operation error.

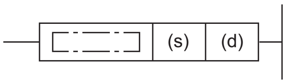
6.7 Data Transfer Instructions

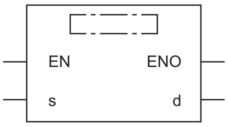
Transferring 16-bit binary data

MOV(P)

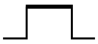
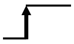
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions transfer the 16-bit binary data in the device specified.

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

FBD/LD


■Execution condition

Instruction	Execution condition
MOV	
MOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source data or the number of the device where the transfer source data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination device number	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

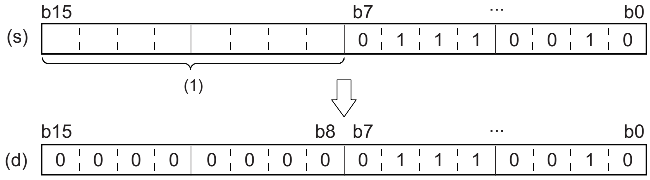
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions transfer 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

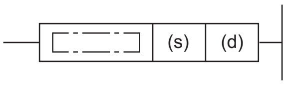
There is no operation error.

Transferring 32-bit binary data

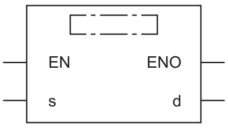
DMOV(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

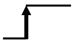
These instructions transfer the 32-bit binary data in the device specified.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DMOV	
DMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source data or the number of the device where the transfer source data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination device number	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

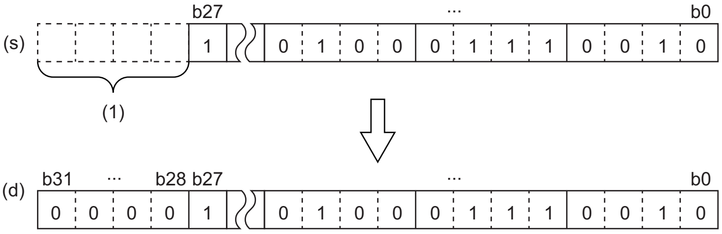
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions transfer 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

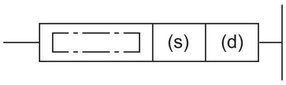
There is no operation error.

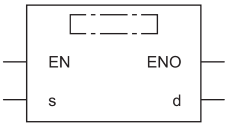
Inverting and transferring 16-bit binary data

CML(P)

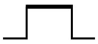
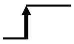
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions invert the specified 16-bit binary data bit by bit, and transfer the inverted data.

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

FBD/LD


■Execution condition

Instruction	Execution condition
CML	
CMLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Number of the device for storing the inverted data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

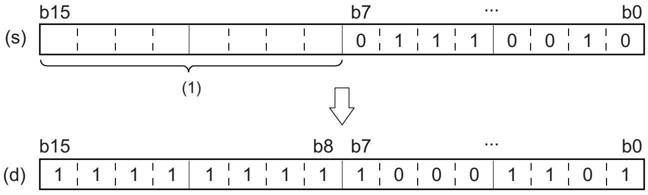
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions invert 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

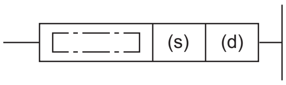
There is no operation error.

Inverting and transferring 32-bit binary data

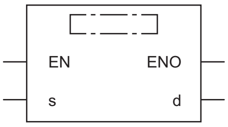
DCML(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

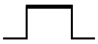
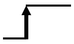
These instructions invert the specified 32-bit binary data bit by bit, and transfer the inverted data.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DCML	
DCMLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Number of the device for storing the inverted data	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

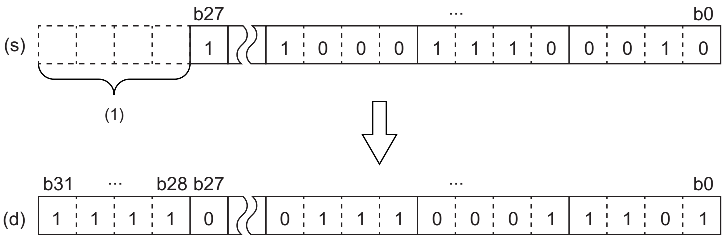
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

Processing details

- These instructions invert 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.

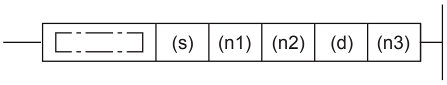
Shifting data in units of 4 bits

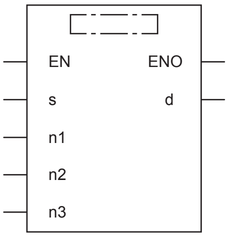
SMOV(P)

RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

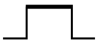
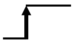
- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

These instructions distribute and combine data in units of 4 bits.

Ladder	ST
	ENO:=SMOV(EN,s,n1,n2,n3,d); ENO:=SMOVP(EN,s,n1,n2,n3,d);

FBD/LD


■Execution condition

Instruction	Execution condition
SMOV	
SMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the specified units of data is stored	—	16-bit signed binary	ANY16
(n1)[*1]	Start position where the data to be shifted is stored	1 to 4	16-bit unsigned binary	ANY16_U
(n2)[*1]	Number of units to be shifted	1 to 4	16-bit unsigned binary	ANY16_U
(d)	Device where the shifted data is stored	—	16-bit signed binary	ANY16
(n3)[*1]	Start unit position of shit destination	1 to 4	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1) and (n2)≤(n3).

■Applicable devices

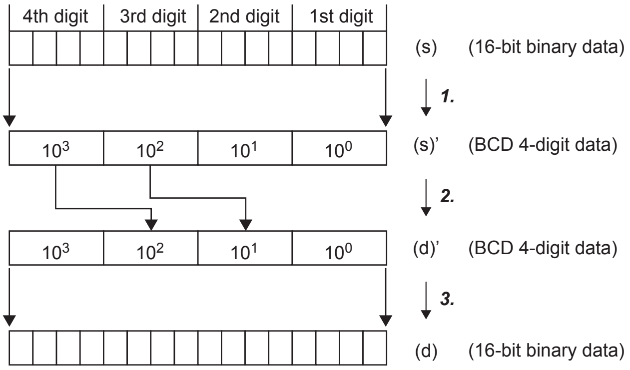
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n3)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- When SM773 is OFF, the contents of (s) and (d) are converted to 4-digit BCD (0000 to 9999). After the data in lower (n2) digits from the (n1)th digit is transferred (combined) to (d) starting from the (n3)th digit, the data is converted into BIN and then stored in (d).

In the case of (n1) = 4, (n2)= 2 and (n3) = 3



1. Conversion

(s) is converted from binary to BCD data.

2. Shifting

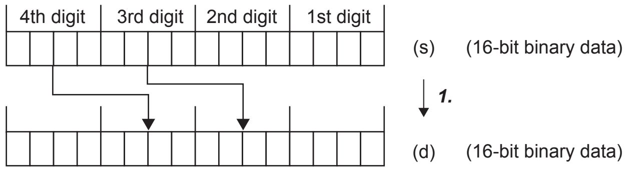
Lower (n2) digits starting from the (n1)th digit are transferred (combined) to (d)' starting from the (n3)th digit. In the case of this example, the first and fourth digits of (d)' are not affected even if data is transferred from (s)'.

3. Conversion

The combined data (BCD) is converted into binary, and stored to (d).

- When SM773 is on, conversion from BIN to BCD is not performed and data are shifted in units of 4 bits.

In the case of (n1) = 4, (n2)= 2 and (n3) = 3



1. Shifting data in units of 4 bits

Lower (n2)×4-bit data from (n1)×4th bit is transferred (combined) by lower (n2)×4-bit data from (n3)×4th bit of (d). In the case of this example, the first and fourth digits of (d) are not affected even if data is transferred from (s).

- When digit specification of bit is used for (n1), the number of digits becomes the specified number of digits × 4 bits from the start bit device.

Ex.

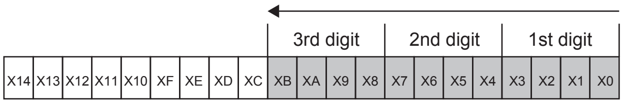
To specify "3 digits × 4 bits" from X0:

[SMOV K3X0 K3 K2 K2X100 K2]

(1) (2)

(1) Specified constant

(2) Start bit device



Digit specification denotes the "specified number of digits 3×4 bits" = 12 bits from the start bit device X0.

Operation error

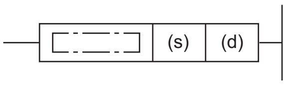
Error code (SD0)	Description
3405H	Any one of (n1), (n2), or (n3) is other than 1 to 4.
	Either (s) or (d) is other than 0 to 9999 when SM773 is OFF.
	(n2) is greater than (n1) or (n3).

Inverting and transferring 1-bit data

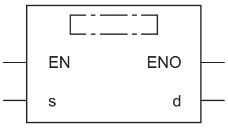
CMLB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

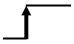
These instructions invert the specified bit data, and transfer the inverted data.

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

FBD/LD



■Execution condition

Instruction	Execution condition
CMLB	
CMLBP	

Setting data

■Description, range, data type

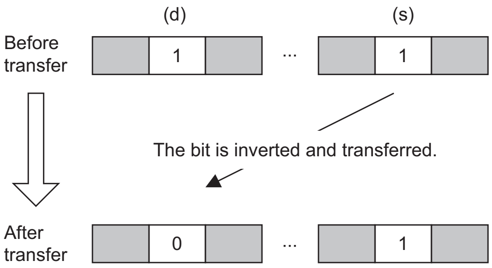
Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	—	Bit	ANY_BOOL
(d)	Transfer destination device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	—	—	○	—	○	—	—	—	—
(d)	○	○	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions invert the bit data in the device specified by (s), and transfer the inverted data to the device specified by (d).



Operation error

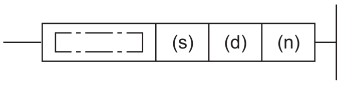
There is no operation error.

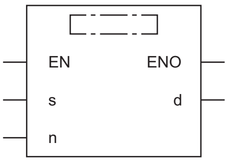
Transferring 16-bit binary data block (16 bits)

BMOV(P)

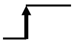
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions batch-transfer the (n) points (0 to 65535) of 16-bit binary data starting from the device specified.

Ladder	ST
	ENO:=BMOV(EN,s,n,d); ENO:=BMOVP(EN,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BMOV	
BMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the transfer target data is stored	—	16-bit signed binary	ANY16[*1][*2]
(d)	Transfer destination start device	—	16-bit signed binary	ANY16[*1][*2]
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16[*2]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

*2 Data types other than INT and WORD are available to the engineering tool with version "1.030G" or later (except for BOOL and POINTER data type). (Available data types are the same as the ones to GX Works2.) Note that available data types are INT and WORD (in range of ANY16) when "Yes" is selected for "Check the data type of instruction argument" of option settings.
For the option settings, refer to the following.
🕮 GX Works3 Operating Manual

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

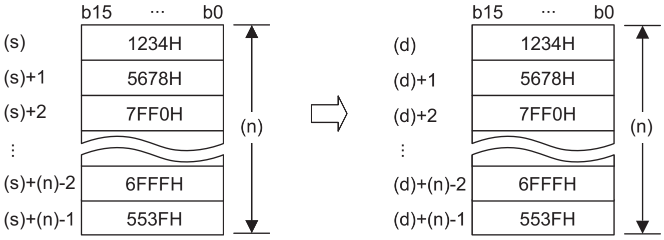
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—
(n)	○	○	○

Processing details

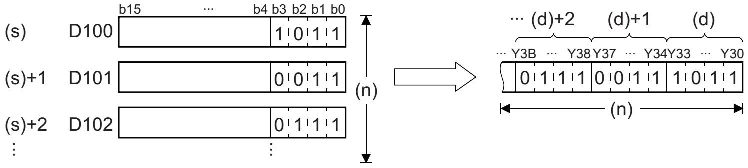
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- Data can be transferred even when the transfer source device and destination device overlap. A transfer to smaller device numbers begins from the device specified by (s), and a transfer to larger device numbers begins from the device specified by (s)+(n)-1.
- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

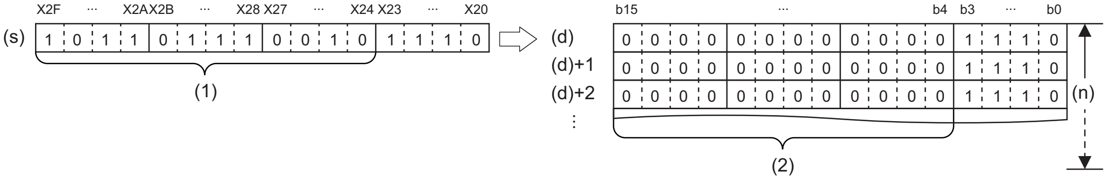
When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When (s) is a digit-specified bit device and (d) is a word device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1X20 is specified in (s), the data is transferred to the lower 4 bits of the word device specified by (d).



(1) Ignored.

(2) Filled with 0s.

- When both (s) and (d) are bit devices, set the same number of digits for both devices.
- To use the link direct device, module access device, or CPU buffer memory access device for (s) or (d), specify it only for one of the devices. Note that the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification can be specified by both (s) and (d).

Operation error

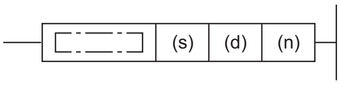
Error code (SD0)	Description
3420H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d). However, this is not true when the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified is specified.

Transferring 16-bit binary data block (32 bits)

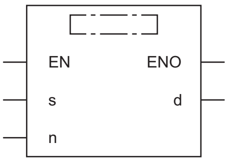
BMOVL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

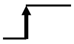
These instructions batch-transfer the (n) points (1 to 4294967295) of 16-bit binary data starting from the device specified.

Ladder	ST
 (s) (d) (n)	ENO:=BMOVL(EN,s,n,d); ENO:=BMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BMOVL	
BMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the transfer target data is stored	—	16-bit signed binary	ANY16[*1]
(d)	Transfer destination start device	—	16-bit signed binary	ANY16[*1]
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

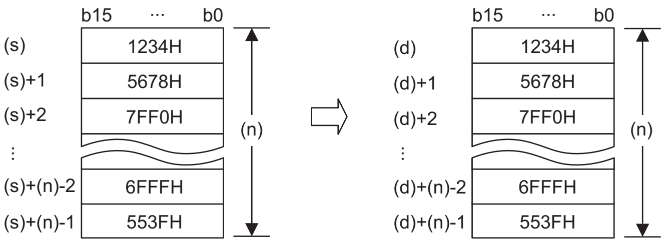
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—
(n)	○	○	○

Processing details

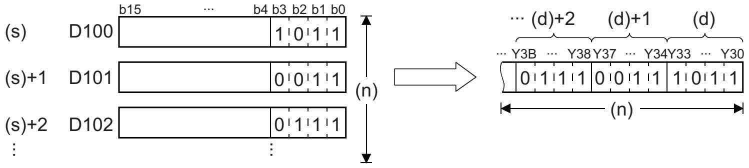
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- Data can be transferred even when the transfer source device and destination device overlap. A transfer to smaller device numbers begins from the device specified by (s), and a transfer to larger device numbers begins from the device specified by (s)+(n)-1.
- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When both (s) and (d) are bit devices, set the same number of digits for both devices.
- To use the link direct device, module access device, or CPU buffer memory access device for (s) or (d), specify it only for one of the devices. Note that the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification can be specified by both (s) and (d).

Operation error

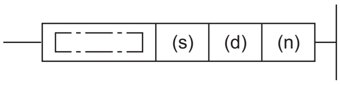
Error code (SD0)	Description
3420H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d). However, this is not true when the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified is specified.

Transferring the same 16-bit binary data block (16 bits)

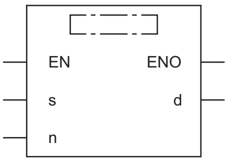
FMOV(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

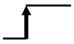
These instructions transfer 16-bit binary data to the (n) points (0 to 65535) starting from the device specified.

Ladder	ST
	ENO:=FMOV(EN,s,n,d); ENO:=FMOVP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FMOV	
FMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the device where the transfer target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination start device	—	16-bit signed binary	ANY16[*1]
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

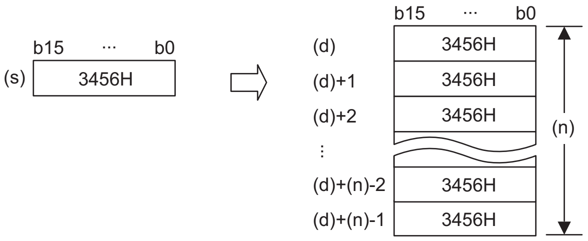
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

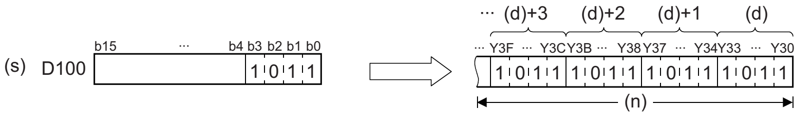
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

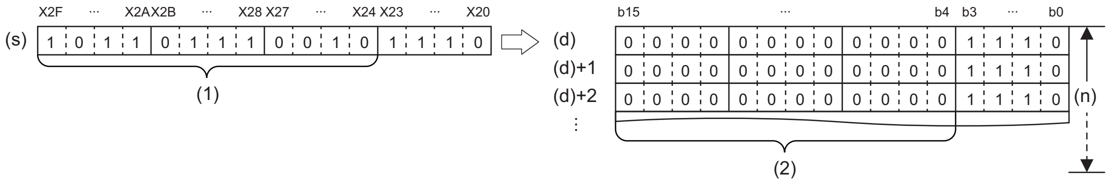
When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When (s) is a digit-specified bit device and (d) is a word device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1X20 is specified in (s), the data is transferred to the lower 4 bits of the word device specified by (d).



(1) Ignored.
(2) Filled with 0s.

- When both (s) and (d) are bit devices, set the same number of digits for both devices.

Operation error

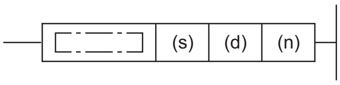
There is no operation error.

Transferring the same 16-bit binary data block (32 bits)

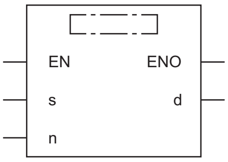
FMOVL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

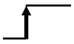
These instructions transfer 16-bit binary data to the (n) points (1 to 4294967295) starting from the device specified.

Ladder	ST
	ENO:=FMOVL(EN,s,n,d); ENO:=FMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FMOVL	
FMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the device where the transfer target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination start device	—	16-bit signed binary	ANY16[*1]
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

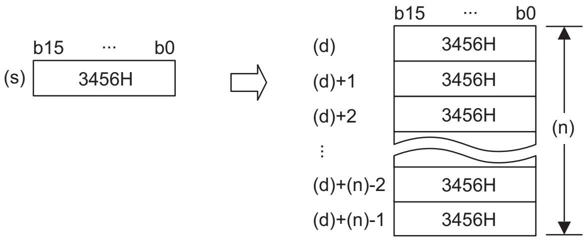
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

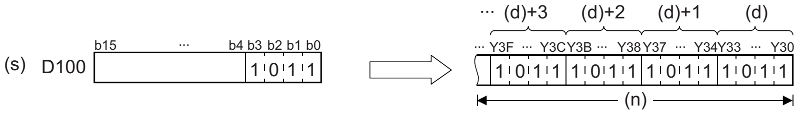
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When both (s) and (d) are bit devices, set the same number of digits for both devices.

Operation error

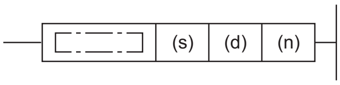
There is no operation error.

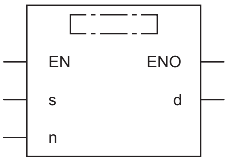
Transferring the same 32-bit binary data block (16 bits)

DFMOV(P)

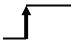
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions transfer 32-bit binary data to the (n) points (1 to 65535) starting from the device specified.

Ladder	ST
	ENO:=DFMOV(EN,s,n,d); ENO:=DFMOVP(EN,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DFMOV	
DFMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the start device where the transfer target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination start device	—	32-bit signed binary	ANY32[*1]
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

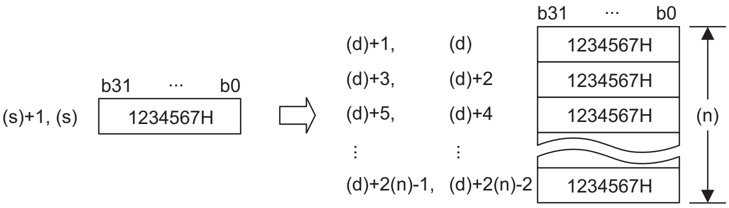
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

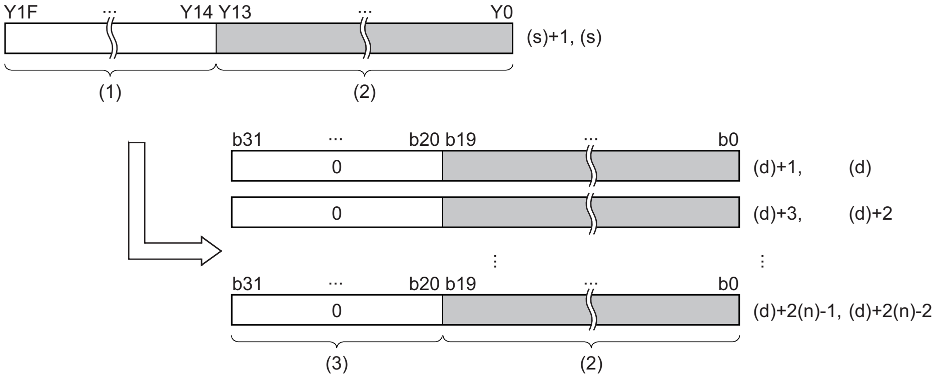
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y0 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.

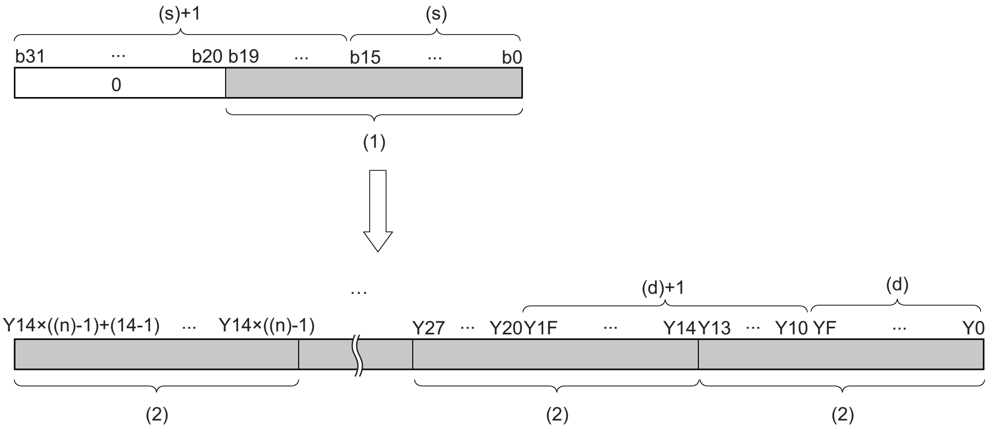


(1) Ignored.

(2) Data of 20 bits (5 digits)

(3) Filled with 0s.

- When the number of digits is specified for the data in the device specified by (d), the data corresponding to the specified digits are transferred. When K5Y0 is specified in (d), the lower 20 bits (five digits) of the word device specified by (s) are transferred. When the number of digits is specified for both data in the devices specified by (s) and (d), the data corresponding to the digits specified in (d) are transferred.



(1) Data (20 bits (5 digits) specified by (d))

(2) Data of 20 bits (5 digits)

- If (n) is 0, no processing is performed.

Operation error

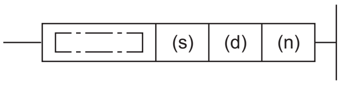
There is no operation error.

Transferring the same 32-bit binary data block (32 bits)

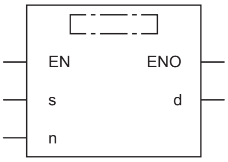
DFMOVL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

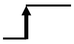
These instructions transfer 32-bit binary data to the (n) points (1 to 4294967295) starting from the device specified.

Ladder	ST
	ENO:=DFMOVL(EN,s,n,d); ENO:=DFMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DFMOVL	
DFMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the start device where the transfer target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination start device	—	32-bit signed binary	ANY32[*1]
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

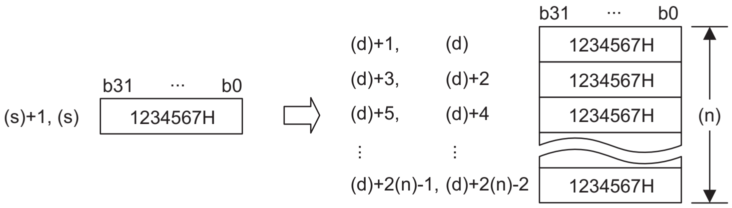
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

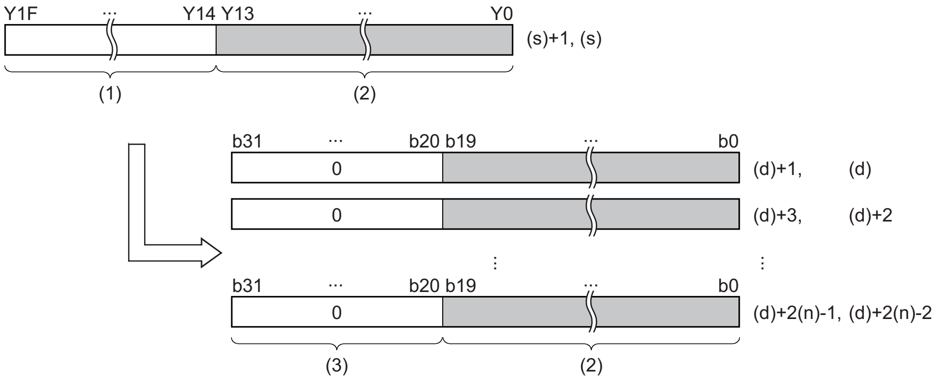
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y0 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.

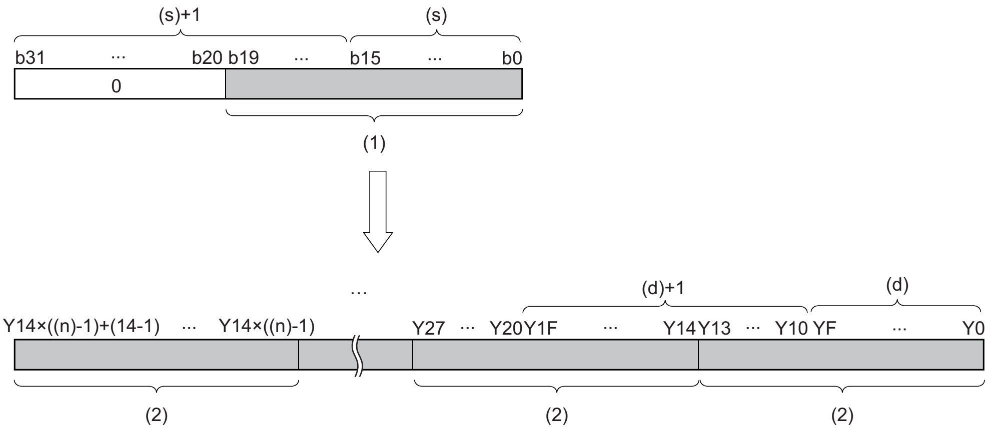


(1) Ignored.

(2) Data of 20 bits (5 digits)

(3) Filled with 0s.

- When the number of digits is specified for the data in the device specified by (d), the data corresponding to the specified digits are transferred. When K5Y0 is specified in (d), the lower 20 bits (five digits) of the word device specified by (s) are transferred. When the number of digits is specified for both data in the devices specified by (s) and (d), the data corresponding to the digits specified in (d) are transferred.



(1) Data (20 bits (5 digits) specified by (d))

(2) Data of 20 bits (5 digits)

- If (n) is 0, no processing is performed.

Operation error

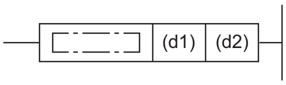
There is no operation error.

Exchanging 16-bit binary data

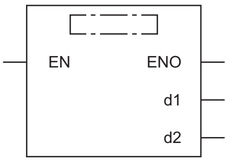
XCH(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

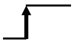
These instructions exchange the 16-bit binary data specified.

Ladder	ST
	ENO:=XCH(EN,d1,d2); ENO:=XCHP(EN,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
XCH	
XCHP	

Setting data

■Description, range, data type

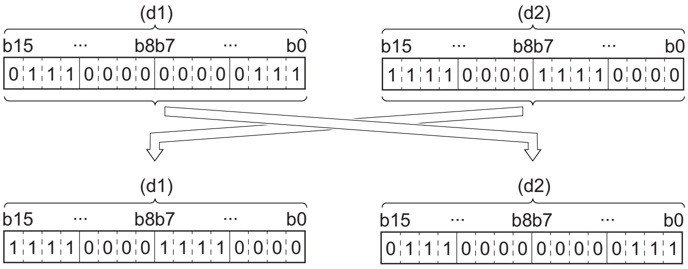
Operand	Description	Range	Data type	Data type (label)
(d1)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
(d2)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d1)	○	○	○	○	○	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instruction exchange the 16-bit binary data between the devices specified by (d1) and (d2).



Operation error

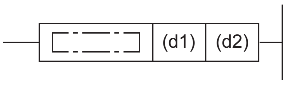
There is no operation error.

Exchanging 32-bit binary data

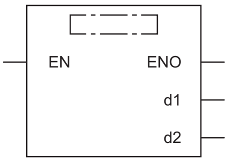
DXCH(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

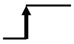
These instructions exchange the 32-bit binary data specified.

Ladder	ST
	ENO:=DXCH(EN,d1,d2); ENO:=DXCHP(EN,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DXCH	
DXCHP	

Setting data

■Description, range, data type

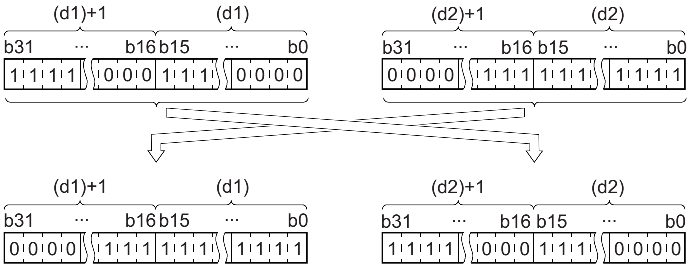
Operand	Description	Range	Data type	Data type (label)
(d1)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
(d2)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d1)	○	○	○	○	○	○	○	○	—	—	—	—
(d2)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instruction exchange the 32-bit binary data between the device ranges specified by (d1) and (d2).



Operation error

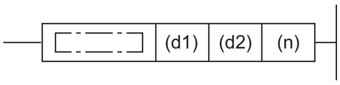
There is no operation error.

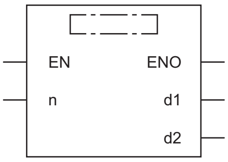
Exchanging 16-bit binary block data

BXCH(P)

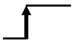
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions exchange the (n) points of 16-bit binary data starting from the devices specified.

Ladder	ST
	ENO:=BXCH(EN,n,d1,d2); ENO:=BXCHP(EN,n,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
BXCH	
BXCHP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d1)	Start device where the exchange target data is stored	—	Word	ANY16[*1]
(d2)	Start device where the exchange target data is stored	—	Word	ANY16[*1]
(n)	Number of exchange data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

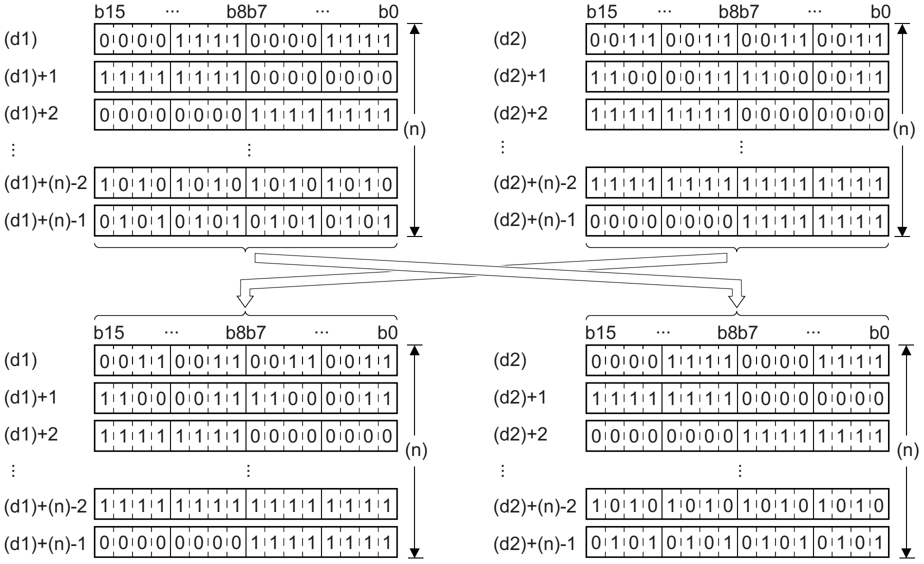
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions exchange the (n) points of 16-bit binary data starting from the device specified by (d1) with the (n) points of 16-bit binary data starting from the device specified by (d2).



Operation error

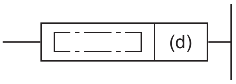
Error code (SD0)	Description
2821H	The device ranges specified by (d1) and (d2) are overlapping.

Exchanging the upper and lower bytes of 16-bit binary data

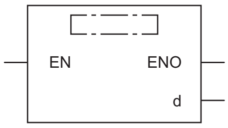
SWAP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

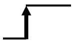
These instructions exchange upper and lower 8-bit data in the specified device.

Ladder	ST
	ENO:=SWAP(EN,d); ENO:=SWAPP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SWAP	
SWAPP	

Setting data

■Description, range, data type

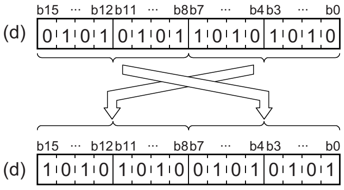
Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the data whose upper and lower 8-bit data is exchanged	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- The device specified by (d) exchanges its upper and lower 8-bit data.



Operation error

There is no operation error.

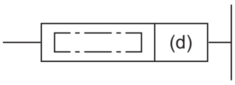
Exchanging the upper and lower bytes of 32-bit binary data

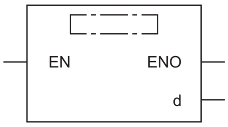
DSWAP(P)

RnCPU **RnENCPU** ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

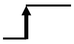
- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

These instructions exchange upper and lower 8-bit data in the specified device.

Ladder	ST
 (d)	ENO:=DSWAP(EN,d); ENO:=DSWAPP(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DSWAP	
DSWAPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the data whose upper and lower 8-bit data is exchanged	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

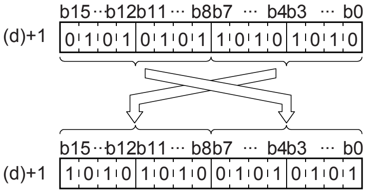
■Applicable devices

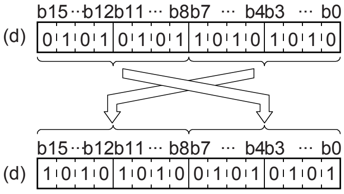
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- The device specified by (d) exchanges its upper and lower 8-bit data.





Operation error

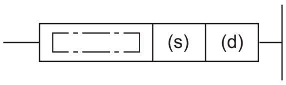
There is no operation error.

Transferring 1-bit data

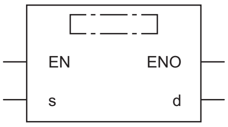
MOVB(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

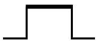
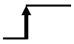
These instructions transfer the specified 1-bit data.

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

FBD/LD



■Execution condition

Instruction	Execution condition
MOVB	
MOVBP	

Setting data

■Description, range, data type

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

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

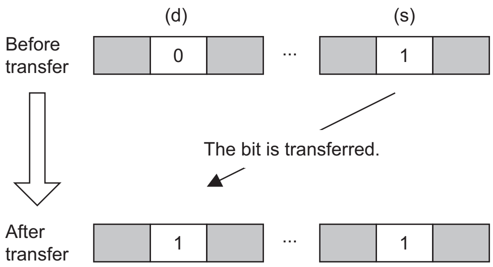
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	—	—	○	—	○	—	—	—	—
(d)	○	○	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—

Processing details

- These instructions transfer the bit data in the device specified by (s) to the device specified by (d).



Operation error

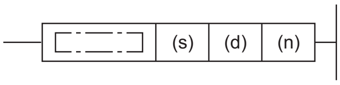
There is no operation error.

Transferring n-bit data

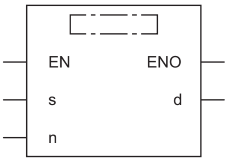
BLKMOVB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

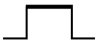
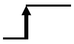
These instructions batch-transfer the (n) points of bit data.

Ladder	ST
	ENO:=BLKMOVB(EN,s,n,d); ENO:=BLKMOVBP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BLKMOVB	
BLKMOVBP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source block data	—	Bit	ANY_BOOL[*1]
(d)	Transfer destination block data	—	Bit	ANY_BOOL[*1]
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

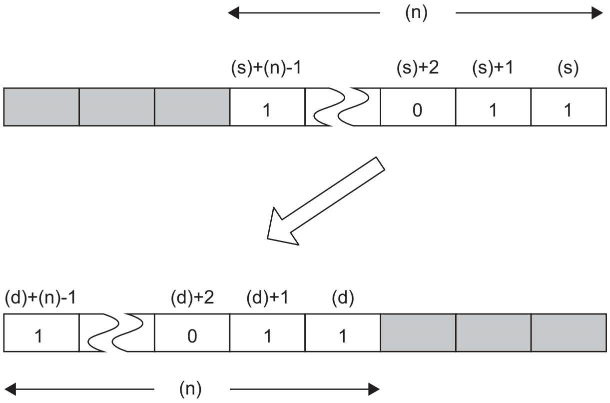
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions batch-transfer the (n) points of bit data starting from the device specified by (s) to the (n) points of bits starting from the device specified by (d).
- Data can be transferred even when the transfer source device and destination device overlap.



Operation error

There is no operation error.

7 APPLICATION INSTRUCTIONS

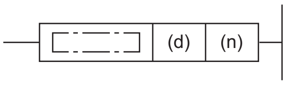
7.1 Rotation Instructions

Rotating 16-bit binary data to the right

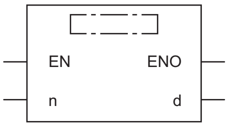
ROR(P), RCR(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

- ROR(P): These instructions rotate the 16-bit binary data to the right by (n) bit(s), excluding the carry flag.
- RCR(P): These instructions rotate the 16-bit binary data to the right by (n) bit(s), including the carry flag.

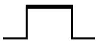
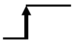
Ladder	ST[*1]
	ENO:=RORP(EN,n,d); ENO:=RCR(EN,n,d); ENO:=RCRP(EN,n,d);

FBD/LD[*1]



*1 The ROR instruction does not support the ST and FBD/LD. Use the standard function, ROR.
☞ Page 1925 ROR(_E)

■Execution condition

Instruction	Execution condition
ROR RCR	
RORP RCRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Rotation target device	—	16-bit signed binary	ANY16
(n)	Number of bits to be rotated	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

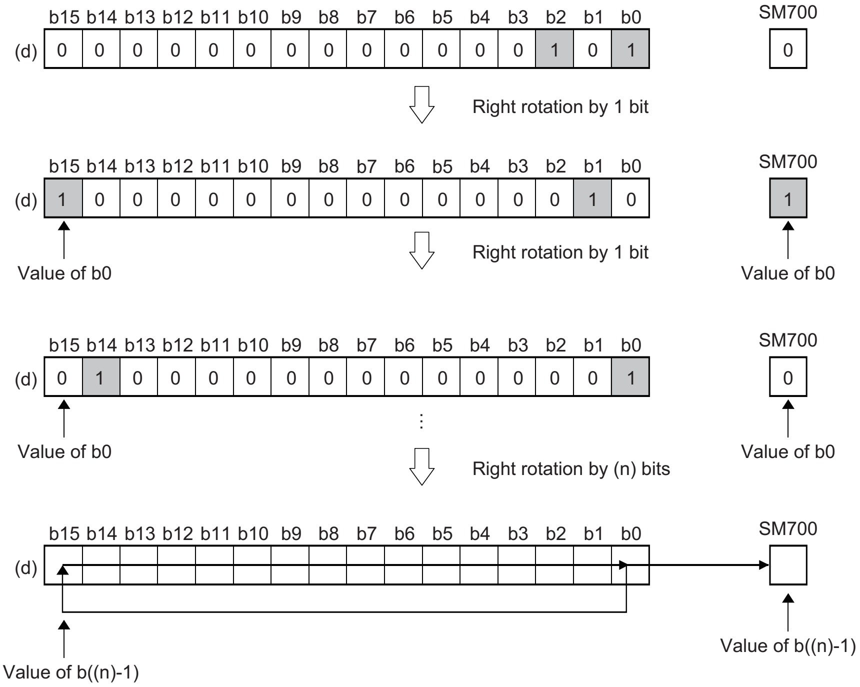
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROR(P)

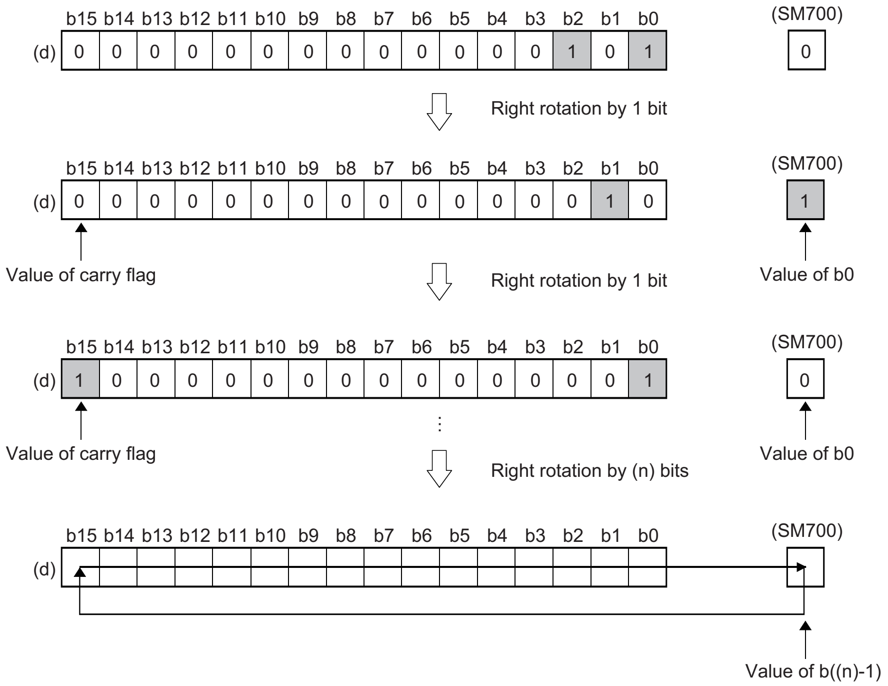
- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

■RCR(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the RCR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

Operation error

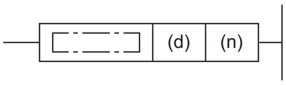
There is no operation error.

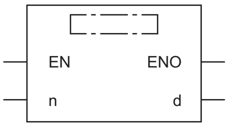
Rotating 16-bit binary data to the left

ROL(P), RCL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

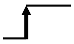
- ROL(P): These instructions rotate the 16-bit binary data to the left by (n) bit(s), excluding the carry flag.
- RCL(P): These instructions rotate the 16-bit binary data to the left by (n) bit(s), including the carry flag.

Ladder	ST[1]
	ENO:=ROLP(EN,n,d); ENO:=RCL(EN,n,d); ENO:=RCLP(EN,n,d);

FBD/LD[1]


[1] The ROL instruction does not support the ST and FBD/LD. Use the standard function, ROL.
☞ Page 1923 ROL(_E)

■Execution condition

Instruction	Execution condition
ROL RCL	
ROLP RCLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Rotation target device	—	16-bit signed binary	ANY16
(n)	Number of bits to be rotated	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

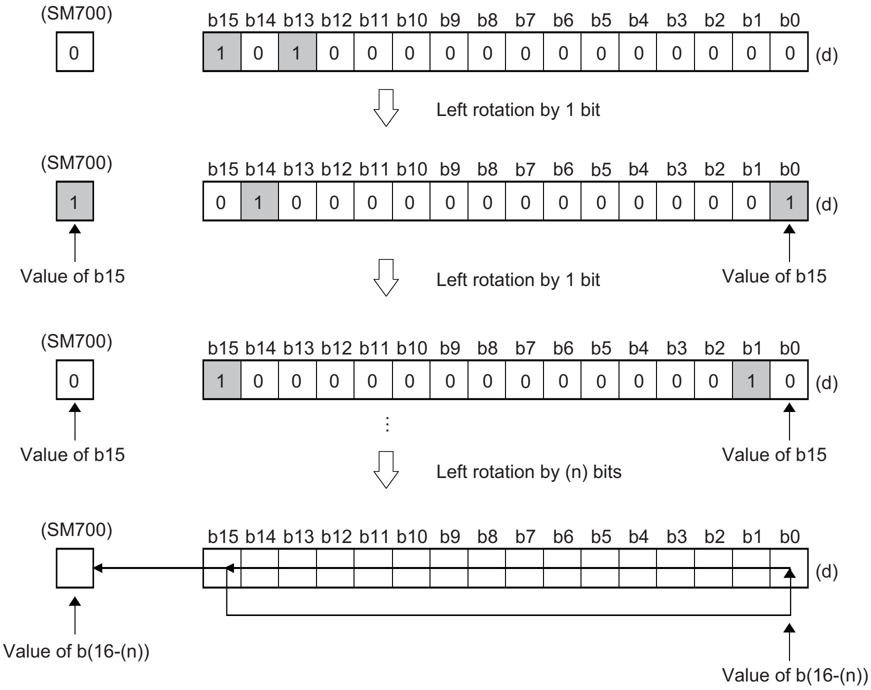
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROL(P)

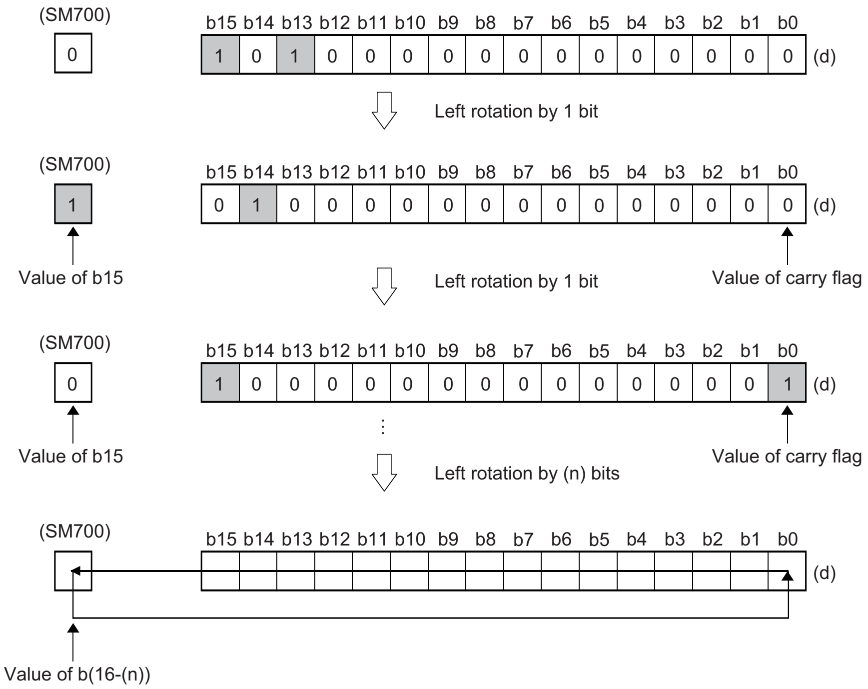
- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated to the left because 18 divided by 16 equals 1 with a remainder of 2.

■RCL(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the RCL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated to the left because 18 divided by 16 equals 1 with a remainder of 2.

Operation error

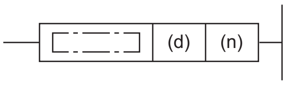
There is no operation error.

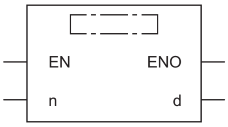
Rotating 32-bit binary data to the right

DROR(P), DRCR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

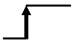
- DROR(P): These instructions rotate the 32-bit binary data to the right by (n) bit(s), excluding the carry flag.
- DRCR(P): These instructions rotate the 32-bit binary data to the right by (n) bit(s), including the carry flag.

Ladder	ST[*1]
	ENO:=DRORP(EN,n,d); ENO:=DRCR(EN,n,d); ENO:=DRCRP(EN,n,d);

FBD/LD[*1]


[*1] The DROR instruction does not support the ST and FBD/LD. Use the standard function, ROR.
☞ Page 1925 ROR(_E)

■Execution condition

Instruction	Execution condition
DROR DRCR	
DRORP DRCRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the rotation target data is stored	—	32-bit signed binary	ANY32
(n)	Number of bits to be rotated	0 to 31	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

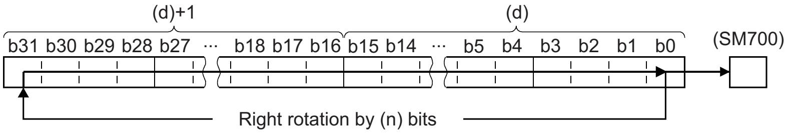
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROR(P)

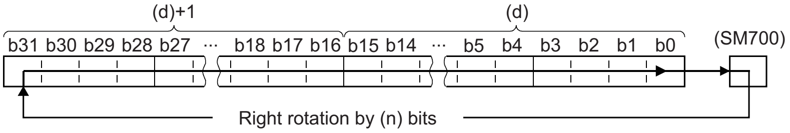
- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

■DRCR(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DRCR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

Operation error

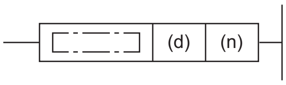
There is no operation error.

Rotating 32-bit binary data to the left

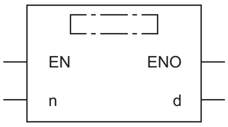
DROL(P), DRCL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

- DROL(P): These instructions rotate the 32-bit binary data to the left by (n) bit(s), excluding the carry flag.
- DRCL(P): These instructions rotate the 32-bit binary data to the left by (n) bit(s), including the carry flag.

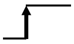
Ladder	ST[1]
	ENO:=DROLP(EN,n,d); ENO:=DRCL(EN,n,d); ENO:=DRCLP(EN,n,d);

FBD/LD[1]



[1] The DROL instruction does not support the ST and FBD/LD. Use the standard function, ROL.
☞ Page 1923 ROL(_E)

■Execution condition

Instruction	Execution condition
DROL DRCL	
DROLP DRCLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the rotation target data is stored	—	32-bit signed binary	ANY32
(n)	Number of bits to be rotated	0 to 31	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

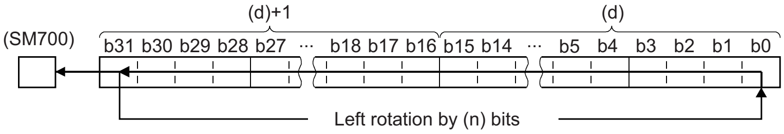
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROL(P)

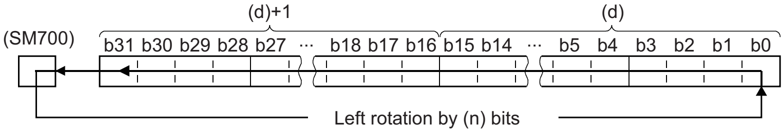
- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated to the left because 34 divided by 32 equals 1 with a remainder of 2.

■DRCL(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DRCL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated to the left because 34 divided by 32 equals 1 with a remainder of 2.

Operation error

There is no operation error.

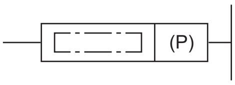
7.2 Program Branch Instructions

Pointer branch

CJ, SCJ, JMP

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

- CJ: This instruction executes the program specified by the pointer number within the same program file.
- SCJ: This instruction executes the program specified by the pointer number within the same program file starting with the next scan.
- JMP: This instruction unconditionally executes the program specified by the pointer number within the same program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
CJ SCJ	
JMP	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Pointer number of the jump destination	—	Device name	POINTER

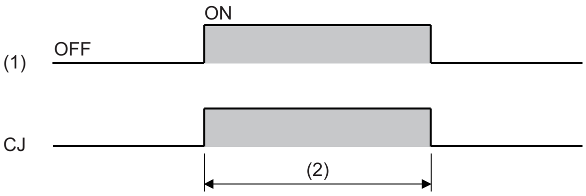
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, 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	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

■CJ

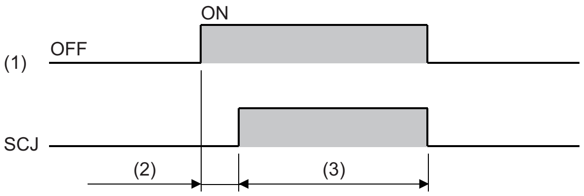
- This instruction executes the program specified by the pointer number within the same program file when the execution command is on.
- When the execution command is off, the program in the next step is executed.



(1) Execution command
(2) Executed in each scan

■SCJ

- This instruction executes the program specified by the pointer number within the same program file starting with the scan immediately after the execution command turns on.
- When the execution command is off or turned off, the program in the next step is executed.



(1) Execution command

(2) One scan

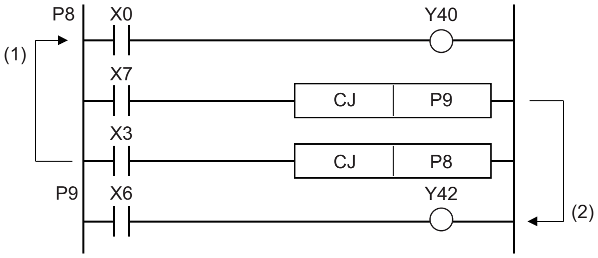
(3) Executed in each scan

■JMP

- This instruction unconditionally executes the program specified by the pointer number within the same program file.

Precautions

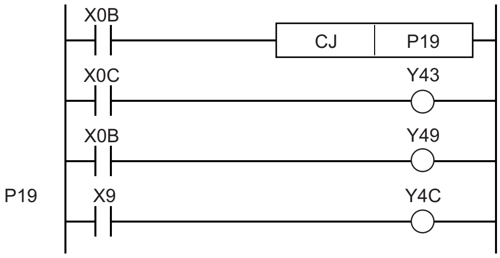
- If the timer with its coil on is skipped by these instructions, time cannot be measured correctly.
- If the OUT instruction is skipped by these instructions, the scan time will be shortened.
- If these instructions specify and jump to the program with a bigger step number, the scan time will be shortened.
- These instructions can specify and jump to the program with a smaller step number. In this case, consider a method to exit a loop so that the watchdog timer does not time out.



(1) While X3 is on, the loop is repeated.

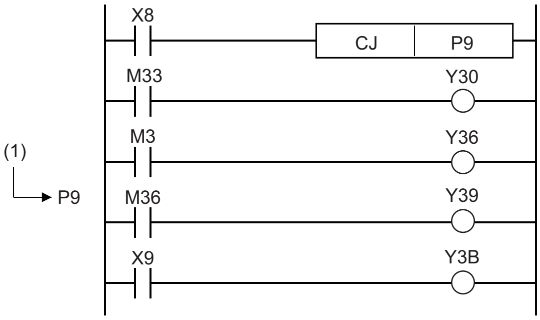
(2) To exit the loop, turn on X7.

- The value in the device skipped with these instructions remains unchanged.



When XB turns on, the program jumps to the label, P19.
Y43 and Y49 remain unchanged even if XB and XC turn on/off during the execution of the CJ instruction.

- A label (P□) occupies one step.



(1) A label occupies one step.

- Only the pointer numbers within the same program file can be specified.
- If the program jumps to the pointer number within the skip range, the programs of the jump destination pointer number and later are executed.

Operation error

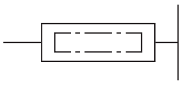
Error code (SD0)	Description
3380H	The pointer number specified by (P) is not set before the END instruction.
	The pointer number specified by (P) is not used as a label in the same program.
	The pointer number specified by (P) is a global pointer in another program.

Jumping to END

GOEND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction jumps the program to the FEND or END instruction within the same program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
GOEND	

Processing details

- This instruction jumps the program to the FEND or END instruction within the same program file.

Operation error

Error code (SD0)	Description
3340H	After execution of the FOR instruction, the GOEND instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P) or ECALL(P) instruction, the GOEND instruction is executed before the RET instruction.
33A1H	The GOEND instruction is executed before the IRET instruction in the interrupt program specified by the interrupt pointer (I).

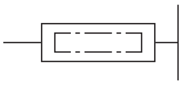
7.3 Program Execution Control Instructions

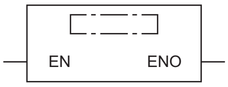
Disabling/enabling interrupt programs

DI, EI

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- DI: This instruction disables execution of interrupt programs.
- EI: This instruction clears the interrupt disabled state.

Ladder	ST
	ENO:=DI(EN); ENO:=EI(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
DI EI	Every scan

Processing details

■DI

- This instruction disables execution of interrupt programs until the EI instruction is executed even though they are requested.
- 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.
- For the operation of the DI (Disabling interrupt programs) instruction used with the DI (Disabling interrupt programs with specified priority or lower) instruction, refer to the following.

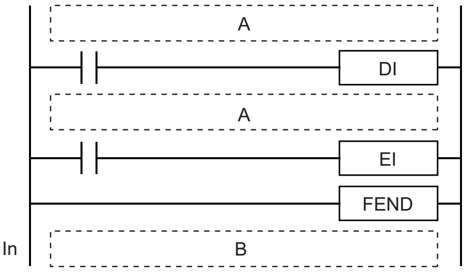
☞ Page 544 DI

- The DI (Disabling interrupt programs) instruction cannot be executed in interrupt programs. If executed, no processing is performed.

■EI

- This instruction clears the interrupt disabled state that has been set by the DI (Disabling interrupt programs) instruction, and enables execution of interrupt programs with the interrupt pointer numbers permitted by the IMASK instruction. When the IMASK instruction is not executed, I32 to I43 are disabled.
- For the operation of the EI instruction used with the DI (Disabling interrupt programs with specified priority or lower) instruction, refer to the following.

☞ Page 544 DI

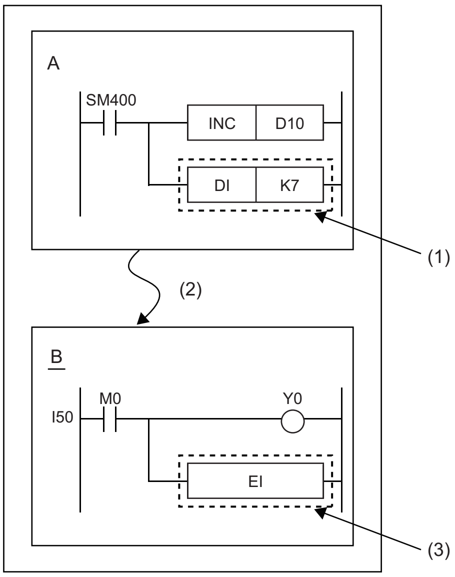
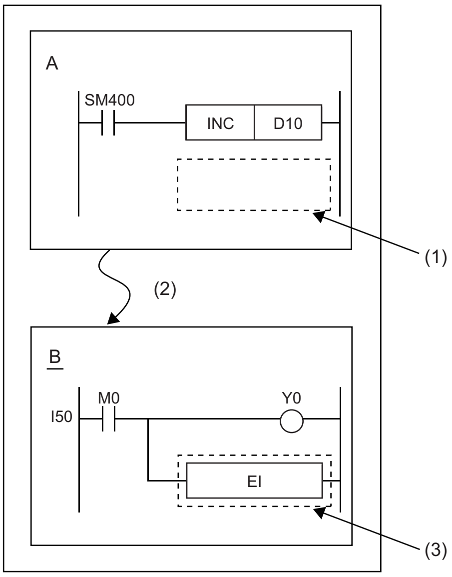


Even if an interrupt factor occurs between the DI and EI instructions, the corresponding interrupt program will not be executed until the processing between the DI and EI instructions completes.

A: Sequence program

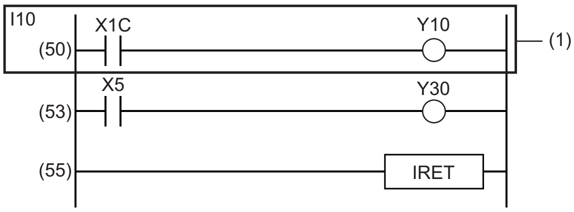
B: Interrupt program

- The operation of the EI instruction in interrupt programs differs depending on the execution status of the DI (Disabling interrupt programs with specified priority or lower) instruction before the EI instruction. The EI instruction in the interrupt program after the execution of the DI (Disabling interrupt programs with specified priority or lower) instruction can be executed.

When the EI instruction is executed in the interrupt program after the DI (Disabling interrupt programs with specified priority or lower) instruction was executed in the main routine program	When the EI instruction is executed in the interrupt program without executing the DI (Disabling interrupt programs with specified priority or lower) instruction in the main routine program
 A: Main routine program B: Interrupt program (1) The DI (Disabling interrupt programs with specified priority or lower) instruction is executed. (2) An interrupt has occurred. (3) Execution of the interrupt program is enabled.	 A: Main routine program B: Interrupt program (1) The DI (Disabling interrupt programs with specified priority or lower) instruction is not executed. (2) An interrupt has occurred. (3) Execution of the interrupt program is disabled. (No processing is performed.)

Point

- An interrupt pointer occupies one step. (At the program (1) below, for example, I10 is step 50, X1C is step 51, and Y10 is step 52.)



- If the EI and DI instructions are provided in the master control, these instructions are executed regardless of the execution status of the MC instruction.

Operation error

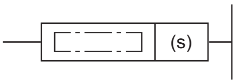
Error code (SD0)	Description
3362H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.

Disabling interrupt programs with specified priority or lower

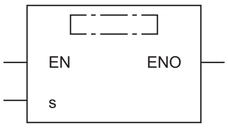
DI

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction disables execution of interrupt programs with the specified priority or lower.

Ladder	ST
	ENO:=DI_1(EN,s);

FBD/LD



(□ is replaced by DI_1.)

■Execution condition

Instruction	Execution condition
DI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Interrupt disable priority	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

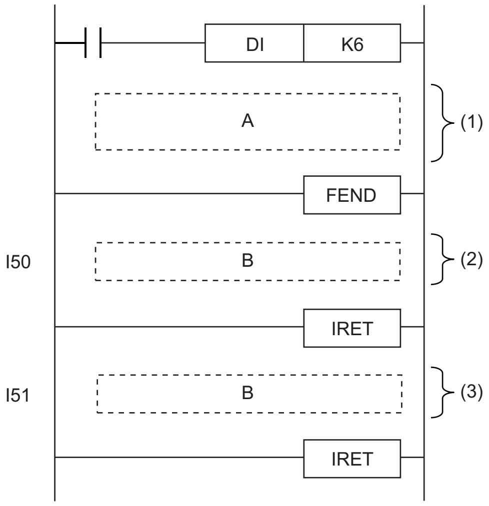
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction disables execution of interrupt programs with the interrupt pointer numbers specified by (s) or lower.

Interrupt priority setting

I No.	Priority
I50	5
I51	6



A: Sequence program

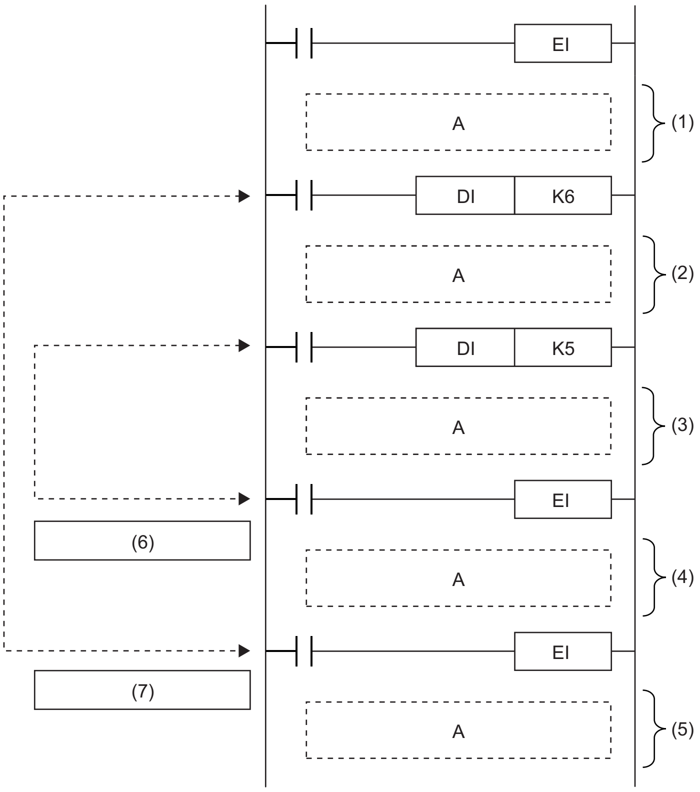
B: Interrupt program

(1) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)

(2) The interrupt program can be executed because its priority is 5.

(3) The interrupt program cannot be executed because its priority is 6.

- The execution of the EI instruction enables the interrupt that has been disabled by a single DI (Disabling interrupt programs with specified priority or lower) instruction. Note that if interrupts have been disabled only by the DI (Disabling interrupt programs) instruction, executing the EI instruction only once enables interrupts in all priorities.



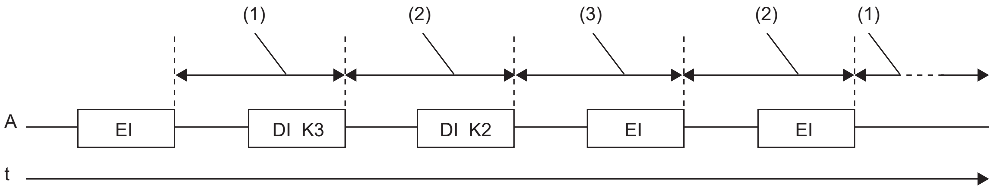
A: Sequence program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)
(3) Interrupt disabled section for priority 5 to 8 (Interrupt enabled section for priority 1 to 4)
(4) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)
(5) Interrupt enabled section for all priorities
(6) EI instruction for [DI K5]
(7) EI instruction for [DI K6]

- When multiple DI (Disabling interrupt programs with specified priority or lower) instructions are executed and the specified interrupt disable priority is lower than the priority of the currently disabled interrupt, the priority of the currently disabled interrupt remains unchanged.
- Up to 16 DI instructions can be nested.
- The priority of the interrupt pointer[*1]can be set in parameter. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

[*1] I0 to I15, I50 to I1023

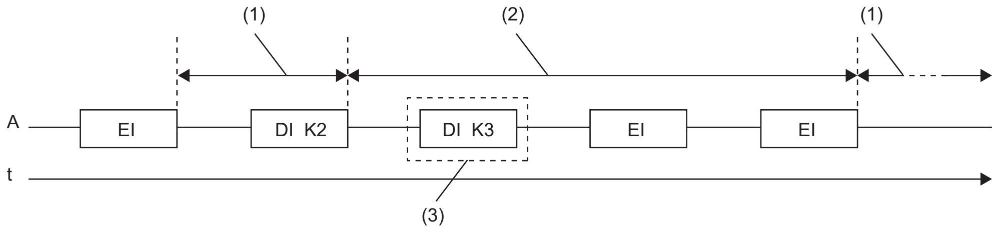
- The interrupt disabled priority currently set can be checked in SD758.
- If the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program and the interrupt disabled priority is changed, the value in SD758 also changes.

- When the DI (Disabling interrupt programs) instruction, DI (Disabling interrupt programs with specified priority or lower) instruction, and EI instruction are executed, the interrupt disabled sections will be as follows.
  - When another DI (Disabling interrupt programs with specified priority or lower) instruction with a wider priority range is executed during execution of the DI (Disabling interrupt programs with specified priority or lower) instruction



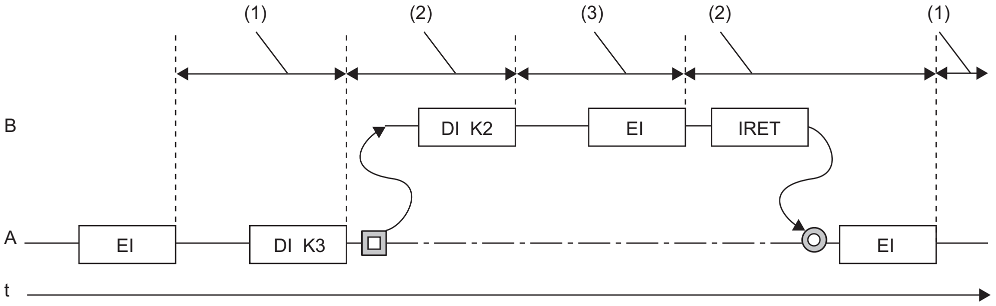
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 to 8 (Interrupt enabled section for priority 1 and 2)
(3) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)

  - When another DI (Disabling interrupt programs with specified priority or lower) instruction with a narrower priority range is executed during execution of the DI (Disabling interrupt programs with specified priority or lower) instruction



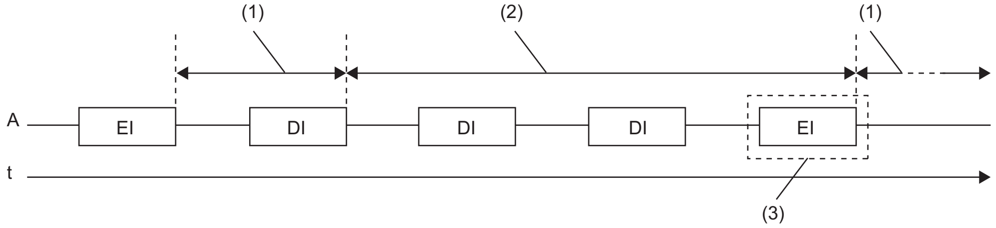
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)
(3) The priority of the disabled interrupt remains unchanged because the interrupt with priority 2 or lower is already disabled.

  - When the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program



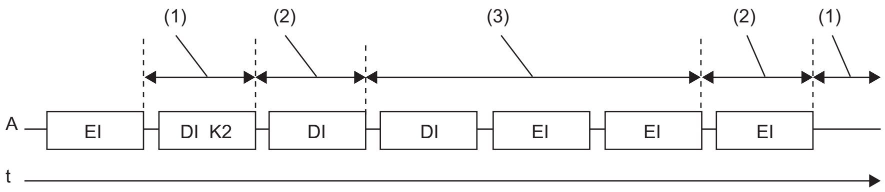
A: Scan execution type program
B: Interrupt program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 to 8 (Interrupt enabled section for priority 1 and 2)
(3) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)

  - When only the DI (Disabling interrupt programs) instruction is executed



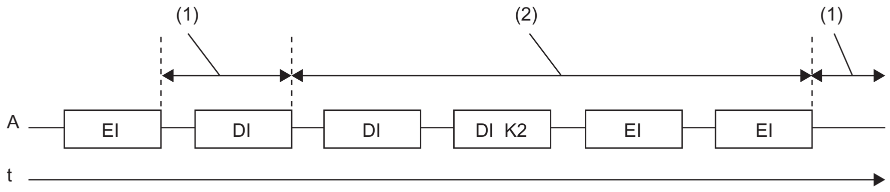
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 1 to 8 (Interrupt disabled section for all priorities)
(3) Executing the EI instruction only once enables interrupts with all priorities.

- When the DI (Disabling interrupt programs with specified priority or lower) instruction and the DI (Disabling interrupt programs) instruction are executed in this order



A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)
(3) Interrupt disabled section for priority 1 to 8 (Interrupt disabled section for all priorities)

- When the DI (Disabling interrupt programs) instruction and the DI (Disabling interrupt programs with specified priority or lower) instruction are executed in this order



A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 1 to 8 (Interrupt disabled section for all priorities)

Operation error

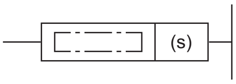
Error code (SD0)	Description
3362H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.
3405H	The priority specified by (s) is out of the range, 1 to 8.

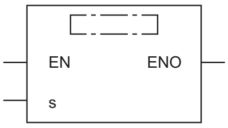
Interrupt program mask

IMASK

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number.

Ladder	ST
	ENO:=IMASK(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
IMASK	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Interrupt mask data or the start device where the interrupt mask data is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 16)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction enables or disables the execution of the interrupt program with the interrupt pointer number specified by the 16 points of bit pattern starting from the device specified by (s).
  - 1 (on): Enables the execution of the interrupt program.
  - 0 (off): Disables the execution of the interrupt program.
- The interrupt pointer numbers correspond to individual bits as shown below.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s)	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
(s)+1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
(s)+2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
(s)+3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
(s)+4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
(s)+5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
(s)+6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
(s)+7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
(s)+8	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
(s)+9	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
(s)+10	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
(s)+11	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
(s)+12	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
(s)+13	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
(s)+14	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
(s)+15	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

- When the system is powered on or the CPU module is reset, the execution of interrupt programs I0 to I31 and I44 to I1023 is enabled and the execution of interrupt programs I32 to I43 is disabled.
- The states of the device range (s) to (s)+15 are stored in SD1400 to SD1463.

Point

The IMASK instruction can enable or disable the execution of interrupt pointers I0 to I255 altogether.
To enable or disable the execution of interrupt pointers I256 to I1023, substitute the SIMASK instruction for the IMASK instruction in the program.
For details on the SIMASK instruction, refer to the following.
☞ Page 551 SIMASK

Operation error

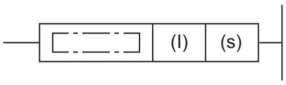
There is no operation error.

Disabling/enabling the specified interrupt pointer

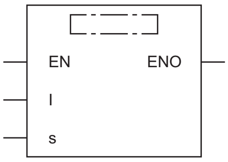
SIMASK

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number.

Ladder	ST
	ENO:=SIMASK(EN,I,s);

FBD/LD



■Execution condition

Instruction	Execution condition
SIMASK	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(I)	Interrupt pointer number to be interrupt-enabled or -disabled	I0 to I1023	Device name	POINTER[*1]
(s)	Execution status of the specified interrupt pointer	0: Disabled 1: Enabled	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the device (I) can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(I)
(I)	—	—	—	—	—	—	—	—	—	—	—	○
(s)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction, according to the data specified by (s), enables or disables the execution of the interrupt program with the interrupt pointer number specified by (I).
  - When (s) is 1: The execution of the interrupt program is enabled.
  - When (s) is 0: The execution of the interrupt program is disabled.
- When the system is powered on or the CPU module is reset, the execution of interrupt programs I0 to I31 and I44 to I1023 is enabled and the execution of interrupt programs I32 to I43 is disabled.
- The execution status of the interrupt pointers are stored in SD1400 to SD1463.

Point

The device (I) can be index modified. Using the SIMASK instruction specifying the index-modified device can enable or disable the execution of interrupt pointers I0 to I1023.

Operation error

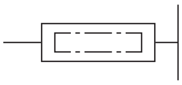
Error code (SD0)	Description
3405H	The interrupt pointer number specified by (I) is out of the valid range.
	The value in the device specified by (s) is neither 0 (disabled) nor 1 (enabled).

Returning from the interrupt program

IRET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

This instruction indicates the end of the processing of an interrupt program.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
IRET	Every scan

Processing details

- This instruction indicates the end of the processing of the interrupt program specified by the interrupt pointer (I).
- The instruction returns control to the sequence program after execution.

Operation error

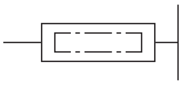
Error code (SD0)	Description
33A0H	There is no pointer corresponding to the interrupt number.
33A1H	After an interrupt occurs, the END, FEND, GOEND, or STOP instruction is executed before the IRET instruction.
33A2H	The IRET instruction is executed before the interrupt program specified by the interrupt pointer (I) is executed.
33A3H	The IRET instruction is executed in a fixed scan execution type program.
33A4H	The IRET instruction is executed in an event execution type program.

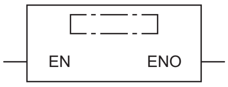
Resetting the watchdog timer

WDT(P)

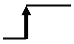
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions reset the watchdog timer.

Ladder	ST
	ENO:=WDT(EN); ENO:=WDTP(EN);

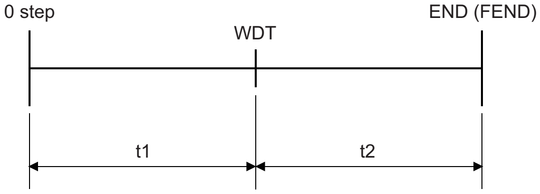
FBD/LD


■Execution condition

Instruction	Execution condition
WDT	
WDTP	

Processing details

- These instructions reset the watchdog timer in a program.
- These instructions are used when the scan time exceeds the value set for the watchdog timer depending on the condition. If the scan time exceeds the value set for the watchdog timer every scan, change the watchdog timer value in parameter using the engineering tool.
- The time values required for t1 from step 0 to the WDT(P) instruction and t2 from the WDT(P) instruction to the END (FEND) instruction must not exceed the value set for the watchdog timer.



- The instruction can be used twice or more in a single scan, but it takes time to turn off the output when an error occurs.
- Executing the instruction does not clear the scan time value stored in the special register. For this reason, the scan time value in the special register may be greater than the value set for the watchdog timer in parameter.

Operation error

There is no operation error.

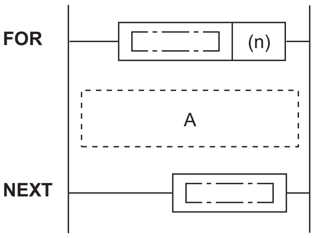
7.4 Structure Creation Instructions

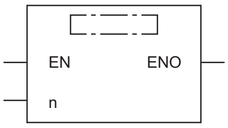
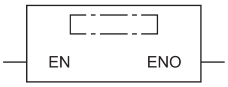
Performing the FOR to NEXT instruction loop

FOR, NEXT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions execute the processing between the FOR instruction and the NEXT instruction (n) times.

Ladder	ST
 A: Repeat program	Not supported

FBD/LD	
[FOR] 	[NEXT] 

■Execution condition

Instruction	Execution condition
FOR NEXT	Every scan

Setting data

■Description, range, data type

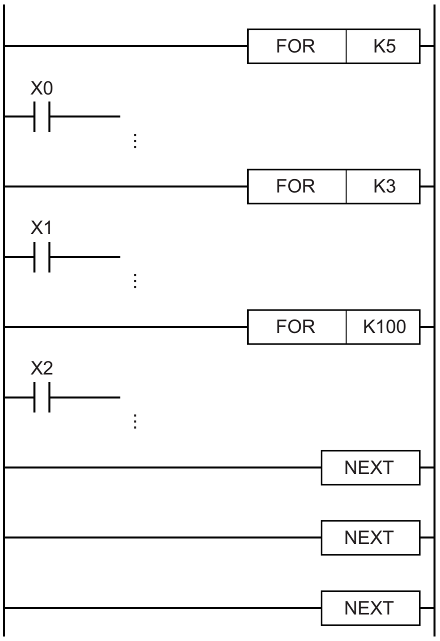
Operand	Description	Range	Data type	Data type (label)
(n)	Number of FOR to NEXT instruction loops	1 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- After the FOR to NEXT instruction loop is executed unconditionally (n) times, the step following the NEXT instruction is executed.
- Specify a value 1 to 32767 for (n). If a value -32768 to 0 is specified, the processing is performed regarding (n) as 1.
- To not to execute the FOR to NEXT instruction loop, use the CJ or SCJ instruction to jump the instruction loop.
- Up to 16 FOR instructions can be nested.



Operation error

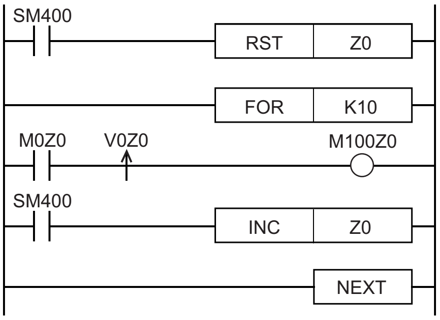
Error code (SD0)	Description
3340H	After execution of the FOR instruction, the END, FEND, or GOEND instruction is executed before the NEXT instruction.
	The STOP instruction is provided within the FOR to NEXT instruction loop.
3341H	The NEXT instruction is executed before the FOR instruction.
3361H	More than 16 FOR instructions are nested. (The 17th instruction is executed.)

Point

- To terminate the FOR to NEXT instruction loop being executed, use the BREAK instruction.

☞ Page 557 BREAK(P)

- To perform pulse operations of an index-modified program within the FOR to NEXT instruction loop, use the EGP or EGF instruction. Note, however, that no rising edge or falling edge instruction can be used on the operation output side.



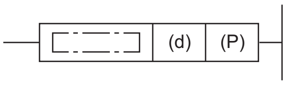
- The JMP instruction cannot be used to branch into the FOR to NEXT instruction loop from the outside.
- To create an easy-to-understand program, use a pair of instructions, the FOR and NEXT instructions, within a single program block.

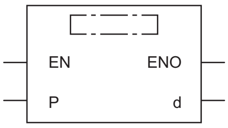
Forcibly terminating the FOR to NEXT instruction loop

BREAK(P)

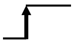
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the specified pointer.

Ladder	ST
	Not supported

FBD/LD


■Execution condition

Instruction	Execution condition
BREAK	
BREAKP	

Setting data

■Description, range, data type

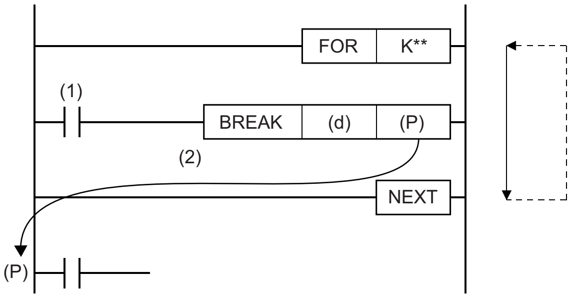
Operand	Description	Range	Data type	Data type (label)
(d)	Number of the device for storing the remaining number of loops	—	16-bit signed binary	ANY16
(P)	Branch destination pointer number at the time of forced termination of loop processing	—	Device name	POINTER
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(P)
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

- These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the pointer specified by (P). Only a pointer in the same program file can be specified for (P). An operation error occurs if a pointer in another program file is specified.



If no BREAK instruction is executed, the program performs loop processing as many times as specified by the FOR instruction.

(1) Forced termination condition

(2) When the condition is met

- The remaining number of FOR to NEXT instruction loops at the time of forced termination is stored in (d). The remaining number of loops includes the processing when the BREAK(P) instruction is executed.
- The BREAK(P) instruction can be used only within the FOR to NEXT instruction loop.
- The BREAK(P) instruction is valid only for one level of nesting. To forcibly terminate multiple levels of nesting, execute as many BREAK(P) instructions as nesting levels.

Operation error

Error code (SD0)	Description
3342H	The instruction is used outside the FOR to NEXT instruction loop.
3380H	The jump destination corresponding to the pointer specified by (P) does not exist.
	A pointer in another program is specified in (P).

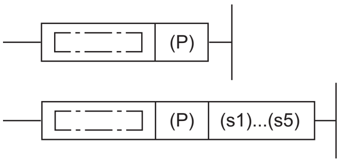
Calling a subroutine program

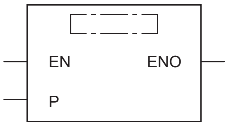
CALL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

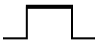
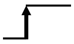
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions execute the subroutine program of the specified pointer.

Ladder	ST
	Not supported

FBD/LD


■Execution condition

Instruction	Execution condition
CALL	
CALLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1) to (s5)[1]	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Cannot be specified in FBD/LD.

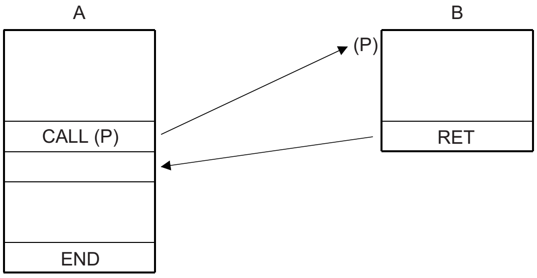
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1) to (s5)	○[1]	○	○	○	○	○	○	○	○	—	—	—

[1] Devices other than F can be used.

Processing details

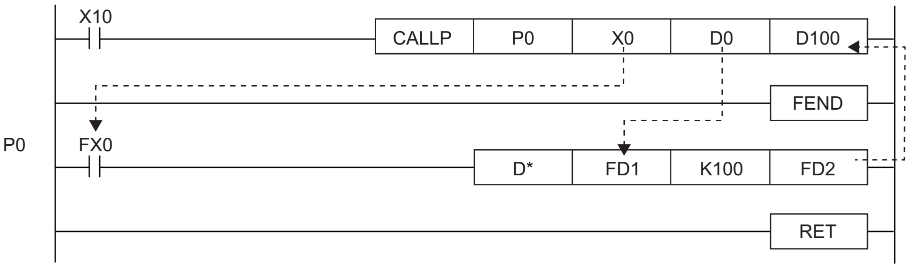
- These instructions execute the subroutine program of the pointer specified by (P). The instructions can execute the subroutine program specified by a pointer in the same program file or the subroutine program specified by a global pointer.



A: Main routine program

B: Subroutine program

- When the function device (FX, FY, or FD) is used in the subroutine program, specify the device corresponding to the function device in (s1) to (s5). The following figure shows the data of the device areas specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the data in FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument. The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

- The following table lists the data sizes of each function device.

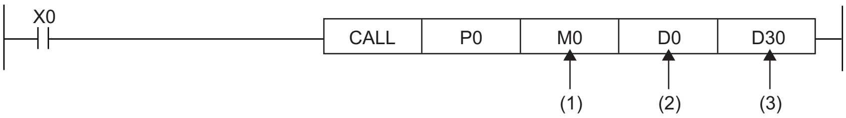
Function device	Device to be used	Data size
FX, FY	Bit device	1 point
	Bit-specified word device	1 bit
FD	Digit-specified bit device[*2]	4 words[*3]
	Word device	4 words[*3]

[*2] An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16.

[*3] The data size varies depending on the instruction used.

Ex.

Data in the specified devices



(1) M0 occupied (The data is transferred to FX0.)

(2) D0 to D3 occupied (The data is transferred to FD1.)

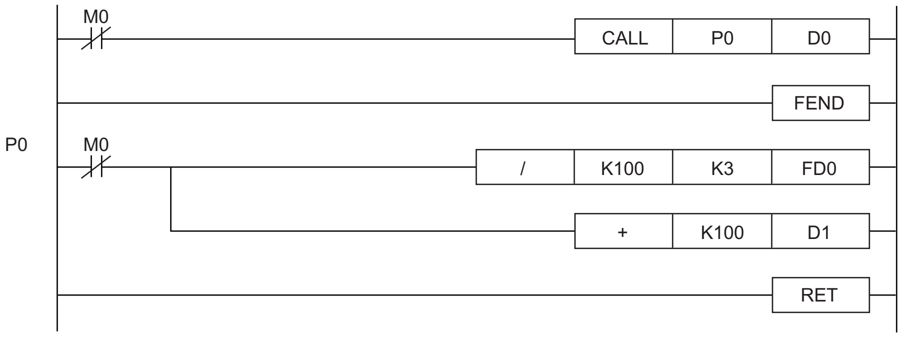
(3) D30 to D33 occupied (The data is transferred to FD2.)

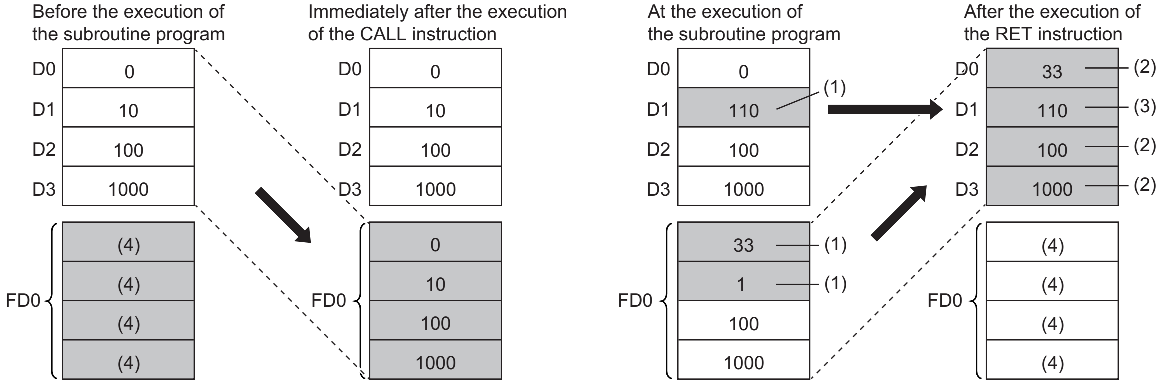
- The CALL(P) instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the CALL(P) instruction. Also, the types of function devices and CALL(P) instruction arguments must be the same.
- Set the device numbers in the argument of the CALL(P) instruction so that they do not overlap. If they overlap, normal operation cannot be performed.
- If the timer or counter is specified as a device in the argument of the CALL(P) instruction, only the current value is transmitted/received.

- Do not use any device used in the argument of the CALL(P) instruction in the subroutine program. If used, normal operation cannot be performed.

Ex.

Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

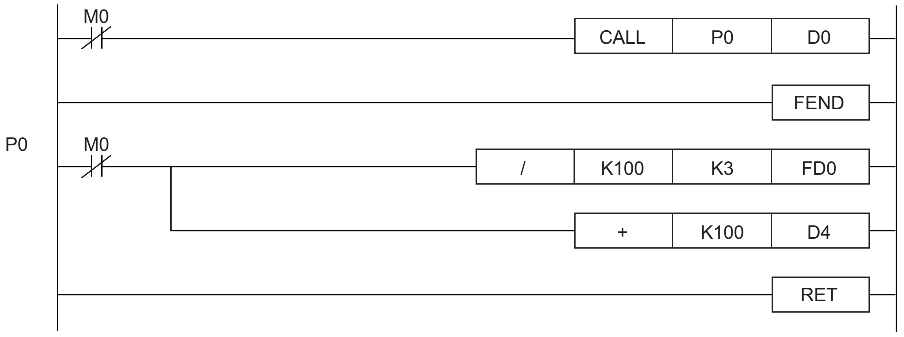


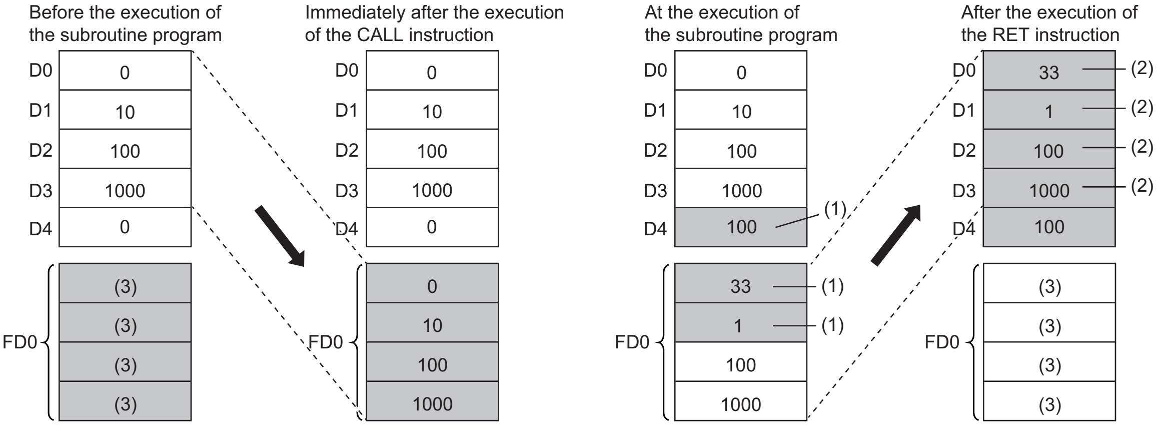


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

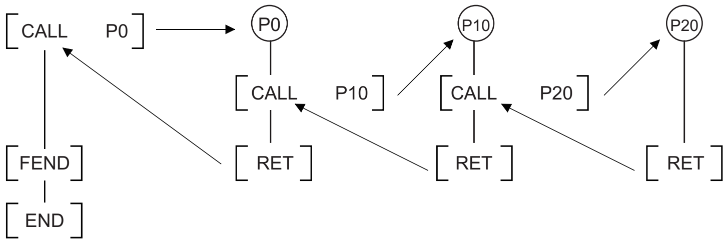
Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) Undefined values are stored.

- Up to 16 CALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FACLL(P), ECALL(P), EFCALL(P), and XCALL instructions.



- Devices which are turned on in the subroutine program hold the on state even when the subroutine program is not executed. The on state can be changed to off by executing the FCALL(P) instruction.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

Operation error

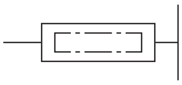
Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3360H	More than 16 CALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the CALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the CALL(P) instruction.

Returning from the subroutine program called

RET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction indicates the end of a subroutine program.

Ladder	ST
	Not supported

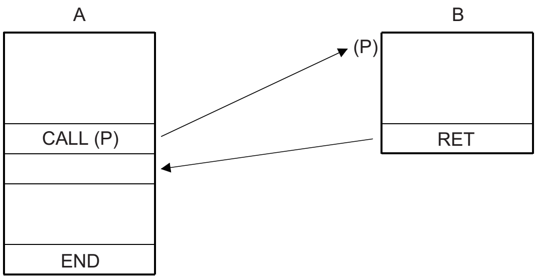
FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
RET	Every scan

Processing details

- This instruction indicates the end of a subroutine program.
- When the instruction is executed, the program returns to the next step where the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction that called the subroutine program is executed.



A: Main routine program

B: Subroutine program

Operation error

Error code (SD0)	Description
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction.

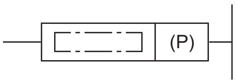
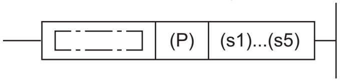
Calling a subroutine program and turning the output off

FCALL(P)

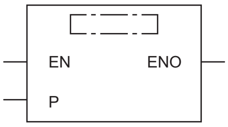
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

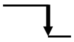
These instructions perform non-execution processing of the subroutine program of the specified pointer.

Ladder	ST
 	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
FCALL	
FCALLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)[*1]	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] Cannot be specified in FBD/LD.

- In safety programs executed by the Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Devices other than F can be used.

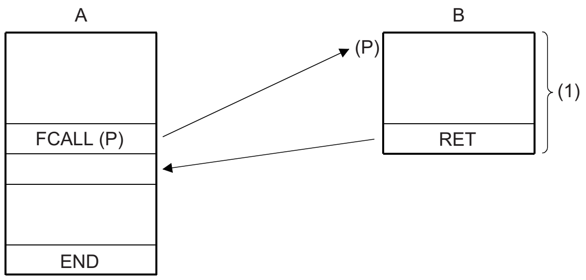
- In safety programs executed by the Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

Processing details

- When the FCALL (P) instruction is executed, this instruction executes non-execution processing*2 of the subroutine program of the pointer (P). The FCALL(P) instruction can disable the execution of the subroutine program specified by a pointer in the same program file or the subroutine program specified by a global pointer.

*2 Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



A: Main routine program
B: Subroutine program
(1) Non-execution processing is performed when the command of the FCALL(P) instruction changes from on to off.

- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

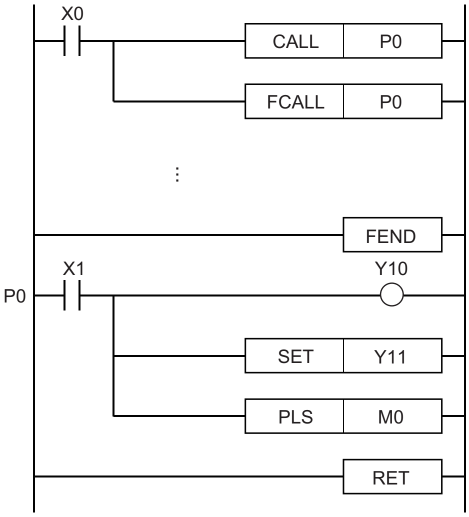
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

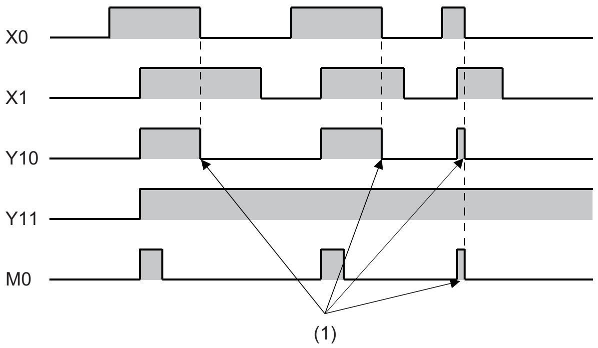
- The FCALL(P) instruction is used in combination with the CALL(P) instruction. If the FCALL(P) instruction is not used in combination with the CALL(P) instruction, non-execution processing of the subroutine program is not performed even if the execution command is turned off, and therefore the output status of each coil instruction is retained.

- When the execution command is turned off, non-execution processing of the subroutine program is performed, enabling the OUT instruction and PLS instruction (including pulse conversion instructions) to be forcibly turned off.

Ex.

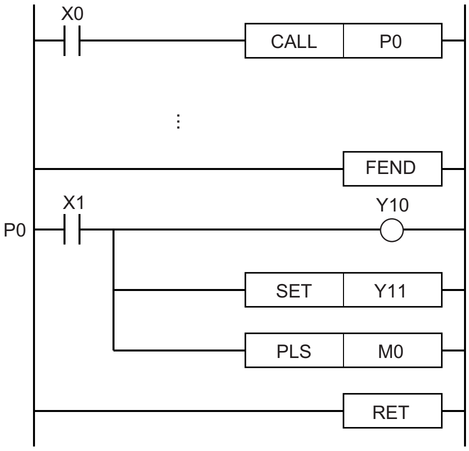
When the FCALL(P) instruction is used

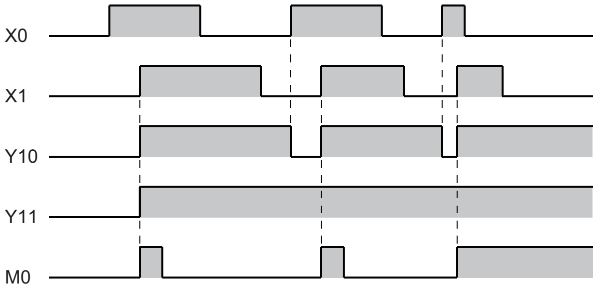




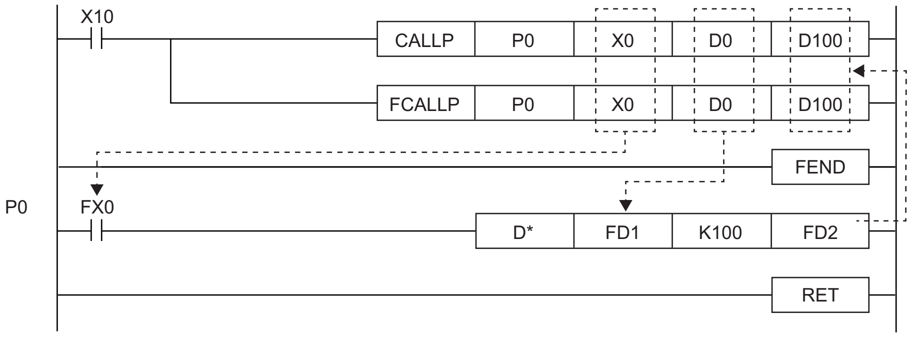
(1) Forced off by the FCALL instruction

When the FCALL(P) instruction is not used





- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument. The device specified as a function device should be secured for the data size. An error occurs if it cannot be secured for the data size.

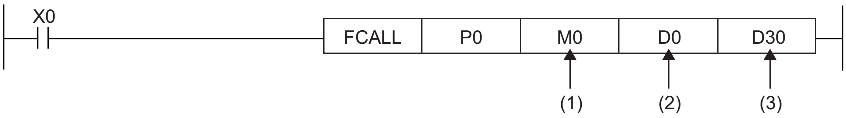
- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified[3]	4 words[4]
	Word device	4 words

[3] An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
[4] The upper two words of FD are 0.

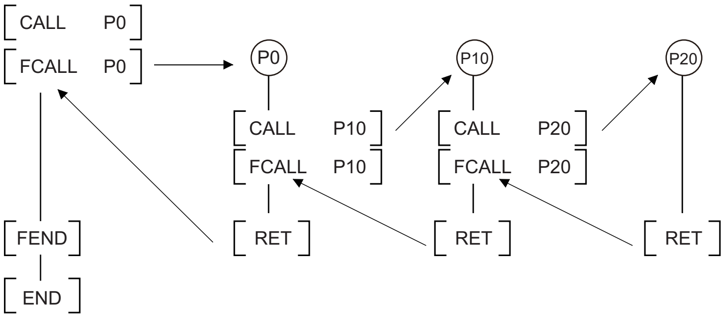
Ex.

Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- The FCALL(P) instruction can use (s1) to (s5).
- Up to 16 FCALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3360H	More than 16 FCALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by the FCALL(P) instruction does not exist.
3381H	After execution of the FCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the FCALL(P) instruction.

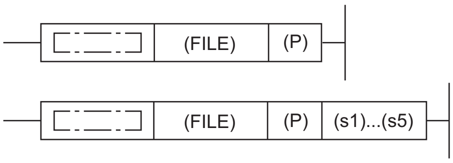
Calling a subroutine program in the specified program file

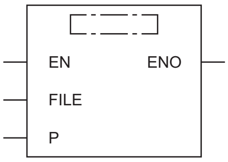
ECALL(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~RnSFCPU (Safety)~~

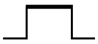
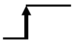
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions execute the subroutine program corresponding the specified pointer of the specified program file name.

Ladder	ST
 FILE: File name	Not supported

FBD/LD
 FILE: File name

■Execution condition

Instruction	Execution condition
ECALL	
ECALLP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(File name)	Program file name to be called	—	Unicode string	ANYSTRING_DOUBLE
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

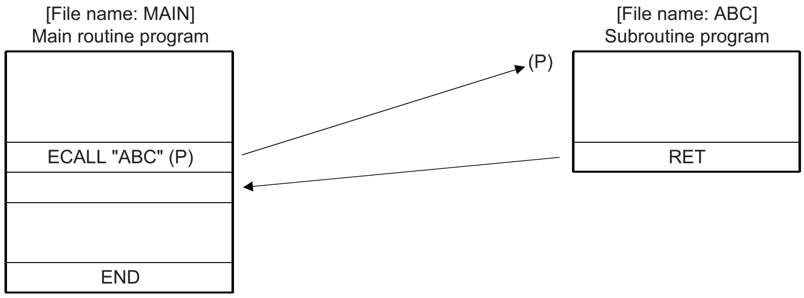
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	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	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

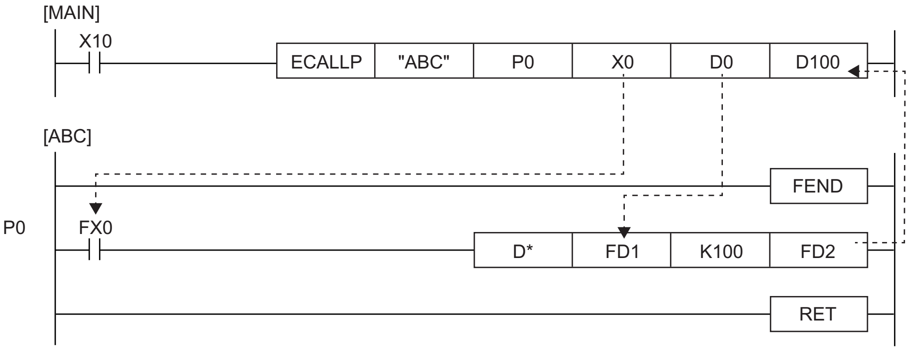
*1 Any value other than F can be used.

Processing details

- When the ECALL(P) instruction is executed, these instructions execute the subroutine program corresponding to the pointer specified by (P) of the specified program file name. The ECALL(P) instructions can also call the subroutine program using the local pointer of another program file.



- Only a program file stored in program memory can be specified for the file name.
- Extension ".PRG" need not be specified for the file name. (Only .PRG files can be processed by these instructions.)
- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument. The device specified as a function device should be secured for the data size. An error occurs if it cannot be secured for the data size.

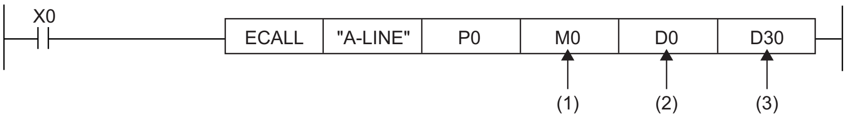
- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified[*2]	4 words[*3]
	Word device	4 words[*3]

[*2] An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
[*3] The data size varies depending on the instruction used.

Ex.

Content of specified device



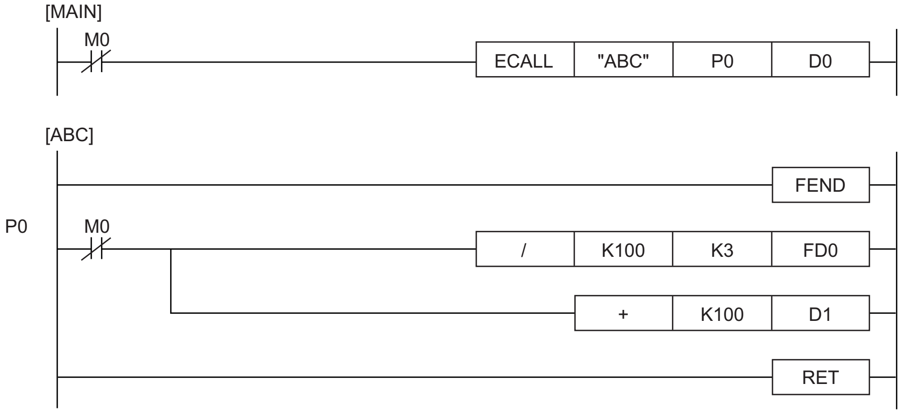
(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

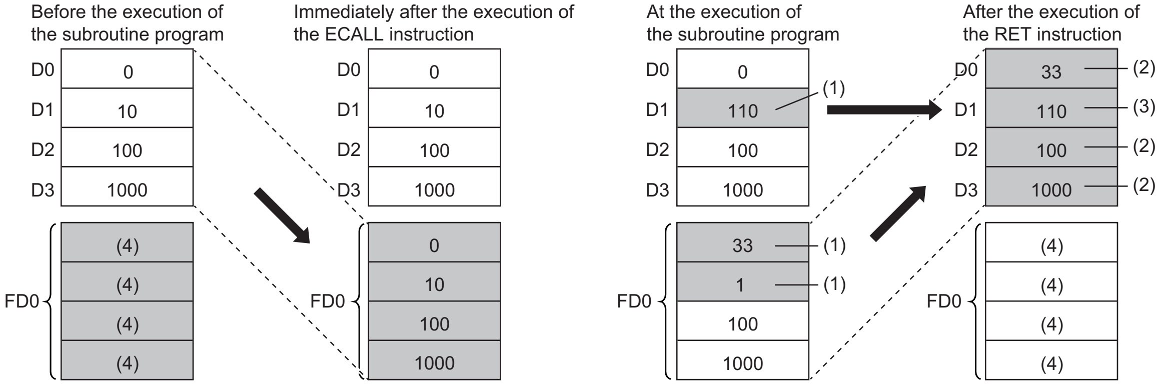
- The ECALL(P) instruction can use (s1) to (s5).

- Any device used in the argument of the ECALL(P) instruction must not used in the subroutine program. Otherwise, normal operation cannot be performed.

Ex.

Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.

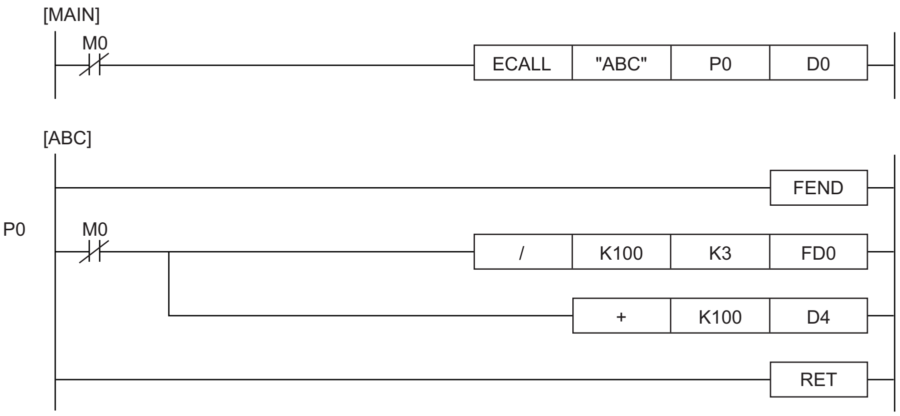
(2) These values are replaced with the function device values.

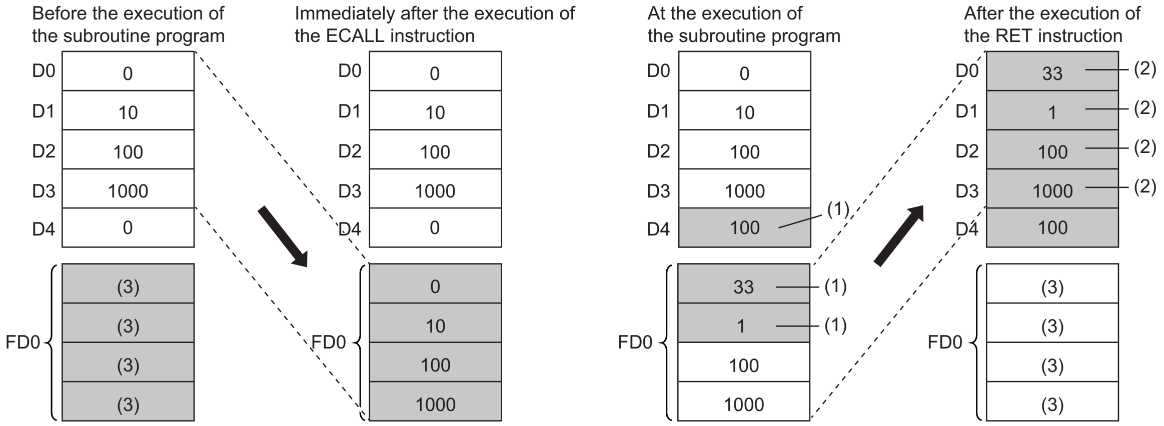
(3) The value of D1 is not replaced with the function device value.

(4) Undefined values are stored.

Ex.

Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.



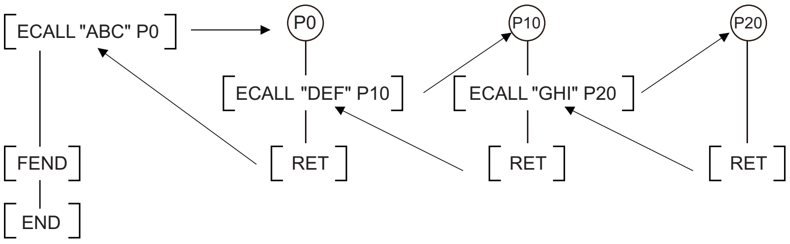


(1) The operation result of the subroutine program is stored.

(2) These values are replaced with the function device values.

(3) Undefined values are stored.

- The device numbers specified by the ECALL(P) instruction arguments must not be overlapping. If they are overlapping, normal operation cannot be performed.
- Up to 16 ECALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



- Devices which are turned on in the subroutine program are retained even while the subroutine program is not executed. Devices which are turned on during execution of the subroutine program can be turned off by the EFCALL(P) instruction.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
2840H	The file specified by (file name) does not exist.
2841H	The file specified by (file name) cannot be executed.
2842H	The type of the program with the file name specified by (file name) is not supported.
3360H	More than 16 ECALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the ECALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the ECALL(P) instruction.

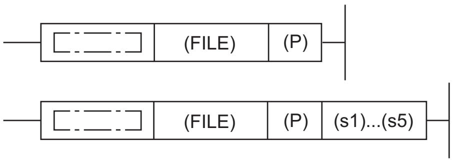
Calling a subroutine program in the specified program file and turning the output off

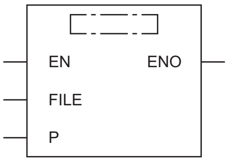
EFCALL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

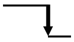
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions perform non-execution processing of the subroutine program corresponding the specified pointer of the specified program file name.

Ladder	ST
 FILE: File name	Not supported

FBD/LD
 FILE: File name

■Execution condition

Instruction	Execution condition
EFCALL	
EFCALLP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(File name)	Program file name subject to non-execution processing	—	Unicode string	ANYSTRING_DOUBLE
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1) to (s5)[*1]	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

■Applicable devices

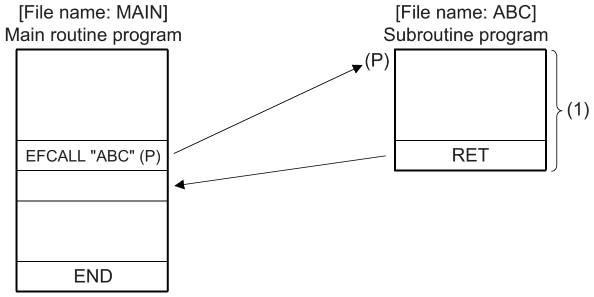
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○[1]	○	○	○	○	○	○	○	○	—	—	—

[1] Any value other than F can be used.

Processing details

- When the EFCALL(P) instruction is executed, these instructions perform non-execution processing[2] of the subroutine program of the pointer specified by (P). The EFCALL(P) instructions can also call the subroutine program using the local pointer of another program file.

[2] Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



(1) Non-execution processing is performed when the command of the EFCALL(P) instruction changes from on to off.

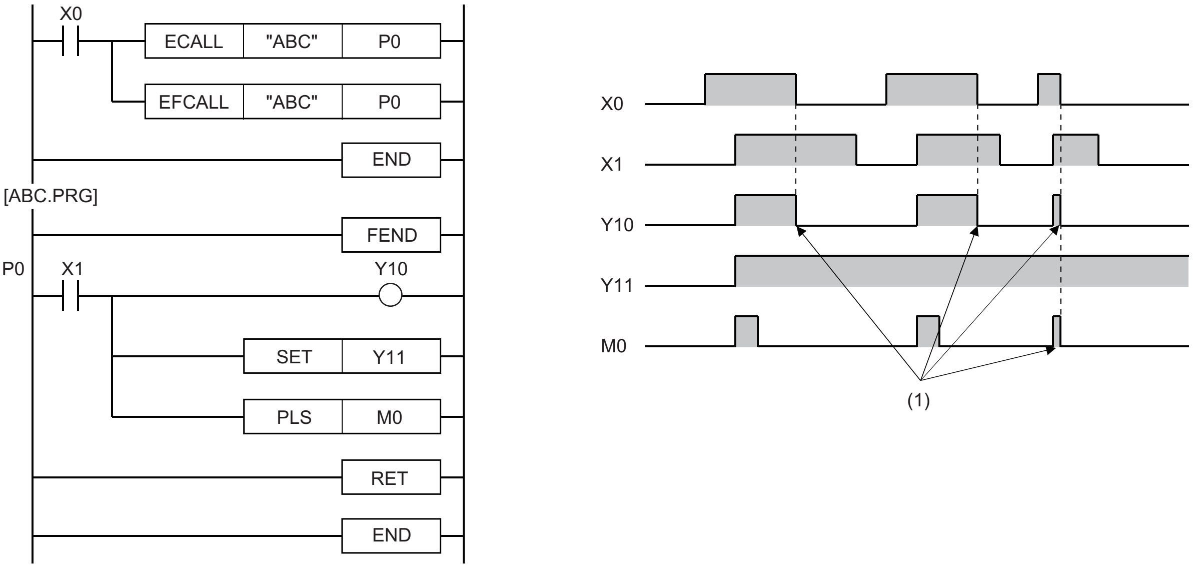
- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

- The EFCALL(P) instruction is used in combination with the ECALL(P) instruction. If the EFCALL(P) instruction is not used in combination with the ECALL(P) instruction, non-execution processing of the subroutine program is not performed even if the execution command is turned off, and therefore the output status of each coil instruction is retained.
- When the execution command is turned off, non-execution processing of the subroutine program is performed, enabling the OUT instruction and PLS instruction (including pulse conversion instructions) to be forcibly turned off.

Ex.

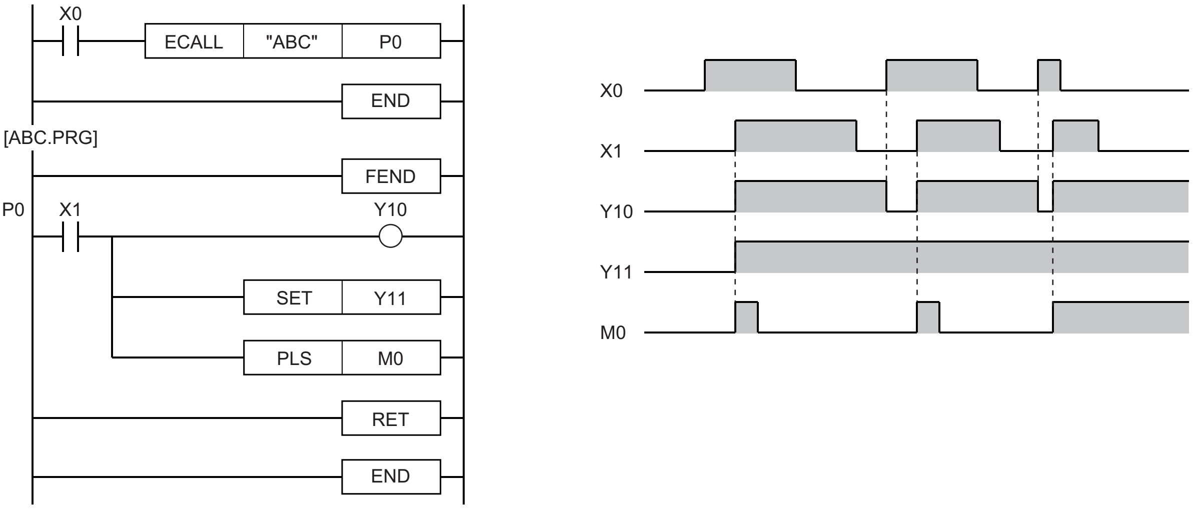
When the EFCALL (P) instruction is used



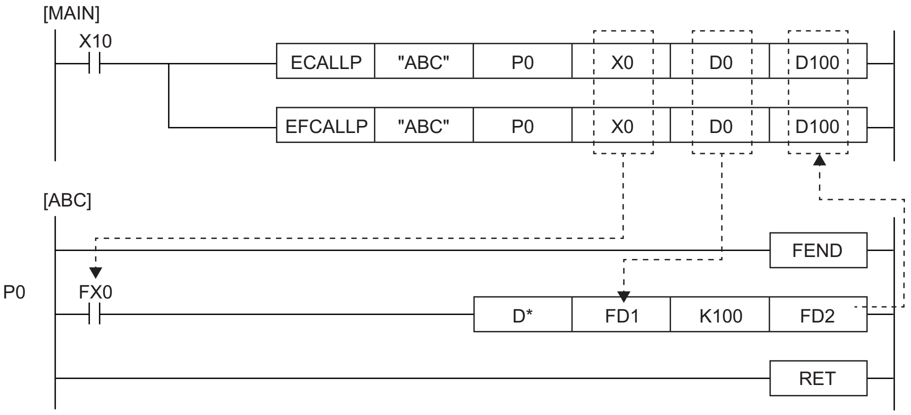
(1) Forced off by the EFCALL instruction

Ex.

When the EFCALL (P) instruction is not used



- Only a program file stored in program memory can be specified for the file name.
- Extension ".PRG" need not be specified for the file name. (Only .PRG files can be processed by these instructions.)
- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument. The device specified as a function device should be secured for the data size. An error occurs if it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

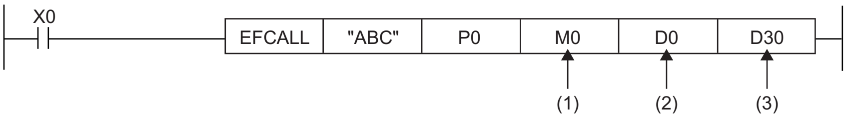
Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified[3]	4 words[4]
	Word device	4 words

[3] An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

[4] The upper two words of FD are 0.

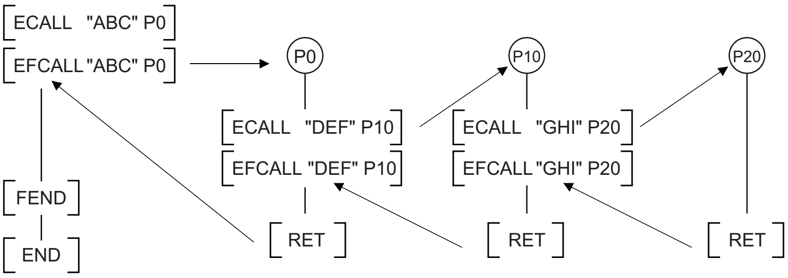
Ex.

Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- The EFCALL(P) instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the EFCALL(P) instruction. Also, the types of function devices and EFCALL(P) instruction arguments must be the same.
- Up to 16 EFCALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

Operation error

Error code (SD0)	**Description**
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
2840H	The file specified by (file name) does not exist.
2841H	The file specified by (file name) cannot be executed.
2842H	The type of the program with the file name specified by (file name) is not supported.
3360H	More than 16 EFCALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the EFCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the EFCALL(P) instruction.

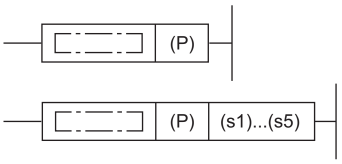
Calling a subroutine program

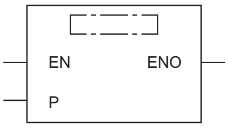
XCALL

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

- [RnPCPU (redundant)] If the instruction is used in a program executed in both systems, there are restrictions on its operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

This instruction performs execution or non-execution processing of a subroutine program. When the condition is satisfied, the instruction triggers CALL for the subroutine. When the condition is broken, it triggers FCALL.

Ladder	ST
	Not supported

FBD/LD


■Execution condition

Instruction	Execution condition
XCALL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)[1]	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Cannot be specified in FBD/LD.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	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	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○[1]	○	○	○	○	○	○	○	○	—	—	—

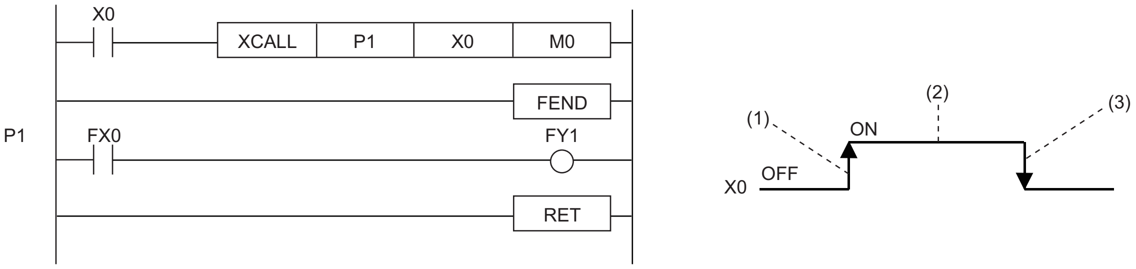
[1] Devices other than F can be used.

Processing details

- This instruction performs execution or non-execution processing of a subroutine program.
  - For execution of the subroutine program, it operates each coil instruction according to the on/off status of condition contacts.
  - For non-execution processing of the subroutine program, it operates each coil instruction in the same way as when condition contacts are off.
- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

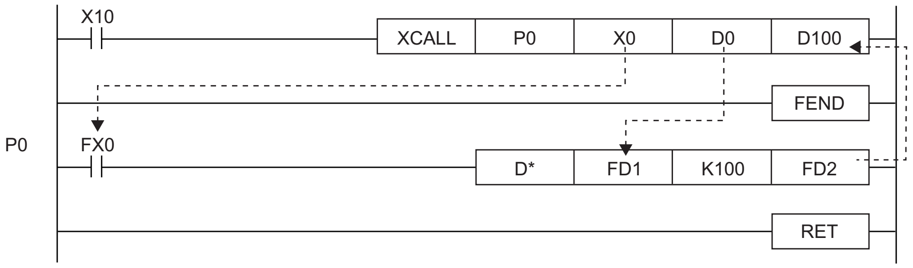
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

- The following figure shows the operation of the XCALL instruction.



(1) On the rising edge of X0 (off→on): Executes the subroutine program specified by P1.
(2) During X0 is on: Executes the subroutine program specified by P1. ("During X0 is on" does not include the rising edge of X0.)
(3) On the falling edge of X0 (on→off): Performs non-execution processing of a subroutine program specified by P1.

- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument. The device specified as a function device should be secured for the data size. An error occurs if it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

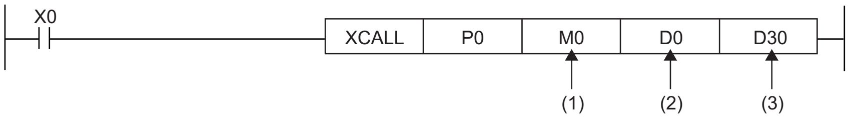
Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit is specified for a word device	1 bit
FD	When the bit device digit is specified[*2]	4 words[*3]
	Word device	4 words[*3]

[*2] An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

[*3] The data size varies depending on the instruction used.

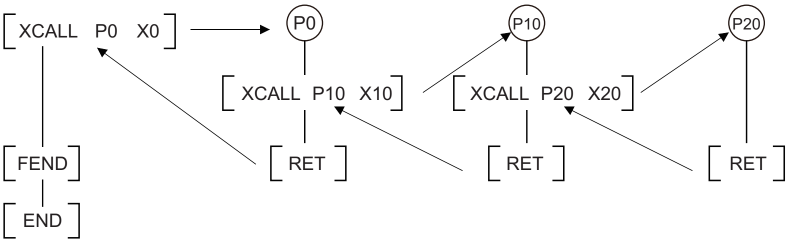
Ex.

Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

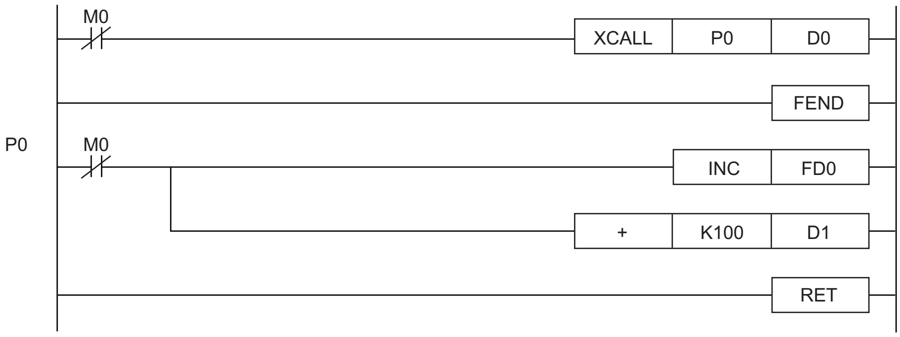
- The XCALL instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the XCALL instruction. Also, the types of function devices and XCALL instruction arguments must be the same.
- The device numbers specified by the XCALL instruction arguments must not be overlapping. If they are overlapping, normal operation cannot be performed.
- Up to 16 XCALL instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.

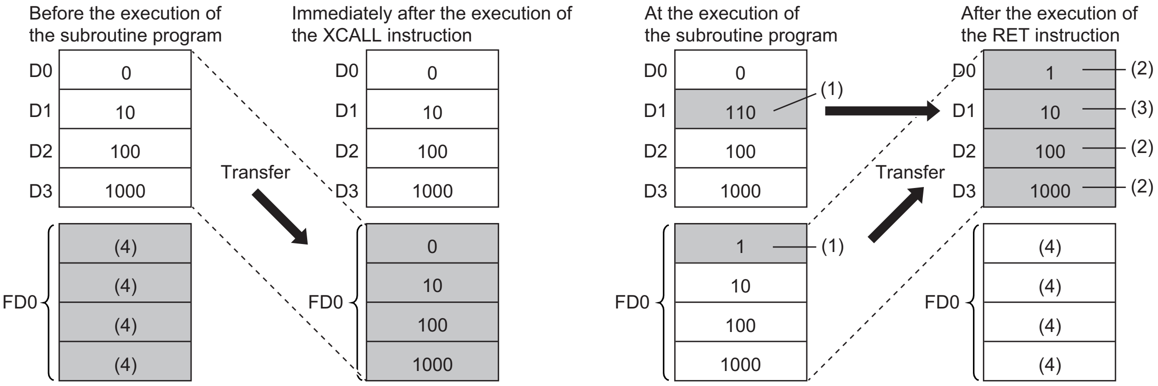


- Any device used in the argument of the XCALL instruction must not used in the subroutine program. Otherwise, normal operation cannot be performed.

Ex.

Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

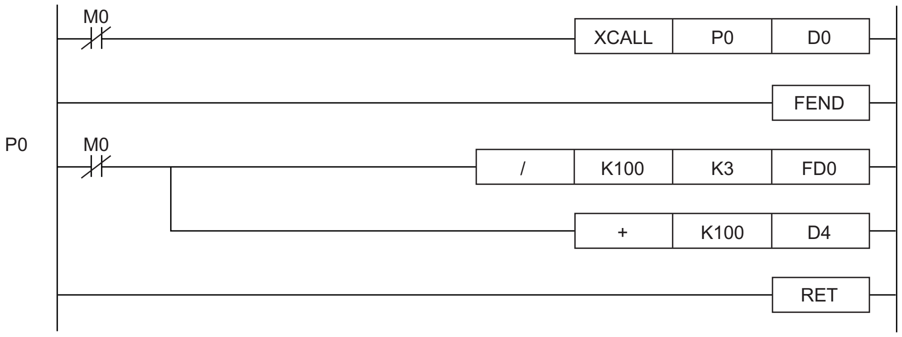


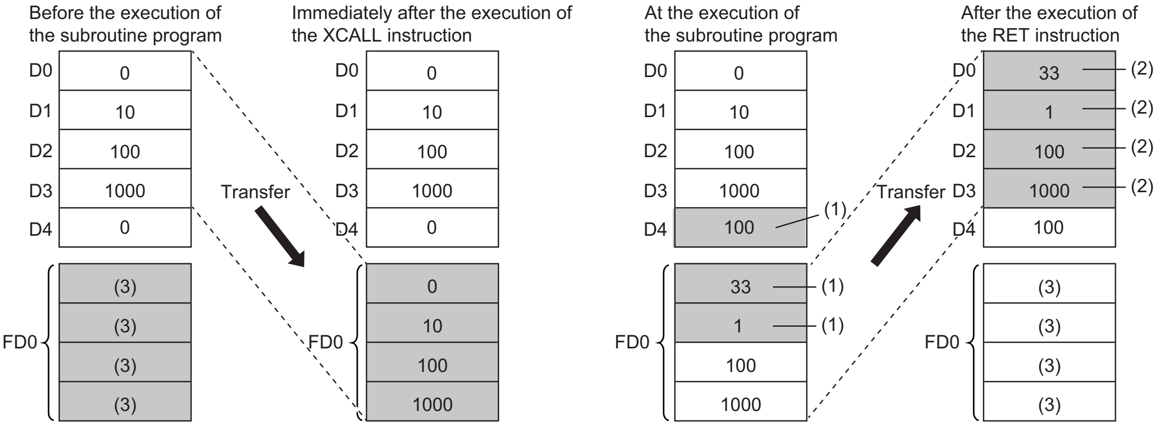


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) Undefined values are stored.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3360H	More than 16 XCALL instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the XCALL instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the XCALL instruction.

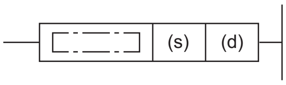
7.5 Data Table Operation Instructions

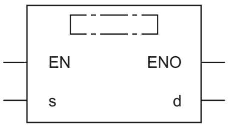
Reading the oldest data from the data table

FIFR(P)

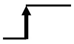
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions store the data first stored in the table in the specified device.

Ladder	ST
	ENO:=FIFR(EN,s,d); ENO:=FIFRP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
FIFR	
FIFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device for storing the data read from the table	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

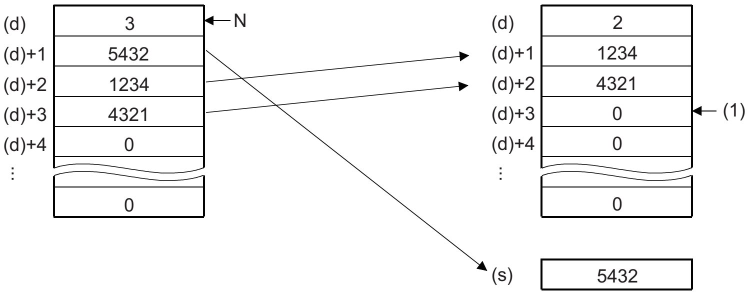
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the first-in data in the device specified by (d)+1 in the table specified by (d) in the device specified by (s). After execution of the FIFR(P) instruction, the data in the data table is moved forward one by one.



N: Number of data
(1) Filled with 0.

- An interlock mechanism should be used to prevent the FIFR(P) instruction from being executed when the value stored in the device specified by (d) is 0.
- A number from 0 to 65535 is stored in the number of data (d).

Operation error

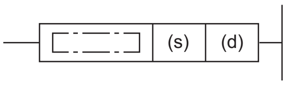
Error code (SD0)	Description
3405H	The FIFR(P) instruction is executed when the value in the device specified by (d) is 0.

Reading the newest data from the data table

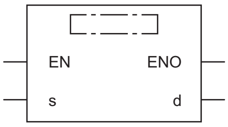
FPOP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

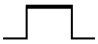
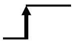
These instructions store the data last stored in the table in the specified device.

Ladder	ST
	ENO:=FPOP(EN,s,d); ENO:=FPOPP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FPOP	
FPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device for storing the data read from the table	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

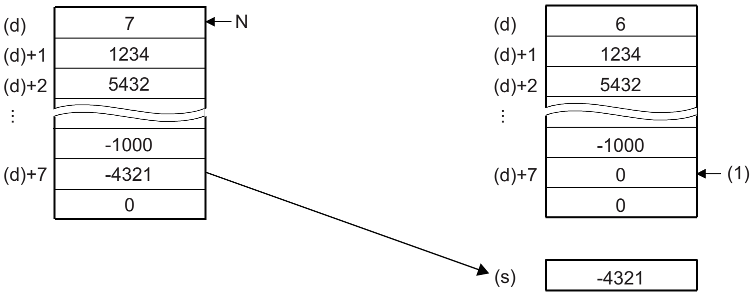
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the data last stored in the table in the device specified by (d) in the device specified by (s). After execution of the FPOP(P) instruction, the device in which the data read by the instruction has been stored is cleared to 0.



N: Number of data
(1) Filled with 0.

- An interlock mechanism should be used to prevent the FPOP(P) instruction from being executed when the value stored in the device specified by (d) is 0.
- A number from 0 to 65535 is stored in the number of data (d).

Operation error

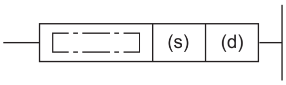
Error code (SD0)	Description
3405H	The FPOP(P) instruction is executed when the value in the device specified by (d) is 0.

Writing data to the data table

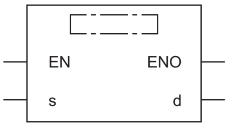
FIFW(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

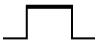
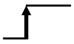
These instructions transfer 16-bit binary data to the specified data table.

Ladder	ST
	ENO:=FIFW(EN,s,d); ENO:=FIFWP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FIFW	
FIFWP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be written to the table, or the device number where the write data is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

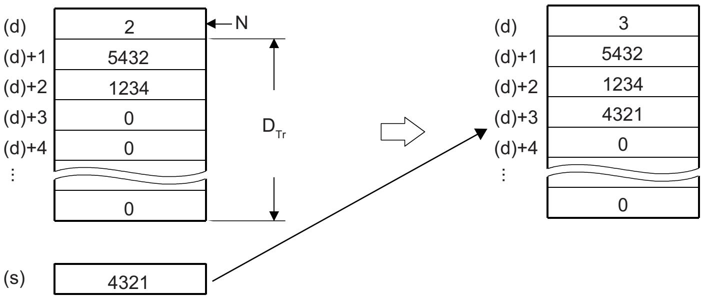
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the 16-bit binary data in the device specified by (s) in the data table in the device specified by (d). The number of data stored in the table is stored in (d), and the data in the device specified by (s) is stored in order in the device specified by (d)+1 and later.



N: Number of data

D_{Tr}: Data table range (managed by users)

- When the FIFW(P) is executed for the first time, the value in the device specified by (d) must be cleared.
- A care must be taken for the data table range because data is stored sequentially in the device specified by (d)+1 and later.
- A number from 0 to 65535 is stored in the number of data (d).

Operation error

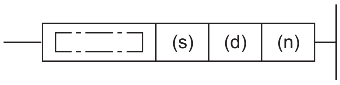
Error code (SD0)	Description
3405H	The FIFW(P) instruction is executed when the value in the device specified by (d) is FFFFH.

Inserting data to the data table

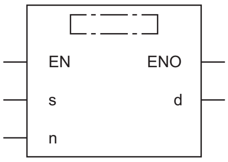
FINS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

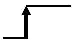
These instructions insert 16-bit binary data to the (n)th position in the specified data table.

Ladder	ST
	ENO:=FINS(EN,s,n,d); ENO:=FINSP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FINS	
FINSP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the insertion data is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16[*1]
(n)	Insertion position in the table	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

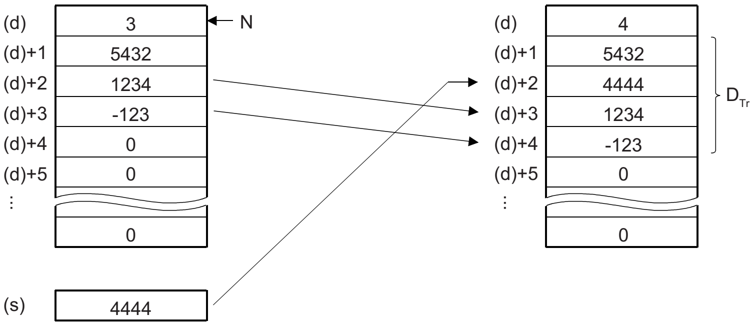
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the 16-bit binary data in the device specified by (s) in the (n)th position in the data table in the device specified by (d). After execution of the FINS(P) instruction, the data after the (n)th position in the data table is moved down one by one.



N: Number of data

D_{Tr}: Data table range (managed by users)

When (n)=2, data is inserted to the device specified by (d)+2.

- A number from 0 to 65535 is stored in the number of data (d).

Operation error

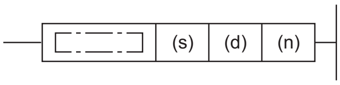
Error code (SD0)	Description
3405H	The FINS(P) instruction is executed when the value specified by (n) is 0.
	The FINS(P) instruction is executed when the value in the device specified by (d) is FFFFH.

Deleting data from data table

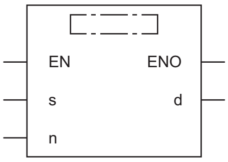
FDEL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

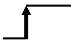
These instructions delete the data at the (n)th position in the data table.

Ladder	ST
	ENO:=FDEL(EN,s,n,d); ENO:=FDELP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FDEL	
FDELP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data to be deleted is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16[*1]
(n)	Deletion position in the table	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

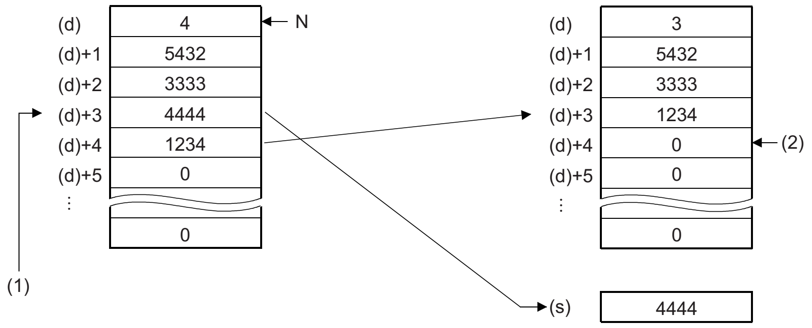
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions delete the (n)th data in the data table in the device specified by (d) and store it in the device specified by (s). After execution of the FDEL(P) instruction, the data at the (n)+1th position and later in the data table is moved forward one by one.



N: Number of data

(1) When (n)=3, data in (d)+3 is deleted.

(2) Filled with 0.

- A number from 0 to 65535 is stored in the number of data (d).

Operation error

Error code (SD0)	Description
3405H	The FDEL(P) instruction is executed when the value specified by (n) is 0.
	The FDEL(P) instruction is executed when the value in the device specified by (d) is 0.

7.6 Reading/Writing Data Instructions

Reading/Writing data to data memory

The data write instruction is an instruction to write arbitrary device data to data memory.

Writing the fixed values used for operation and operation results to data memory can prevent data loss when the battery is low.

The data that has been written to data memory can be read at any time using the data memory read instruction.

■Execution method

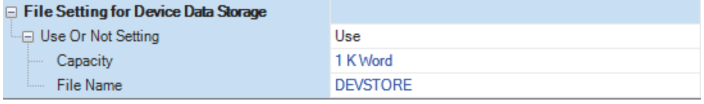
Use the SP.DEVST instruction to write device data to data memory.

Use the S(P).DEVLD instruction to read device data from data memory to any specified device.

■Setting method

When the SP.DEVST and S(P).DEVLD instructions are used, a device data storage file must be set up in advance.

[CPU Parameter]⇨[File Setting]⇨[File Setting for Device Data Storage]



File Setting for Device Data Storage	
Use Or Not Setting	Use
Capacity	1 K Word
File Name	DEVSTORE

Item	Description
Capacity	1K to 512K (words)
File name	DEVSTORE (fixed)

If data memory does not have enough free space for creating a device data storage file, an error (error code: 21A1H) occurs.

When the CPU module is switched from STOP to RUN, it is checked to see whether the actual device data storage file matches the parameter setting. If they do not match, an error (error code: 21A0H) occurs.

Reading/Writing data in the specified file

Use the SP.FREAD instruction to read data from the specified file in the SD memory card. Use the SP.FWRITE instruction to write data to the specified file in the SD memory card.

When the SP.FREAD or SP.FWRITE instruction is terminated with an error, an error code is stored in the completion status.

The following table lists the error codes stored in the completion status.

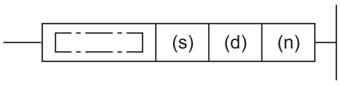
Error code	Error definition and cause	Action
8000H	• SM606 (SD memory card forced disable instruction) is on.	• If SM606 is on, turn it off to cancel the SD memory card forced disable status.
	• No SD memory card is inserted.	• Insert an SD memory card.
	• The SD memory card is not mounted.	• Mount the SD memory card.
8001H	• An access to the SD memory card has failed. • The SD memory card is full. • The SD memory card is write-protected. • The file is set for read only. • The file size has exceeded its limit.	• Check that the file name character string are correctly specified. • Delete unnecessary files in the SD memory card to secure free space. • Unlock the write protect switch of the memory card. • Clear the read-only setting. • Correct the file size so that it is equal to or less than its limit. • Take measures to reduce noise. • Reset the CPU module and run it again. If the same error code is displayed again, the possible cause is a hardware failure of the SD memory card. Please consult your local Mitsubishi representative.
8002H	• The specified file does not exist. • The disk has become full during writing data to a file.	• Check that the specified file exists. • Check that the specified folder path exists. • Delete unnecessary files in the SD memory card to secure free space.
8003H	• The total number of data read from the file has exceeded (d1)+3 (Maximum number of read data).	• Adjust (d1)+2 (Number of read-target data) or (d1)+3 (Maximum number of read data).

Reading data from the data memory

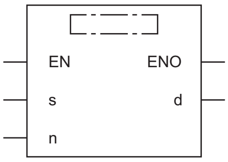
S(P).DEVLD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

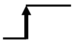
These instructions read data from the device data storage file in data memory.

Ladder	ST
	ENO:=S_DEVLD(EN,s,n,d); ENO:=SP_DEVLD(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DEVLD	
SP.DEVLD	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Read offset of device data storage file (specified in units of 16-bit words)	0 to 524287	32-bit unsigned binary	ANY32
(d)	Device for storing the data that has been read	—	Word	ANY16[*1]
(n)	Number of read points	1 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

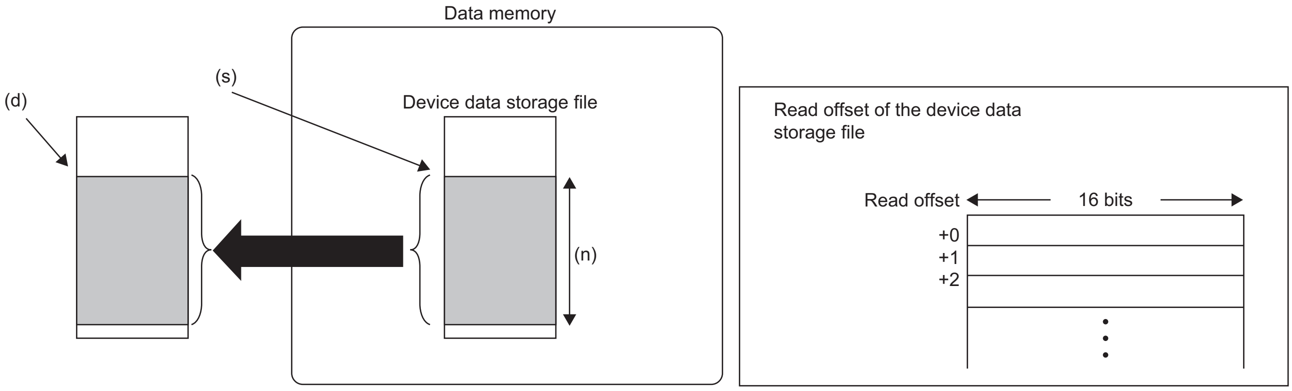
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions read device data by the number of points specified by (n) from the read offset specified by (s) in the device data storage file in data memory, and store it in the device specified by (d). (s) indicates the offset from the start of the device data storage file and can be specified by word offsets (incremented by 1 every 16 bits).



- When the S(P).DEVLD instruction is used, a device data storage file must be set up in advance. (☞ Page 593 Setting method)
- Use the SP.DEVST instruction to write device data to the device data storage file. (☞ Page 596 SP.DEVST)

Operation error

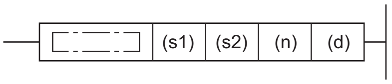
Error code (SD0)	**Description**
2840H	The device data storage file is not set in parameter.
3405H	The value specified by (n) is 0.

Writing data to the data memory

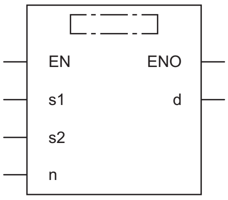
SP.DEVST

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

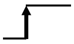
This instruction writes the specified number of points of data to the device data storage file in data memory.

Ladder	ST
	ENO:=SP_DEVST(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.DEVST	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Write offset of device data storage file (specified in units of 16-bit words)	0 to 524287	32-bit unsigned binary	ANY32
(s2)	Start device to which data is to be written	—	Word	ANY16[1]
(n)	Number of write points	1 to 65535	16-bit unsigned binary	ANY16
(d)	(d): Completion device, (d)+1: Error completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

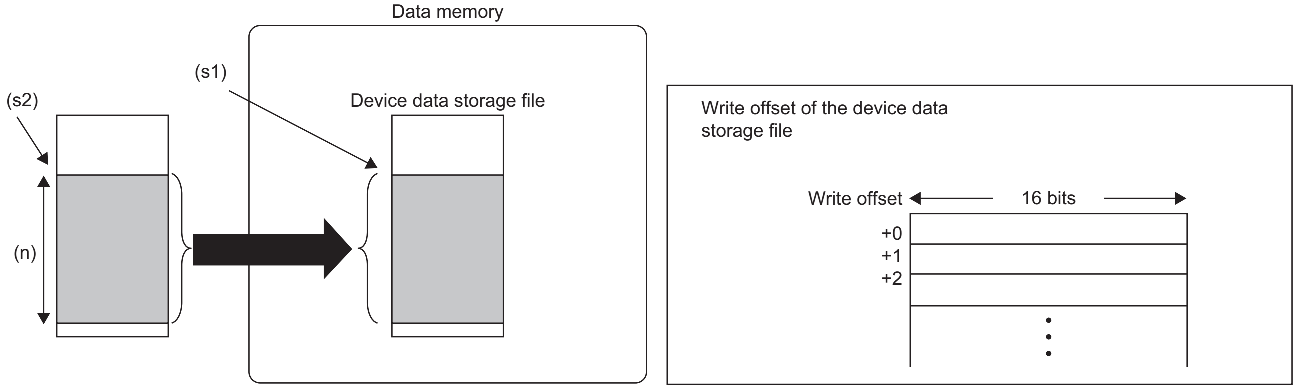
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions retrieves the specified number of points of data specified by (n) from the device specified by (s2) and write it to the write offset in the device specified by (s1) in the device data storage file in data memory. (s1) indicates the offset from the start of the device data storage file and can be specified by word offsets (incremented by 1 every 16 bits).



- The completion device specified by (d) automatically turns on upon execution of the END instruction following the detection of processing completion of the SP.DEVST instruction and turns off upon execution of the END instruction in the next scan, so it is used as the execution complete flag of the SP.DEVST instruction.
- If the SP.DEVST instruction completes with an error, the error completion device specified by (d)+1 turns on or off at the same time as the completion device specified by (d). Therefore, the device is used as the error completion flag of the SP.DEVST instruction.
- SM753 (File being accessed) turns on while the SP.DEVST instruction is executed. If SM753 has already been on, the SP.DEVST instruction cannot be executed. (If executed, no processing is performed.)
- If an error is detected during execution of the SP.DEVST instruction, the completion device (d), error completion device (d)+1, and SM753 do not turn on.
- When the S(P).DEVST instruction is used, a device data storage file must be set up in advance. (☞ Page 593 Setting method)
- Use the S(P).DEVLD instruction to read device data from the device data storage file to any specified device. (☞ Page 594 S(P).DEVLD)

Precautions

- The value written to data memory is the one at execution of the SP.DEVST instruction.
- Execution of the SP.DEVST instruction increases SD634 and SD635. The number of writes to the data memory of the CPU module is limited. If the data memory write count index exceeds 100,000, an error occurs with error code 1080H.
- To prevent the data memory write count from being increased by careless instruction execution, SD771 can be set to limit the write count per day. The maximum number of writes is 36 by default. Change the maximum number of writes by using SD771 as needed. If the specified write count is exceeded, an error occurs with error code 3421H. The number of executions of the instruction to write to data memory per day is initialized to 0 at the following timing.
  - When power off→on, or when reset→reset canceled
  - The date (year, month, day) in clock data is changed by time advancement.
  - CPU module internal clock data (year, month, day) is changed by the clock data change function.
- Data is written to the device data storage file when the END instruction is executed. Data is written to the device data storage file when the END instruction is executed immediately after the SP.DEVST instruction is executed. Thus, depending on the number of write points, writing to the device data storage file may involve multiple scans. Check the completion device to see whether the writing is completed.

Operation error

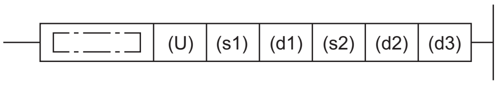
Error code (SD0)	Description
2840H	The device data storage file is not set in parameter.
3405H	The value specified by (n) is 0.
3421H	When the SP.DEVST instruction is executed, the write count of the day exceeds the value specified in SD771.
	When the SP.DEVST instruction is executed, a value out of the range (1 to 32767) is set in SD771.

Reading data from the specified file

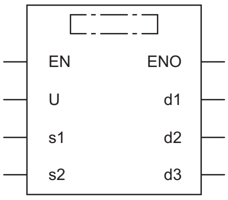
SP.FREAD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

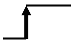
This instruction reads device data from the specified file on the SD memory card.

Ladder	ST
	ENO:=SP_FREAD(EN,U,s1,s2,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FREAD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)[*1]	Word	ANY16
(d1)	Start device where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 8)
(s2)	Start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Start device for storing the data that has been read	—	Word	ANY16[*2]
(d3)	Bit device that turns on upon completion of the processing (In the case of an error completion, the device specified by (d3)+1 also turns on.)	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] Only drive 2 (for the SD memory card) can be set.

[*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

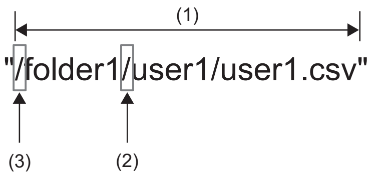
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	○	—	○	—	—	—	—	—	—	—	—	—

■For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

- Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. ■00**H: Reading binary data • 0000H: 16-bit binary data • 0001H: 32-bit binary data ■01**H: Reading data after converted to CSV format • 0100H: Decimal (16-bit data) • 0110H: Decimal (32-bit data) • 0120H: Hexadecimal (16-bit data) • 0121H: Hexadecimal (32-bit data) • 0130H: String (ASCII data) • 0140H: Floating point real number (single-precision real number) • 0141H: Floating point real number (double-precision real number)	0000H 0001H 0100H 0110H 0120H 0121H 0130H 0140H 0141H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Number of read-target data	Specify the number of read-target data (in units of words). The processing unit and setting range depend on (d1)+0. ■When "Reading binary data" is specified by (d1)+0 • When 16-bit binary data is specified: In units of words (1 to 65535) • When 32-bit binary data is specified: In units of double words (1 to 32767) ■When "Reading data after converted to CSV format" is specified by (d1)+0 • When decimal (16-bit data) is specified: Number of elements (1 to 65535) • When decimal (32-bit data) is specified: Number of elements (1 to 32767) • When hexadecimal (16-bit data) is specified: Number of elements (1 to 65535) • When hexadecimal (32-bit data) is specified: Number of elements (1 to 32767) • When a string (ASCII data) is specified: Number of elements (1 to 1023) • When a floating point real number (single-precision real number) is specified: Number of elements (1 to 32767) • When a floating point real number (double-precision real number) is specified: Number of elements (1 to 16383)	1 to 65535	User
+3	Maximum number of read data	■When "0130H: String (ASCII data)" is specified by (d1)+0 • Total size of the characters in the element (in units of words) ■When data other than "0130H: String (ASCII data)" is specified by (d1)+0 • Fixed to 0	0, 1 to 65535	User
+4 +5	File position	■When "Reading binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Cannot be specified. ■When "Reading data after converted to CSV format" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified row • FFFFFFFFH: From the position where the previous reading ends	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Reading binary data" is specified by (d1)+0, set 0. When "Reading data after converted to CSV format" is specified by (d1)+0, set the number of read-target columns. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	• 0: Word • 1: Byte • 2: Unit of the data type specified by (d1)+0 "0: Word" and "1: Byte" can be specified only when "0000H: 16-bit binary data" or "0100H: Decimal (16-bit data)" is specified by (d1)+0.	0, 1, 2	User

- File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the folder path where the files are stored and the file name. • The folder path and file name (including an extension) must be within 253 characters in total. • The folder path must be within 244 characters. (A delimiter is not included.) • The number of folder path hierarchies must be within 10 levels. • When omitting an extension in the file name, omit the ". (period)" as well. • The file name must be within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".  (1): Up to 253 characters (2): Use "\" or "/" as a delimiter of the folder path and file. (3): Can be omitted. When omitted, (1) is up to 252 characters.	Unicode string	User

- Read data (d2)

Operand: (d2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of data actually read	The number of data actually read is set, corresponding to the number of data specified in (d1)+2. The data unit is determined by (d1)+7 (Data type specification).	—	System
+1 to +□	Read data	The read data is stored.	—	System

■For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU

- Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. • 0000H: Reading binary data • 0100H: Reading data after converted to CSV format	0000H 0100H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Number of read-target data	Specify the number of read-target data (in units of words). (in units of words) Even when "Byte" is specified by (d1)+7, specify the number in units of words.	1 to 65535	User
+3	Not used	—	—	—
+4 +5	File position	■When "Reading binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Cannot be specified. ■When "Reading data after converted to CSV format" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified row • FFFFFFFFH: From the position where the previous reading ends	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Reading binary data" is specified by (d1)+0, set 0. When "Reading data after converted to CSV format" is specified by (d1)+0, set the number of read-target columns. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	0: Word 1: Byte	0, 1	User

- File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the string of the file name. When omitting an extension, omit the ". (period)" as well. Specify the name within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".	Unicode string	User

- Read data (d2)

Operand: (d2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of data actually read	The number of data actually read is set. The data unit is determined by (d1)+7.	—	System
+1 to +□	Read data	The read data is stored.	—	System

Processing details

- This instruction reads data from the specified file. Set the execution/completion type in the control data to specify the file read-target format.

☞ Page 600 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

☞ Page 602 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU

- The read target is the data in the SD memory card only.
- The bit device (d3) is used as the execution completion flag of the instruction. The bit automatically turns on upon execution of the END instruction after the instruction processing completion is detected, and the bit turns off upon execution of the END instruction in the next scan.
- If the instruction completes with an error, the bit device (d3)+1 turns on in synchronization with (d3). Therefore, the bit device (d3)+1 can be used as the error completion flag.
- SM753 turns on while the SP.FREAD instruction is being executed.
- While SM753 is on, the instruction cannot be executed. (If executed, no processing is performed.)
- If an error is detected during the execution of the instruction, Processing Complete (d3), Error Completion (d3)+1, and SM753 do not turn on.
- Specify data in (d1)+2, (d1)+4, (d1)+5, and (d2)+0 depending on the combination of (d1)+0 and (d1)+7.

Execution/completion type (d1)+0		**Data type specification (d1)+7**	**Processing unit and setting range**		
			Number of read-target data (d1)+2	**File position (d1)+4, (d1)+5**	**Number of data actually read (d2)+0**
Reading binary data	0000H: 16-bit binary data	0: Word	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH)	Word
		1: Byte	Word (1 to 65535)	Byte (00000000H to FFFFFFFEH)	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH)	Word
	0001H: 32-bit binary data	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Double word (00000000H to 3FFFFFFFH)	Double word

Execution/completion type (d1)+0		Data type specification (d1)+7	Processing unit and setting range		
			Number of read-target data (d1)+2	File position (d1)+4, (d1)+5	Number of data actually read (d2)+0
Reading data after converted to CSV format	0100H: Decimal (16-bit data)	0: Word	Number of elements (1 to 65535)	Number of lines	Word
		1: Byte	Number of elements (1 to 65535)	Number of lines	Byte
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 65535)	Number of lines	Word
	0110H: Decimal (signed 32-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
	0120H: Hexadecimal (16-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 65535)	Number of lines	Word
	0121H: Hexadecimal (32-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
	0130H: String (ASCII data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 1023)	Number of lines	Number of elements
	0140H: Floating point real number (single-precision real number)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
	0141H: Floating point real number (double-precision real number)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 16383)	Number of lines	4 words

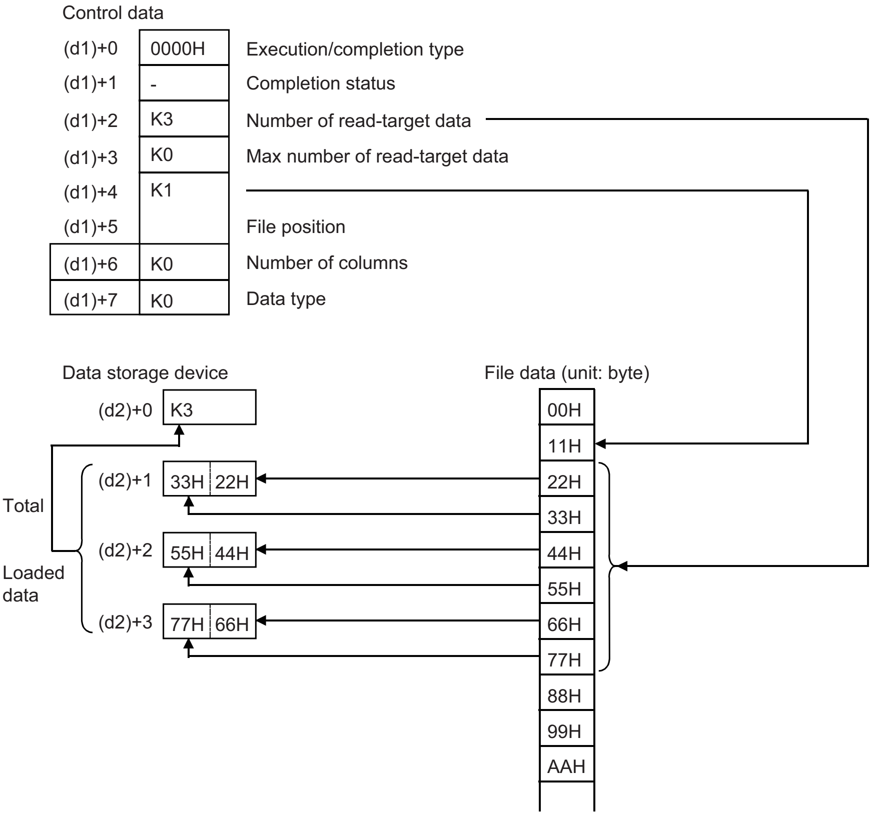
■When reading binary data

- If the extension of the target file is omitted, the extension will be ".BIN".
- If the specified file does not exist, an error occurs.
- If the position is specified exceeding the existing file size, 0 point of data is read and the processing completes successfully.

The following figure shows an example of reading binary data.

Ex.

Example of reading binary data (16-bit binary data)



■When reading data after converted to CSV format

- Elements in the CSV format file (cells in Excel®) are read by each row and stored in the device.
- If the extension of the target file is omitted, the extension will be ".CSV".
- If the specified file does not exist, an error occurs.
- If the position is specified exceeding the existing file size, 0 point of data is read and the processing completes successfully.
- The number of data specified by (d1)+2 is read from the beginning of the file. If the last data in the file is read before reaching to the number specified, only the available number of data is read.
- When the number of columns is set to 0, the data are read by ignoring the rows in the CSV format file.
- The string data in the CSV file and the value stored in the device after it is read are determined by the execution/completion type.

Execution/completion type	Data (one element) in the CSV file	Value stored in the device	Remarks
0100H: Decimal (16-bit data)	-32768 to -1	-32768 to -1 (32768 to 65535)	The value should be within the range of -32768 to -1 as signed 16-bit data and within the range of 32768 to 65535 as unsigned 16-bit data. The same value is stored in the device.
	0 to 32767	0 to 32767	—
	32768 to 65535	-32768 to -1 (32768 to 65535)	The value should be within the range of -32768 to -1 as signed 16-bit data and within the range of 32768 to 65535 as unsigned 16-bit data. The same value is stored in the device.
	• Numeric values other than above • String containing alphabets and symbols	0	Filled with 0 since it cannot be converted.
0110H: Decimal (32-bit data)	-2147483648 to -1	-2147483648 to -1 (2147483648 to 4294967295)	The value should be within the range of -2147483648 to -1 as signed 32-bit data and within the range of 2147483648 to 4294967295 as unsigned 32-bit data. The same value is stored in the device.
	0 to 2147483647	0 to 2147483647	—
	2147483648 to 4294967295	-2147483648 to -1 (2147483648 to 4294967295)	The value should be within the range of -2147483648 to -1 as signed 32-bit data and within the range of 2147483648 to 4294967295 as unsigned 32-bit data. The same value is stored in the device.
	• Numeric values other than above • String containing alphabets and symbols	0	Filled with 0 since it cannot be converted.
0120H: Hexadecimal (16-bit data)	0H to FFFFH	0H to FFFFH	—
	• Numeric values other than above • String containing alphabets other than A to F and symbols	0000H	Filled with 0 since it cannot be converted.
0121H: Hexadecimal (32-bit data)	0H to FFFFFFFFH	0H to FFFFFFFFH	—
	• Numeric values other than above • String containing alphabets other than A to F and symbols	00000000H	Filled with 0 since it cannot be converted.
0130H: String (ASCII data)	String (up to 1999 characters)	String (up to 1999 characters)	NULL (00H) is added to the end of the string. When the number of bytes of the string in the CSV file is even, 0000H is stored in the next one word. When the string in the CSV file contains 00H, it is ignored.
	String (2000 characters or more)		If the number of characters in one element exceeds 1999, characters until the 1999th character are read as one element. The 2000th character and after are not read and the next element is read.

Execution/completion type	Data (one element) in the CSV file	Value stored in the device	Remarks
0140H: Floating point real number (single-precision real number)	Values within the range of: $-2^{128} < \text{data} \leq -2^{-126}$, 0, $2^{-126} \leq \text{data} < 2^{128}$	As given on the left	The value is converted in either decimal point or exponential format.
	Numeric values other than above	0	Filled with 0 since it cannot be converted.
0141H: Floating point real number (double-precision real number)	Values within the range of: $-2^{1024} < \text{data} \leq -2^{-1022}$, 0, $2^{-1022} \leq \text{data} < 2^{1024}$	As given on the left	The value is converted in either decimal point or exponential format.
	Numeric values other than above	0	Filled with 0 since it cannot be converted.

Ex.

When "Reading data after converted to CSV format" (String (ASCII data)) is specified

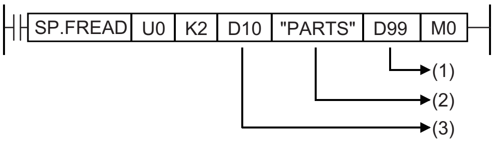
[Data stored in CSV format]

PARTS.CSV

No.	,	Name	,	Value1	,	Value2	CR	LF
AA_0001	,	Prts_A	,	100	,	200	CR	LF
BB_0002	,	Prts_B	,	300	,	400	CR	LF

Enclosed values as shown to the left are read.
(Total of six elements, three columns from two lines of PARTS.CSV, are read.)

[Data to be loaded to the device]

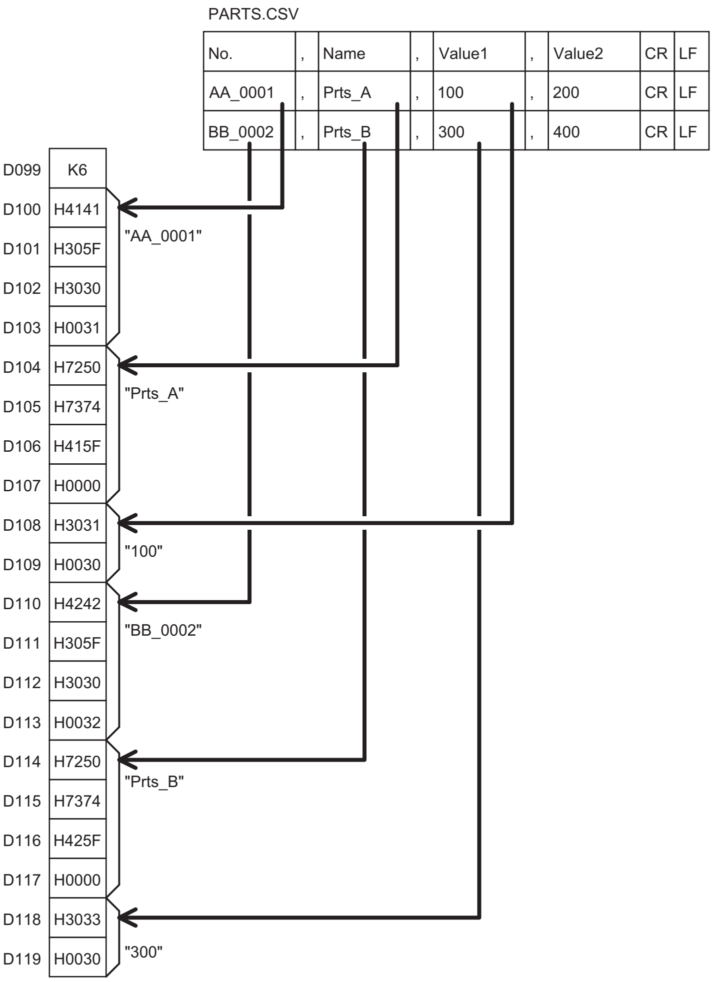


(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	H0130
D11	H0000
D12	K6
D13	K100
D14	K2
D15	
D16	K3
D17	K2

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File position
D16: Number of columns
D17: Data type specification

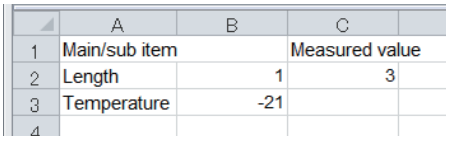


D99: Number of data actually read
D100 to D103: String in the 1st column of the 2nd line
D104 to D107: String in the 2nd column of the 2nd line
D108, D109: String in the 3rd column of the 2nd line
D110 to D113: String in the 1st column of the 3rd line
D114 to D117: String in the 2nd column of the 3rd line
D118, D119: String in the 3rd column of the 3rd line

Ex.

When "Reading data after converted to CSV format" (Decimal (16-bit data)) is specified and the number of columns is set to 0

[Data created in Excel]

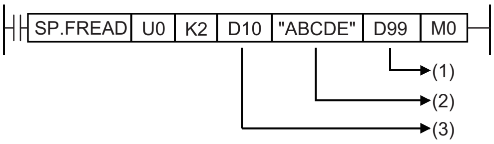


	A	B	C
1	Main/sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

[Data saved in CSV format]

Main/sub item	,	,	Measured value	CR	LF	
Length	,	1	,	3	CR	LF
Temperature	,	-21	,	CR	LF	

[Data to be loaded to the device]



(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	0100H
D11	-
D12	K9
D13	K0
D14	K0
D15	K0
D16	K0
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File position
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K9
D100	K0
D101	K0
D102	K0
D103	K0
D104	K1
D105	K3
D106	K0
D107	K-21
D108	K0

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Measured value" is not a numerical value, the conversion data, 0, is stored.
D103: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D104: Since "1" is a numerical value, it is converted to a binary value.
D105: Since "3" is a numerical value, it is converted to a binary value.
D106: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D107: Since "-21" is a numerical value, it is converted to a binary value.
D108: Since " " is not a numerical value, the conversion data, 0, is stored.

- When the number of columns differs in each row, the data are also read by ignoring the rows.

Ex.

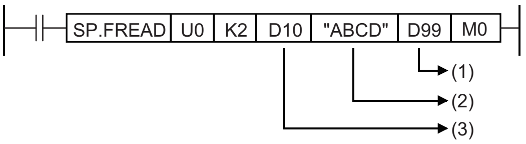
When the number of columns differs in each row

[Data saved in CSV format]

Main/sub item	,	,	Measured value	,	Excess	CR	LF
Length	CR	LF					
Temperature	,	-21	,	CR	LF		

↓

[Data to be loaded to the device]



(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	0100H
D11	-
D12	K7
D13	K0
D14	K0
D15	K0
D16	K0
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File position
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K7
D100	K0
D101	K0
D102	K0
D103	K0
D104	K0
D105	K0
D106	K-21

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Measured value" is not a numerical value, the conversion data, 0, is stored.
D103: Since "Excess" is not a numerical value, the conversion data, 0, is stored.
D104: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D105: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D106: Since "-21" is a numerical value, it is converted to a binary value.

Point

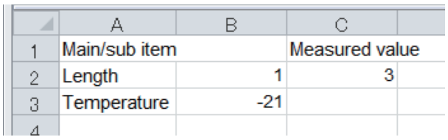
This type of file in which the number of columns vary with individual rows cannot be created by Excel. It is created when the CSV file is modified by a user.

- When the specified number of columns is set to a value other than 0, a CSV format file is read as the table with the specified number of columns. The elements outside the specified number of columns are ignored.

Ex.

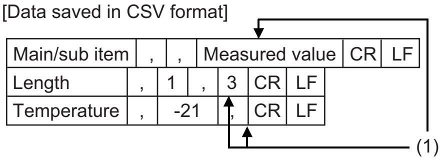
When "Reading data after converted to CSV format" (Decimal (16-bit data)) is specified and the number of columns is set to a value other than 0 ((d1)+6 is set to 2)

[Data created in Excel]



	A	B	C
1	Main/sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

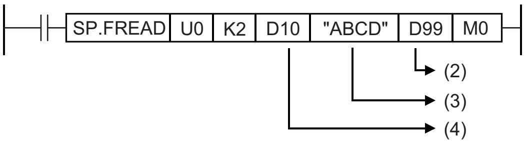
[Data saved in CSV format]



Main/sub item , , Measured value CR LF
Length , 1 , 3 CR LF
Temperature , -21 , CR LF

(1) Columns out of the specified range are ignored.

[Data to be loaded to the device]



(2) Loaded data
(3) File name
(4) Control data

[Control data]

D10	0100H
D11	-
D12	K6
D13	K0
D14	K0
D15	K0
D16	K2
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File position
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K6
D100	K0
D101	K0
D102	K0
D103	K1
D104	K0
D105	K-21

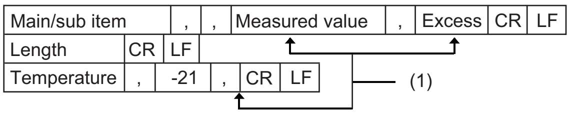
D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D103: Since "1" is a numerical value, it is converted to a binary value.
D104: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D105: Since "-21" is a numerical value, it is converted to a binary value.

- When the number of columns differs in each row, the elements outside the specified number of columns are ignored and 0 is added to the cells where no element exists.

Ex.

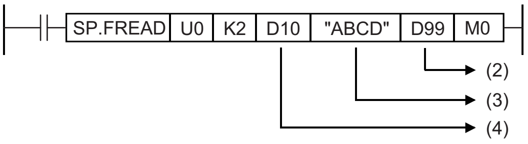
When the number of columns differs in each row

[Data saved in CSV format]



(1) Columns out of the specified range are ignored.

[Data to be read into devices]



(2) Loaded data
(3) File name
(4) Control data

[Control data]

D10	0100H
D11	-
D12	K6
D13	K0
D14	K0
D15	K0
D16	K2
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File position
D16: Number of columns
D17: Data type specification

[Loaded data]

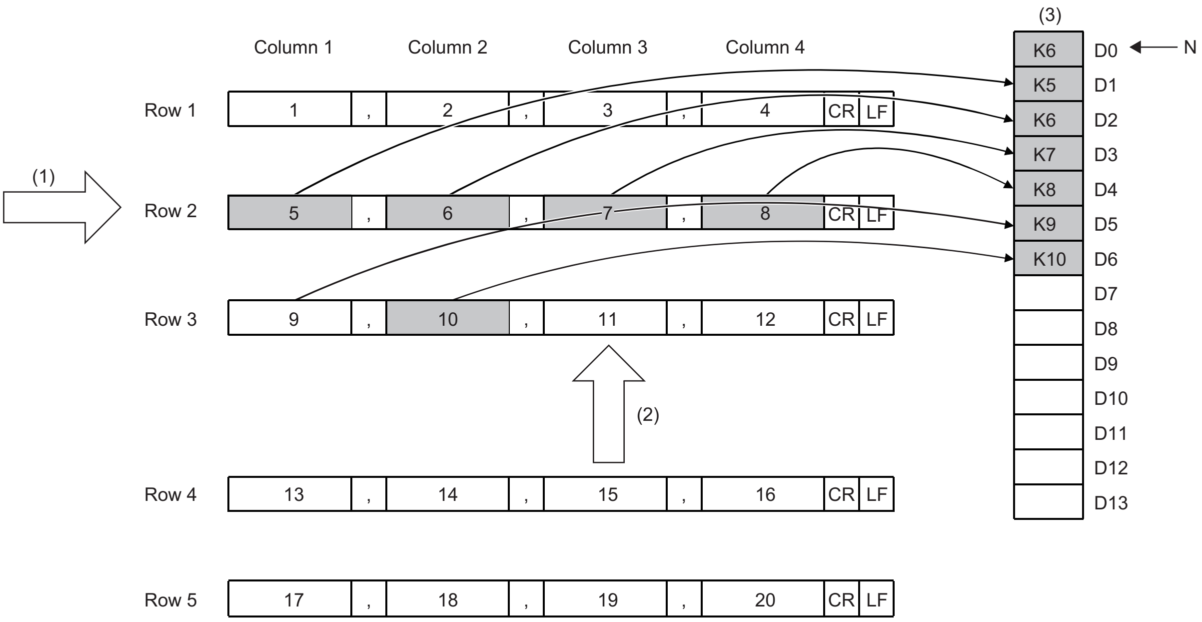
D99	K6
D100	K0
D101	K0
D102	K0
D103	K0
D104	K0
D105	K-21

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D103: Since no element exists, the conversion data, 0, is added.
D104: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D105: Since "-21" is a numerical value, it is converted to a binary value.

- When "Reading data after converted to CSV format" is specified, data can be divided and read.

[Specify row to start reading]

- Execution/completion type: Reading data after converted to CSV format (Decimal (16-bit data))
- Number of columns specification: 4H
- Data type specification: Words
- File position: 2H
- Read start device: D0
- Number of data actually read: 6H



(1) Starting row

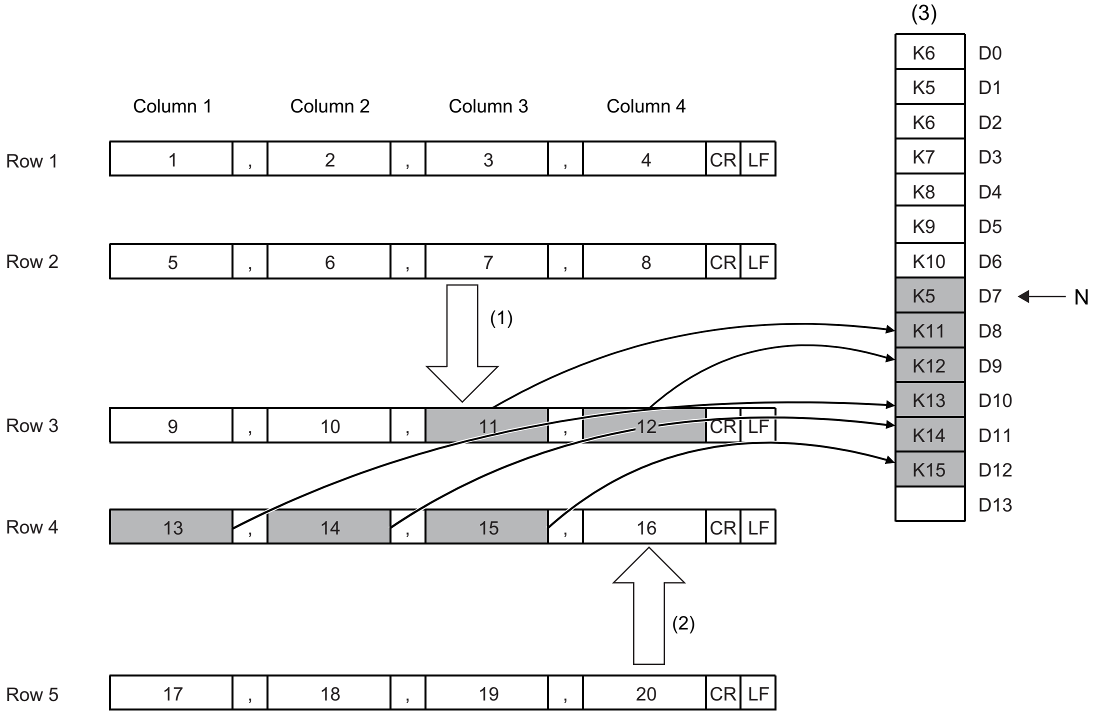
(2) Next starting position

(3) Data in the device (loaded data)

N: Number of data

[Reading data from the position where the previous reading ends]

- Execution/completion type: Reading data after converted to CSV format (Decimal (16-bit data))
- Number of columns specification: 4H
- Data type specification: Words
- File position: FFFFFFFFH (continuing from the position where the previous reading ends)
- Read start device: D7
- Number of data actually read: 5H



(1) Starting row

(2) Next starting position

(3) Data in the device (loaded data)

N: Number of data

Point

- When reading data from the position where the previous reading ends, specify the same values for "Execution/End type", "Number of columns", and "Data type specification". If not, data cannot be added correctly from the position where the previous reading ends.
- While reading data from the position where the previous reading ends, if the SP.FREAD instruction with different settings or the SP.FWRITE instruction is executed, data cannot be added correctly from the position where the previous reading ends.

Precautions

- Do not execute this instruction in interrupt programs. Doing so may cause malfunction of the module.
- When reading multiple elements at reading data after converted to CSV format, secure devices sufficient for the total size of the elements in the read data area before executing the instruction. Since read data is stored from (d2)+1, the number of words required to be secured as (d2) is ((total number of words for each element) + 1) words.
- When "Reading data after converted to CSV format" (String (ASCII data)) is specified, set the total size (in units of words) to (d1)+3 (Maximum number of read data).

[Example] When reading 100 elements each of which contains 100 characters from a CSV file
(100 (characters) + 2 (NULL)) × 100 (elements) = 10200 bytes = 5100 words
Therefore, set 5100 to (d1)+3 to secure an area for 5101 words for (d2).

Operation error

Error code (SD0)	Description
2820H	Data is read exceeding the size of the device.
3405H	The drive specified by (s1) is not the one for the SD memory card.
	Any value that is set in the device specified by (d1) and later as control data is out of the range. ☞ Page 600 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later) ☞ Page 602 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU
	The file name character string specified by (s2) cannot be read. • The number of characters of the string in the file name specified exceeds the range. • An inhibited value is set. ☞ Page 600 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later) ☞ Page 602 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU
3427H	An invalid combination of (d1)+0 (Execution/completion type) and (d1)+7 (Data type specification) is specified. ☞ Page 600 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

When the SP.FREAD instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

For the error code stored in (d1)+1, refer to the following.

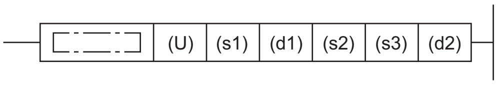
☞ Page 593 Reading/Writing data in the specified file

Writing data to the specified file

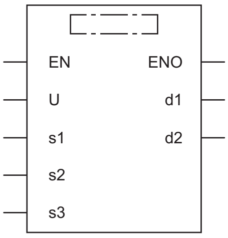
SP.FWRITE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

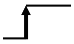
This instruction writes device data to the specified file on the SD memory card.

Ladder	ST
	ENO:=SP_FWRITE(EN,U,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FWRITE	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)[*1]	Word	ANY16
(d1)	Start device where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 8)
(s2)	Start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device where data is stored	—	Word	ANY16[*2]
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only drive 2 (for the SD memory card) can be set.

*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

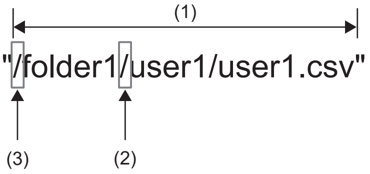
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	—	—	—	—	—

■For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

- Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. ■00**H: Writing binary data • 0000H: 16-bit binary data • 0001H: 32-bit binary data ■01**H: Writing data after converted to CSV format • 0100H: Decimal (signed 16-bit data) • 0101H: Decimal (unsigned 16-bit data) • 0110H: Decimal (signed 32-bit data) • 0111H: Decimal (unsigned 32-bit data) • 0120H: Hexadecimal (16-bit data) • 0121H: Hexadecimal (32-bit data) • 0130H: String (ASCII data) • 0140H: Floating point real number (single-precision real number) • 0141H: Floating point real number (double-precision real number)	0000H 0001H 0100H 0101H 0110H 0111H 0120H 0121H 0130H 0140H 0141H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Number of data actually written	For the data in the device specified by (s3), the number of data actually written is stored. The data unit is determined by (d1)+7 (Data type specification).	—	System
+3	Not used	—	—	—
+4 +5	File position	■When "Writing binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Added to the end of the file. ■When "Converting and writing data in CSV format" is specified by (d1)+0 • 00000000H to FFFFFFFEH: From the beginning of the file • FFFFFFFFH: Added to the end of the file.	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Writing binary data" is specified by (d1)+0, set 0. When "Converting and writing data in CSV format" is specified by (d1)+0, specify the number of columns to write. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	• 0: Word • 1: Byte • 2: Unit of the data type specified by (d1)+0 "0: Word" and "1: Byte" can be specified only when "0000H: 16-bit binary data" or "0100H: Decimal (signed 16-bit data)" is specified by (d1)+0.	0, 1, 2	User

- File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the folder path where the files are stored and the file name. • The folder path and file name (including an extension) must be within 253 characters in total. • The folder path must be within 244 characters. (A delimiter is not included.) • The number of folder path hierarchies must be within 10 levels. • When omitting an extension in the file name, omit the ". (period)" as well. • The file name must be within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".  (1): Up to 253 characters (2): Use "\" or "/" as a delimiter of the folder path and file. (3): Can be omitted. When omitted, (1) is up to 252 characters.	Unicode string	User

- Write data (s3)

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of request write data	Specify the number of data to be requested to write. The processing unit and setting range depend on the value set in (d1)+0. ■When "Writing binary data" is specified by (d1)+0 • When 16-bit binary data is specified: In units of words (1 to 65535) • When 32-bit binary data is specified: In units of double words (1 to 32767) ■When "Converting and writing data in CSV format" is specified by (d1)+0: Number of elements • When decimal (signed 16-bit data) or decimal (unsigned 16-bit data) is specified: In units of words (1 to 65535) • When decimal (signed 32-bit data) or decimal (unsigned 32-bit data) is specified: In units of double words (1 to 32767) • When hexadecimal (16-bit data) is specified: In units of words (1 to 65535) • When hexadecimal (32-bit data) is specified: In units of double words (1 to 32767) • When a string (ASCII data) is specified: Number of elements (1 to 1023) • When a floating point real number (single-precision real number) is specified: In units of double words (1 to 32767) • When a floating point real number (double-precision real number) is specified: In units of 4 words (1 to 16383)	1 to 65535	User
+1 to +□	Write data	The number of data to be requested to write is stored.	0000H to FFFFH	User

■For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU

- Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. • 0000H: Writing binary data • 0100H: Writing data after converted to CSV format	0000H 0100H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Number of data actually written	For the data in the device specified by (s3), the number of data actually written is stored. The data unit is determined by (d1)+7.	—	System
+3	Not used	—	—	—
+4 +5	File position	■When "Writing binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Added to the end of the file. ■When "Converting and writing data in CSV format" is specified by (d1)+0 • 00000000H to FFFFFFFEH: From the beginning of the file • FFFFFFFFH: Added to the end of the file.	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Writing binary data" is specified by (d1)+0, set 0. When "Converting and writing data in CSV format" is specified by (d1)+0, specify the number of columns to write. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	0: Word 1: Byte	0, 1	User

- File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the string of the file name. When omitting an extension, omit the ". (period)" as well. Specify the name within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".	Unicode string	User

- Write data (s3)

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of request write data	Specify the number of data to be requested to write. (in units of words) Even when "Byte" is specified by (d1)+7, specify the number in units of words.	1 to 65535	User
+1 to +□	Write data	The number of data to be requested to write is stored.	0000H to FFFFH	User

Processing details

- This instruction writes the specified number of data to the specified file. Set the execution/completion type in the control data to specify the file write-target format.

☞ Page 617 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

☞ Page 619 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU

- The write target is the SD memory card only.
- The bit device (d2) is used as the execution completion flag of the instruction. The bit automatically turns on upon execution of the END instruction after the instruction processing completion is detected, and the bit turns off upon execution of the END instruction in the next scan.
- If the instruction completes with an error, the bit device (d2)+1 turns on in synchronization with (d2). Therefore, the bit device (d2)+1 can be used as the error completion flag.
- SM753 turns on while the SP.FWRITE instruction is executed.
- While SM753 is on, the SP.FWRITE instruction cannot be executed. (If executed, no processing is performed.)
- Even if an error is detected during the execution of the instruction, the bit devices (d2) and (d2)+1 and SM753 do not turn on.
- Specify data in (s3)+0, (d1)+4, (d1)+5, and (d1)+2 depending on the combination of (d1)+0 and (d1)+7.

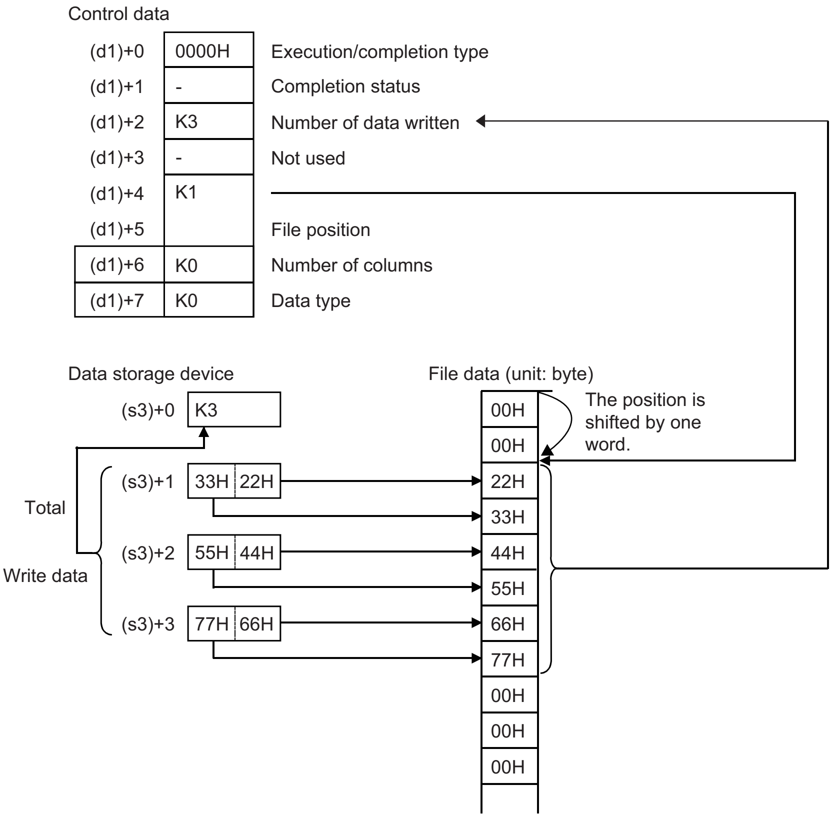
Execution/completion type (d1)+0		**Data type specification (d1)+7**	**Processing unit and setting range**		
			Number of request write data (s3)+0	**File position (d1)+4, (d1)+5**	**Number of data actually written (d1)+2**
Writing binary data	0000H: 16-bit binary data	0: Word	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH, FFFFFFFFH)	Word
		1: Byte	Word (1 to 65535)	Byte (00000000H to FFFFFFFFH)	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH, FFFFFFFFH)	Word
	0001H: 32-bit binary data	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Double word (00000000H to 3FFFFFFFH, FFFFFFFFH)	Double word

Execution/completion type (d1)+0		Data type specification (d1)+7	Processing unit and setting range		
			Number of request write data (s3)+0	File position (d1)+4, (d1)+5	Number of data actually written (d1)+2
Writing data after converted to CSV format	0100H: Decimal (signed 16-bit data)	0: Word	Word (1 to 65535)	Head/end[1]	Word
		1: Byte	Word (1 to 65535)	Head/end[1]	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end[1]	Word
	0101H: Decimal (unsigned 16-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end[1]	Word
	0110H: Decimal (signed 32-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end[1]	Double word
	0111H: Decimal (unsigned 32-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end[1]	Double word
	0120H: Hexadecimal (16-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end[1]	Word
	0121H: Hexadecimal (32-bit data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end[1]	Double word
	0130H: String (ASCII data)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 1023)	Head/end[1]	Number of elements
	0140H: Floating point real number (single-precision real number)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end[1]	Double word
	0141H: Floating point real number (double-precision real number)	0: Word 1: Byte	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	4 words (1 to 16383)	Head/end[1]	4 words

[1] The set value is from 00000000H to FFFFFFFEH (from the beginning of the file) or FFFFFFFFH (added to the end of the file).

■Writing binary data

- If the extension of the target file is omitted, the extension will be ".BIN".
- If a file that does not exist is specified, it will be created and data will be saved to the beginning of the file. The newly created file has the archive attribute.
- When an existing file is specified, data will be saved to the beginning of the file. If the size of data exceeds the size of the existing area in the file during writing, the excess data is additionally stored.
- If the specified position exceeds the existing file size, 0 point of data is written and the processing completes successfully.
- If the media runs out of free space during additional saving of data, an error occurs. In this case, the data already added and saved successfully is held as is and as much remaining data as possible is added and saved before termination with an error.
- In 16-bit binary data write mode, when the number of request write data and file position are specified, data is written as follows.

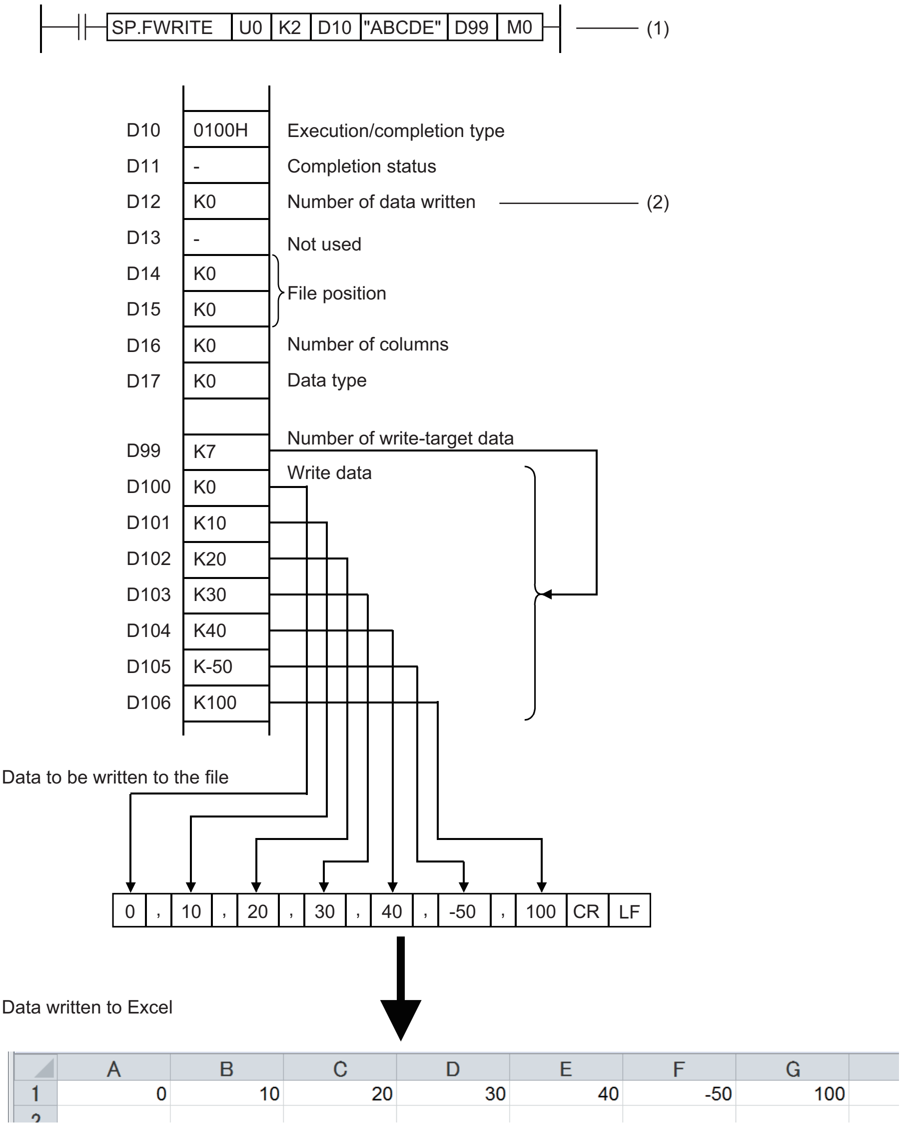


■Converting and writing data in CSV format

- If the extension is omitted, the extension will be ".CSV".
- When an existing file is specified, the following occurs.
  - When a value other than FFFFFFFFH is specified in (d1)+4 or (d1)+5, data is saved to the file after deleting the existing data in the file.
  - When a value other than FFFFFFFFH is specified in (d1)+4 or (d1)+5, data is added and saved to the end of the file.
- If a file that does not exist is specified, it will be created and data will be saved to the beginning of the file. The newly created file has the archive attribute.
- If the media runs out of free space during additional saving of data, an error occurs. In this case, the data already added and saved successfully is held as is and as much remaining data as possible is added and saved before termination with an error.
- When the number of columns is set to 0, data is read as a single-row data in a CSV format file.

Ex.

The number of columns is set to 0 when writing data after conversion to the CSV format.



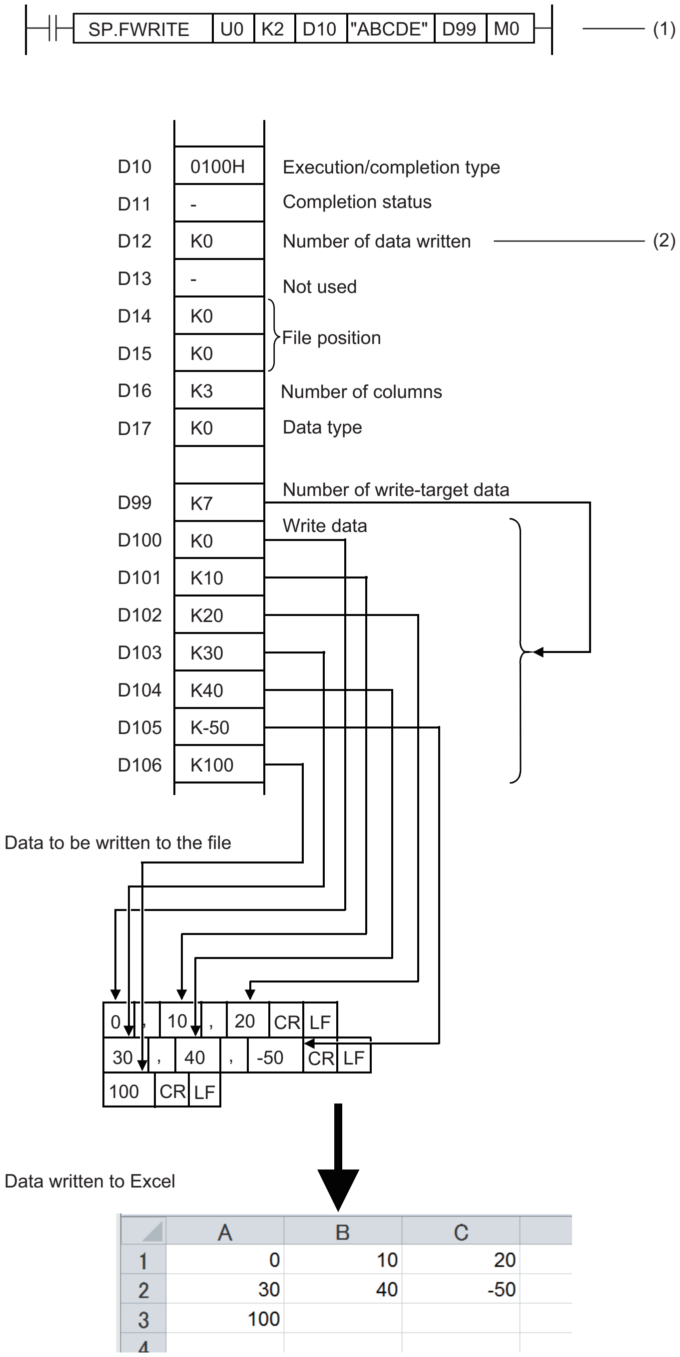
(1) Specified in units of words

(2) Same as the number of write data if case of normal completion

- When the specified number of columns is set to a value other than 0, a CSV format file is stored as the table with the specified number of columns.

Ex.

The number of columns is set to a value other than 0 when writing data after conversion to the CSV format.



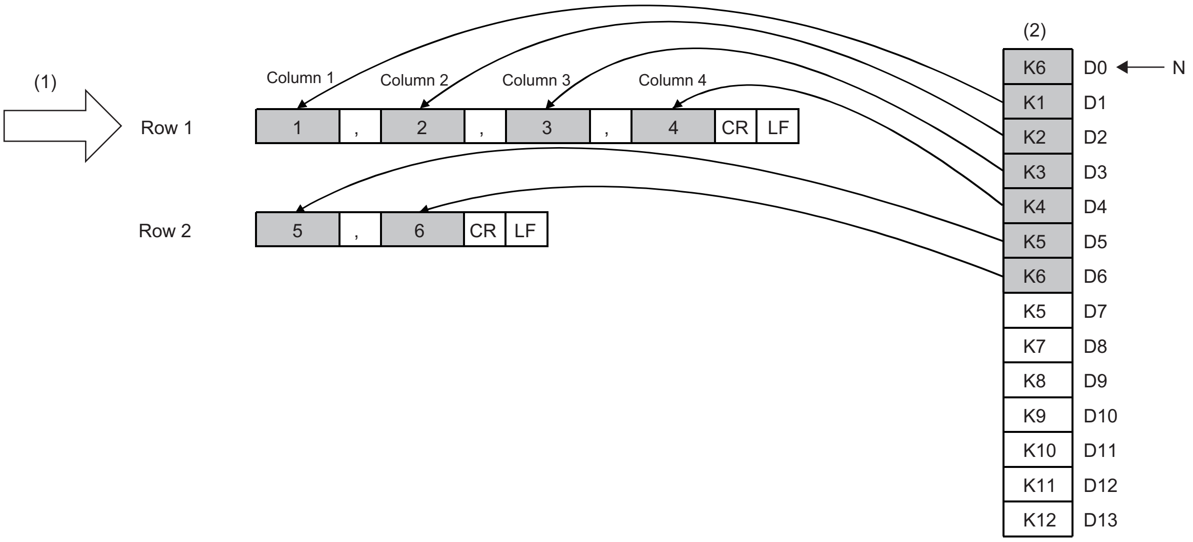
(1) Specified in units of words

(2) Same as the number of write data if case of normal completion

- The following figure shows how data is added.

[Specify the file to which data will be written.] (Even if the file exists, it is deleted and re-created.)

- Execution/completion type: Writing data after converted to CSV format (Decimal (signed 16-bit data))
- Number of columns specification: 4H
- Data type specification: Words
- File position: 00000000H (from the beginning of the file)
- Write start device: D0
- Number of request write data: 6H



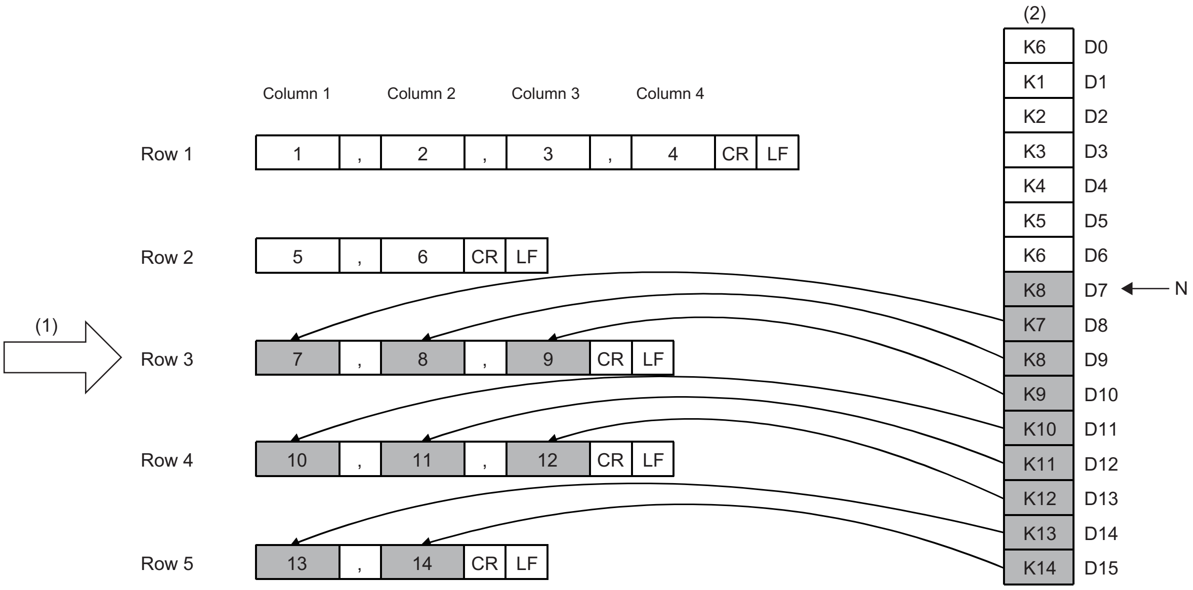
(1) Starting row

(2) Data in the device (write data)

N: Number of data

[Added to the end of the file]

- Execution/completion type: Writing data after converted to CSV format (Decimal (signed 16-bit data))
- Number of columns specification: 3H
- Data type specification: Words
- File position: FFFFFFFFH (added to the end of the file)
- Write start device: D7
- Number of request write data: 8H



(1) Starting position

(2) Data in the device (write data)

N: Number of data

Point

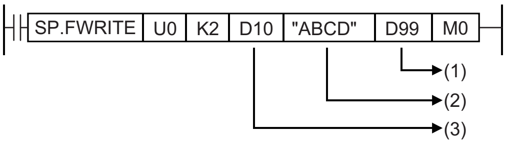
- An integral multiple of "Number of columns" should be specified for "Number of request write data". Otherwise, numbers of columns will be apart.
- The last data is always followed by a line feed code. In addition mode, therefore, data is added starting from the beginning of a new line.
- When data is added to the end of a file, columns are shifted if "Number of columns" is changed from the previous writing.

- The following figures show an example of specifying "String (ASCII data)" for the execution/completion type.

Ex.

Writing data after converted to CSV format (string (ASCII data))

[Data to be written to a file]

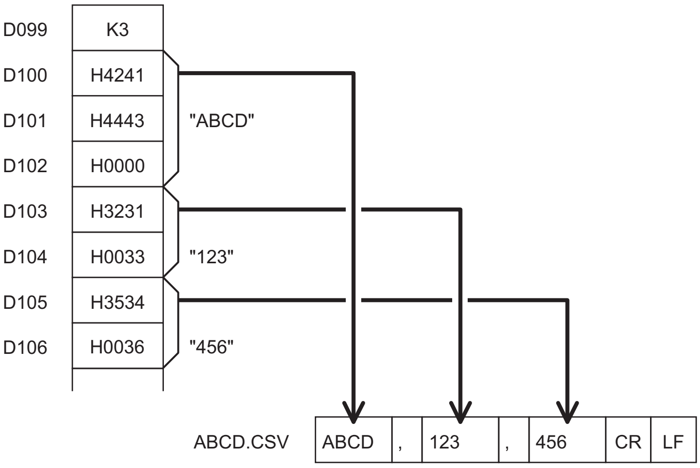


(1) Data to be written
(2) File name
(3) Control data

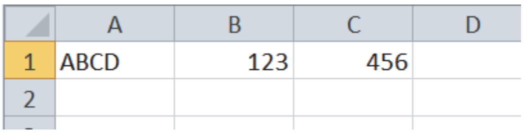
[Control data]

D10	H0130
D11	H0000
D12	-
D13	-
D14	K0
D15	
D16	K0
D17	K2

D10: Execution/completion type: String (ASCII data)
D11: Completion status
D12: Number of data actually written
D13: (Not used)
D14, D15: File position
D16: Number of columns
D17: Data type specification



D99: Number of request write data
D100 to D102: String to be written to the 1st column of the 1st line
D103, D104: String to be written to the 2nd column of the 1st line
D105, D106: String to be written to the 3rd column of the 1st line



	A	B	C	D
1	ABCD	123	456	
2				

Point

- Set 00H (NULL) in the end of the string in an element. When the number of bytes of the string is even, set 0000H (two bytes of NULL) in the next one word.
- The maximum number of characters in one element is 1999. If this maximum number is exceeded and 00H (NULL) is not stored, characters of 2000th character and after are not written and the write processing shifts to the next element.
- A maximum of 1023 elements can be written in a single instruction execution.

- The following table lists values to set in (s3)+1 and later and data to be written in a CSV file when "0140H: Floating point real number (single-precision real number)" or "0141H: Floating point real number (double-precision real number)" is set to (d1)+0.

Execution/completion type ((d1)+0)	Value to set in the write data ((s3)+1 and later)	Data to be written in a CSV file
0140H: Floating point real number (single-precision real number)	Values within the range of: $-2^{128} <$ data $\leq -2^{-126}$, 0, $2^{-126} \leq$data $< 2^{128}$	A value (0 to 7 digits in the decimal part) as given on the left is written in exponential format.
	Values other than above	0 is written. (Data cannot be converted.)
0141H: Floating point real number (double-precision real number)	Values within the range of: $-2^{1024} <$ data $\leq -2^{-1022}$, 0, $2^{-1022} \leq$data $< 2^{1024}$	A value (0 to 15 digits in the decimal part) as given on the left is written in exponential format.
	Values other than above	0 is written. (Data cannot be converted.)

- The following shows how the file size (total number of bytes) is calculated when a CSV format file is written to the SD memory card.

[Total number of bytes] = [Total number of bytes excluding the last row] + [Number of bytes of the last row]

([Number of bytes of each row] = [Number of columns[*1]] +1+ [total number of bytes of all data values per line][*2])

[*1] The specified number of columns applies to rows other than the last row. The number of columns of the last row is calculated as shown below because it may differ from the specified number of rows depending on the number of write data.

- The number of rows excluding the last row is calculated. (Number of rows excluding the last row = number of requested write data÷number of columns (remainders rounded down)
- The number of columns of the last row is calculated. (Number of columns of the last row = number of requested write data - (number of rows excluding the last row × number of columns))

[*2] The following shows how the number of bytes of each data value is calculated.

Sign of data value	Number of bytes of each data value	Range of bytes	Example
Positive	Number of digits	1 to 5 (word specification) 1 to 3 (byte specification)	• 12345: 5 bytes • 67: 2 bytes
Negative	Number of digits + 1	2 to 6 (word specification) 2 to 4 (byte specification)	• -12345: 6 bytes • -67: 3 bytes

Precautions

- Do not execute the SP.FWRITE instruction in interrupt programs. Doing so may cause malfunction of the module.

Operation error

Error code (SD0)	Description
2820H	The value in the device specified by (s3)+0 is out of the range (1 to 65535), or exceeds the setting area specified by (s3)+1 and later in the device/label memory.
3405H	The drive specified by (s1) is not the one for the SD memory card.
	Any value that is set in the device specified by (d1) and later as control data is out of the range. ☞ Page 617 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later) ☞ Page 619 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU
	The file name character string specified by (s2) cannot be read. • The number of characters of the string in the file name specified exceeds the range. • An inhibited value is set. ☞ Page 617 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later) ☞ Page 619 For RnCPU (firmware version "27" or earlier), RnENCPU (firmware version "27" or earlier), RnPCPU, and RnSFCPU
3427H	An invalid combination of (d1)+0 (Execution/completion type) and (d1)+7 (Data type specification) is specified. ☞ Page 617 For RnCPU (firmware version "28" or later) and RnENCPU (firmware version "28" or later)

When the SP.FWRITE instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

For the error code stored in (d1)+1, refer to the following.

☞ Page 593 Reading/Writing data in the specified file

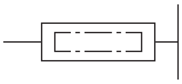
7.7 Debugging and Failure Diagnostic Instructions

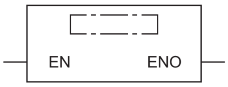
Resetting the error display and the annunciator display

LEDR

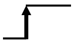
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the CPU module.

Ladder	ST
	ENO:=LEDR(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
LEDR	

Processing details

- This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the CPU module. Executing the instruction once resets both the error display and the annunciator display.
- When a self-diagnostic error has occurred, the CPU module operates as follows:
  - When a self-diagnostic error (continuation error) has occurred

The ERROR LED on the front of the CPU module remains off.

The values in SM0, SM1, and SD0 are not reset automatically at this time. Reset the values by the program.

- When a battery error has occurred

When the LEDR instruction is executed after battery replacement, the BATTERY LED on the front of the CPU module turns off. At this time, SM51 also turns off.

- When the annunciator (F) is on, the CPU module operates as follows:
  - The USER LED turns off.
  - The values in SD62, SD63, and SD64 to SD79 are all cleared to 0.

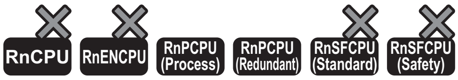
(1) All data are cleared.

Operation error

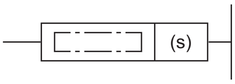
There is no operation error.

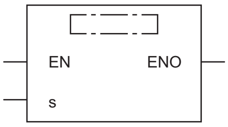
Generating a continuation error

PALERT(P)

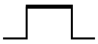
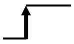


This instruction generates a continuation error in the CPU module.

Ladder	ST
	ENO:=PALERT(EN,s); ENO:=PALERTP(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
PALERT	
PALERTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be stored in the detailed information 2 of the error code 1810H or device number where the data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction generates a continuation error (error code: 1810H) in the CPU module. When the instruction is executed, SM0 turns on and the ERROR LED of the CPU module turns on. SM1 does not turn on.
- Data (16-bit signed binary) specified by (s) is stored in the detailed information 2 of the error code 1810H.

Point

- The PALERT(P) instruction is useful for debugging since a continuation error can be simulated at the start-up of the system. The instruction execution point can be identified by checking the detailed information 2 of the error code 1810H.

Operation error

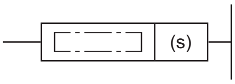
There is no operation error, except the error (error code: 1810H) which occurs by executing the PALERT(P) instruction.

Generating a stop error

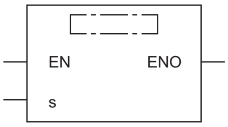
PABORT

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

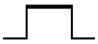
This instruction stops program execution and generates a stop error in the CPU module.

Ladder	ST
	ENO:=PABORT(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PABORT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be stored in the detailed information 2 of the error code 3070H or device number where the data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction stops program execution and generates a stop error (error code: 3070H) in the CPU module. When the instruction is executed, SM0 turns on and the ERROR LED of the CPU module flashes. SM1 does not turn on.
- Data (16-bit signed binary) specified by (s) is stored in the detailed information 2 of the error code 3070H.

Point

- The PABORT instruction is useful for debugging since a stop error can be simulated at the start-up of the system. The instruction execution point can be identified by checking the detailed information 2 of the error code 3070H.
- If any problem occurs in an external device connected to the standby system, the PABORT instruction prevents the systems to be switched by generating a stop error in the standby system.

Operation error

There is no operation error, except the error (error code: 3070H) which occurs by executing the PABORT instruction.

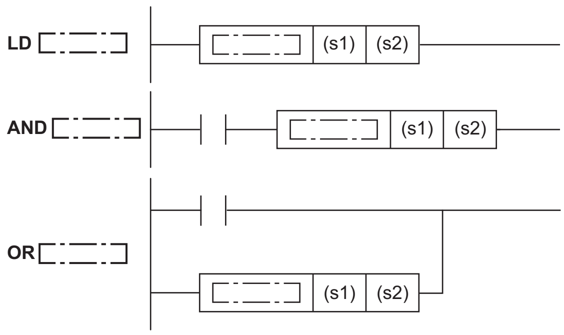
7.8 String Processing Instructions

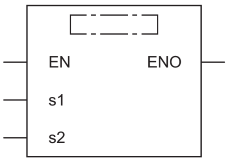
Comparing string data

LD$□, AND$□, OR$□

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions compare string data as normally open contacts.

Ladder	ST[1]
 (□ is replaced by $=, $<>, $>, $<=, $<, or $>=.)	ENO:=LDSTRING_□(EN,s1,s2); ENO:=ANDSTRING_□(EN,s1,s2); ENO:=ORSTRING_□(EN,s1,s2); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[2]

FBD/LD
 (□ is replaced by a combination of LDSTRING_, ANDSTRING_, or ORSTRING_ and EQ, NE, GT, LE, LT, or GE.)[2]

---

[1] The engineering tool with version "1.035M" or later supports the ST.

[2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LD$□, AND$□, OR$□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
(s2)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

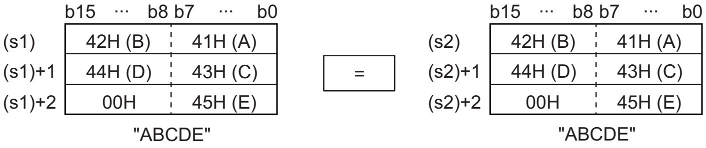
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

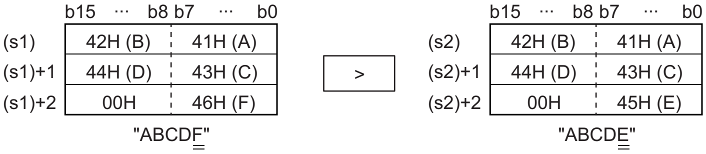
- These instructions perform a comparison operation between the character string data in the device specified by (s1) and the character string data in the device specified by (s2). (Devices are used as normally open contacts).
- In comparison operation, the ASCII code of character string is compared character by character from the beginning of the character string.
- The character strings in the devices specified by (s1) and (s2) mean those in the device numbers from the specified one to the one containing 00H.

- The comparison result turns out matching if all character strings match.



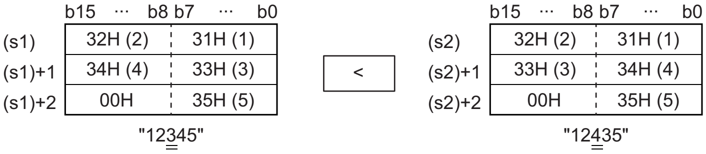
☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Continuity state (ENO is on.)
$<>, NE	Non-continuity state (ENO is off.)
$>, GT	Non-continuity state (ENO is off.)
$<=, LE	Continuity state (ENO is on.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

- When different character strings are compared, the character string with a larger character code is greater.



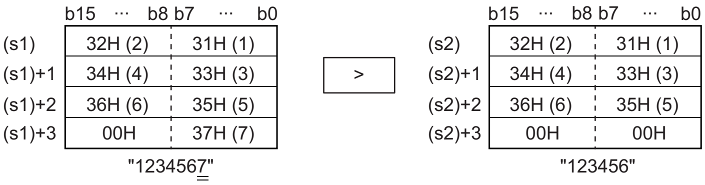
☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Continuity state (ENO is on.)
$<=, LE	Non-continuity state (ENO is off.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

- When different character strings are compared, the relative size of a character string is determined by the relative size of the first different character codes.



☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Non-continuity state (ENO is off.)
$<=, LE	Continuity state (ENO is on.)
$<, LT	Continuity state (ENO is on.)
$>=, GE	Non-continuity state (ENO is off.)

- When the lengths of the character string data in the devices specified by (s1) and (s2) are different, the longer character string data is greater.



□ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Continuity state (ENO is on.)
$<=, LE	Non-continuity state (ENO is off.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

- The character string in the device specified by (s1) or (s2) exceeds 16383 characters, the operation result will be non-continuity (ENO OFF).
- If the LDSTRING_□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORSTRING□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

There is no operation error.

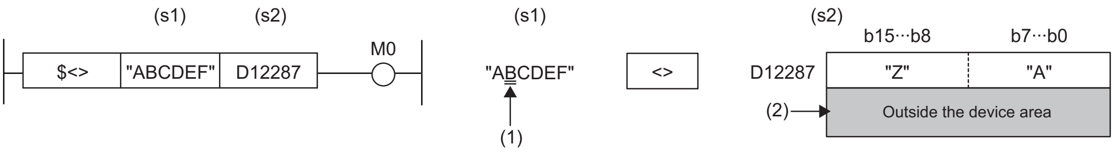
Point

The character string comparison instructions perform the following checks while comparing character strings.

- Checking whether the device area range is exceeded
- Checking whether the character string is within 16383 characters

If 00H does not exist in the device area or the character string exceeds 16383 characters and a character mismatch is detected, the instruction outputs comparison operation results without causing non-continuity (ENO OFF).

The following example shows the operation result when the last device number of the device area is D12287.



(1) The second character of (s1) differs from that of (s2) ((1)≠(s2)), and accordingly the operation result will be continuity (ENO OFF).

(2) D12287 and later are outside the device area, and accordingly character string data comparison is performed using data up to D12287.

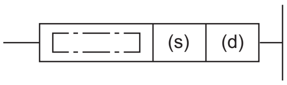
Since a character string mismatch has been detected, the condition is satisfied and processing ends.

Concatenating string data

$+(P) [when two operands are set]

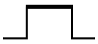
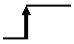
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions concatenate string data.

Ladder	ST
	Not supported (☞ Page 636 $+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 636 $+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
$+	
$+P	

Setting data

■Description, range, data type

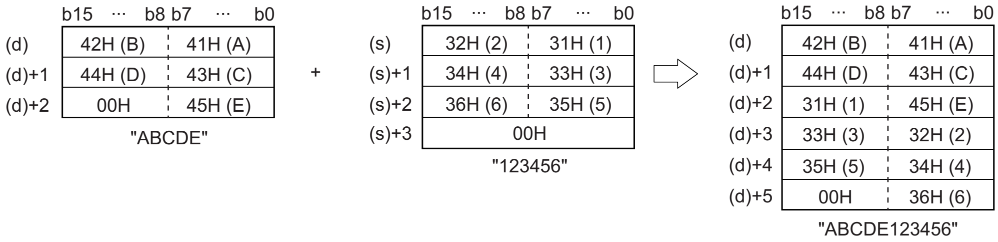
Operand	Description	Range	Data type	Data type (label)
(s)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(d)	Start device where the data to be concatenated is stored	—	String	ANYSTRING_SINGLE

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the character string stored in the device number specified by (s) and later to the end of the character string data stored in the device number specified by (d) and later, and stores the connected data in the device number specified by (d) and later.



- For concatenating character strings, the instruction ignores 00H that indicates the end of the character string in the device specified by (d) and appends the character string in the device specified by (s) following the last character in the device specified by (d).

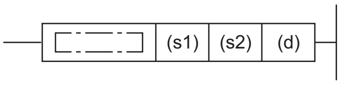
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) and later in the device/label memory.
	There is no NULL code (00H) in the setting area specified by (d) and later in the device/label memory.
2821H	The device numbers for storing the strings in the devices specified by (s) and (d) are overlapping.
3405H	The number of characters in the string specified by (s) exceeds 16383.
	The number of characters in the string specified by (d) exceeds 16383.
3406H	The number of characters in the concatenated string ((s)+(d)) exceeds 16383.
	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

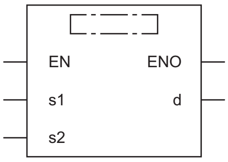
$+(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions concatenate string data.

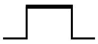
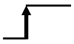
Ladder	ST
	ENO:=STRINGPLUS(EN,s1,s2,d); ENO:=STRINGPLUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by STRINGPLUS or STRINGPLUSP.)

■Execution condition

Instruction	Execution condition
$+	
$+P	

Setting data

■Descriptions, ranges, and data types

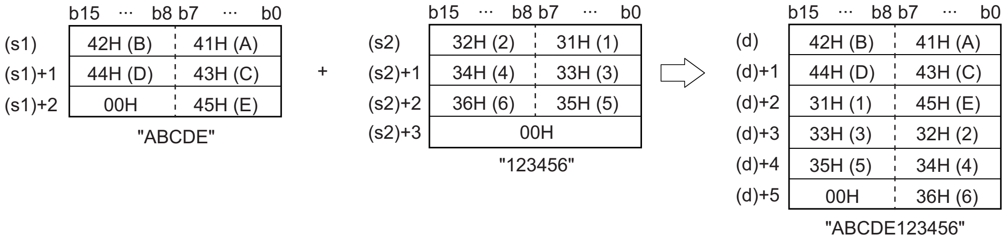
Operand	Description	Range	Data type	Data type (label)
(s1)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(s2)	Data to be concatenated or the start device containing the data to be concatenated	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the concatenated data	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the character string stored in the device number specified by (s2) and later to the end of the character string data stored in the device number specified by (s1) and later, and stores the connected data in the device number specified by (d) and later.



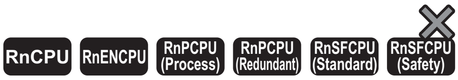
- For concatenating character strings, the instruction ignores 00H that indicates the end of the character string in the device specified by (s1) and appends the character string in the device specified by (s2) following the last character in the device specified by (s1).

Operation error

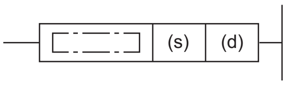
Error code (SD0)	Description
2820H	After the device number specified by (s1) and later, there is no 00H before the relevant device number.
	After the device number specified by (s2) and later, there is no 00H before the relevant device number.
2821H	The device numbers for storing the strings in the devices specified by (s2) and (d) are overlapping.
3405H	The number of characters in the string specified by (s1) exceeds 16383.
	The number of characters in the string specified by (s2) exceeds 16383.
3406H	The string stored in the device specified by (d) is out of the output enable range. • The number of characters in the concatenated string exceeds 16383. • The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Transferring string data

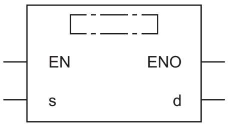
$MOV(P)



These instructions transfer string data to the specified device number and later.

Ladder	ST[*1]
	ENO:=STRINGMOV(EN,s,d); ENO:=STRINGMOVP(EN,s,d);

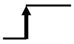
FBD/LD



(□ is replaced by STRINGMOV or STRINGMOVP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
$MOV	
$MOVP	

Setting data

■Descriptions, ranges, and data types

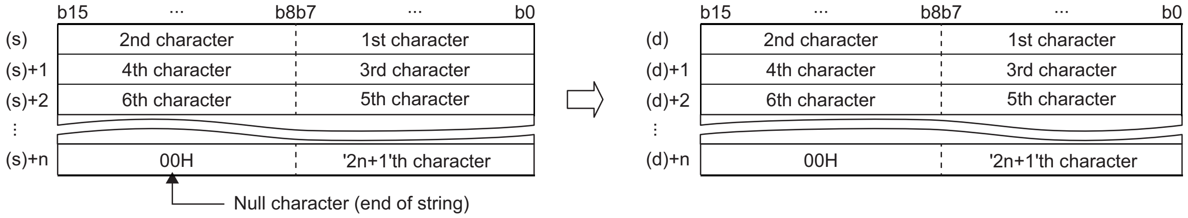
Operand	Description	Range	Data type	Data type (label)
(s)	Character string to be transferred (maximum of 255 characters) or the start device containing such character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the transferred character string	—	String	ANYSTRING_SINGLE
EN	Execution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

■Applicable devices

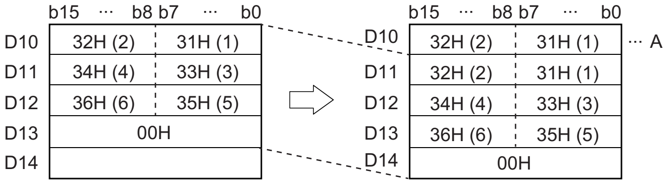
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the character string data in the device specified by (s) to the device number specified by (d) and later. The character strings from the one enclosed by double quotes (") or the device number specified by (s) to the device number containing 00H are transferred all at once.

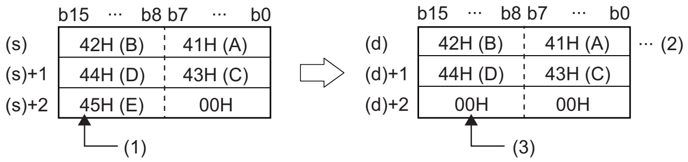


- Even when the device range ((s) to (s)+n) in which the character string data to be transferred and the device range ((d) to (d)+n) for storing the transferred data are overlapping, the processing is performed normally. For example, the character strings stored in the devices specified by D10 to D13 are transferred to the devices specified by D11 to D14 as shown below.



A: Data remain the same.

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



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

(2) Data remain the same.

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

Operation error

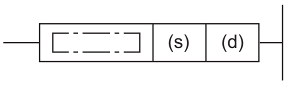
Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) and later in the device/label memory.
3405H	The number of characters in the string specified by (s) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Transferring Unicode string data

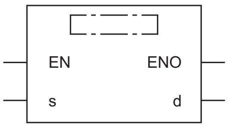
$MOV(P)_WS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions transfer Unicode string data to the specified device number and later.

Ladder	ST[*1]
	ENO:=STRINGMOV_WS(EN,s,d); ENO:=STRINGMOVP_WS(EN,s,d);

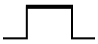
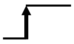
FBD/LD



(□ is replaced by STRINGMOV_WS or STRINGMOVP_WS.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
$MOV_WS	
$MOVP_WS	

Setting data

■Descriptions, ranges, and data types

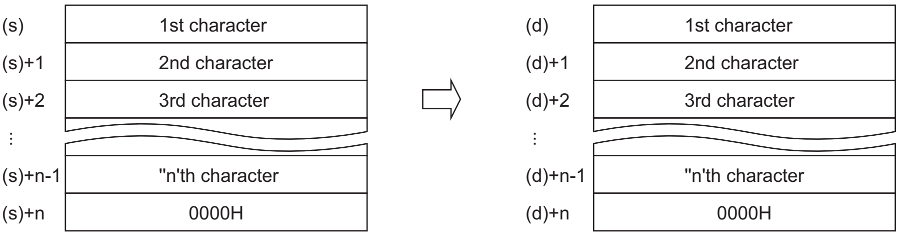
Operand	Description	Range	Data type	Data type (label)
(s)	Unicode character string to be transferred (maximum of 255 characters) or the start device containing the Unicode character string	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the transferred Unicode character string	—	Unicode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

■Applicable devices

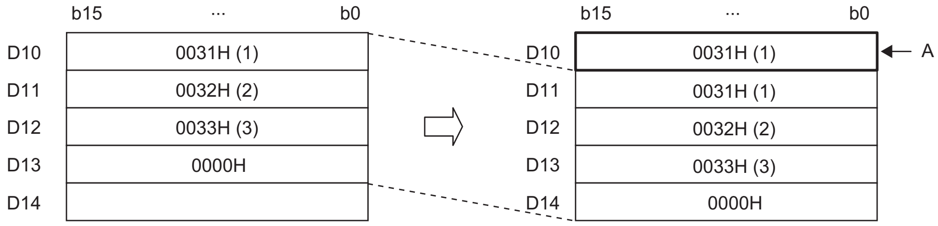
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the Unicode character string data in the device specified by (s) to the device number specified by (d) and later. The Unicode character strings from the one enclosed by double quotes (") or the device number specified by (s) to the device number containing 0000H are transferred all at once.



- Even when the device range ((s) to (s)+n)) in which the Unicode character string data to be transferred and the device range ((d) to (d)+n) for storing the transferred data are overlapping, the processing is performed normally. For example, the character strings stored in the devices specified by D10 to D13 are transferred to the devices specified by D11 to D14 as shown below.



A: Same as before transfer

Operation error

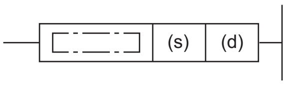
Error code (SD0)	Description
2820H	There is no 0000H in the setting area specified by (s) and later in the device/label memory.
3405H	The number of characters in the Unicode string specified by (s) exceeds 16383.
3406H	The entire Unicode string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Converting 16-bit binary data to decimal ASCII

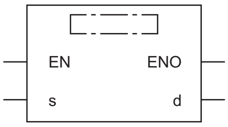
BINDA(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

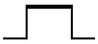
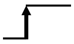
These instructions convert 16-bit binary data to the decimal ASCII code.

Ladder	ST	
	ENO:=BINDA(EN,s,d); ENO:=BINDAP(EN,s,d)	ENO:=BINDA_U(EN,s,d); ENO:=BINDAP_U(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
BINDA BINDA_U	
BINDAP BINDAP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	BINDA(P)	Binary data used for ASCII conversion	-32768 to 32767	16-bit signed binary	ANY16_S
	BINDA(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in decimal notation of the 16-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (a sign + 5 digits)	Page 644 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 644 Operation of when SM705 (Number of conversion digits selection) is on

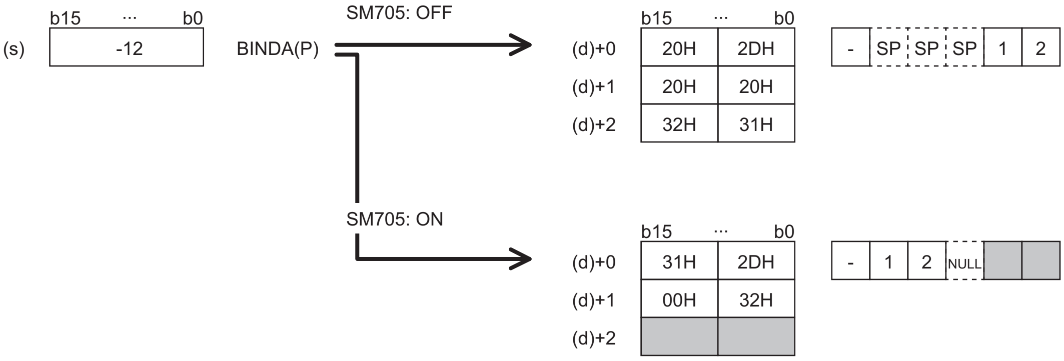
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

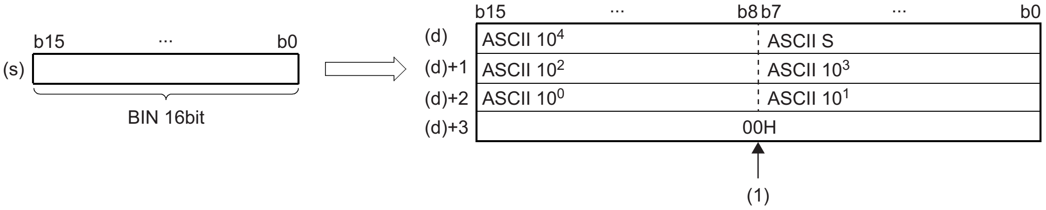
When the BINDA(P) instruction is executed with the numeric value "-12" stored in (s)



- When SM705 is off, the number of digits is fixed. The first character is a sign and it is 2DH(-) in the above example. (If (s) is 0 or positive, the first character is 20H (space).) The numeric part is right-justified. When the length of the numeric part is less than 5 digits, the ASCII code 20H (space) is stored for the ASCII code of the upper digit(s).
- When SM705 is on, data is left-justified. When the length of the numeric part is less than 5 digits, 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits in (d) to (d)+2.



ASCII S: Sign data of ASCII code*1

ASCII 10^4: Ten-thousands place of ASCII code*2

ASCII 10^3: Thousands place of ASCII code*3

ASCII 10^2: Hundreds place of ASCII code*4

ASCII 10^1: Tens place of ASCII code*5

ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+3 when SM701 (Number of output characters selection) is off. When it is on, the value in (d)+3 remains unchanged.

*1 When the value is 0 or positive, 20H (space) is stored. When the value is negative, 2DH (-) is stored.

*2 When the length of the numeric part is 4 digits or less, 20H (space) is stored in ASCII 10^4.

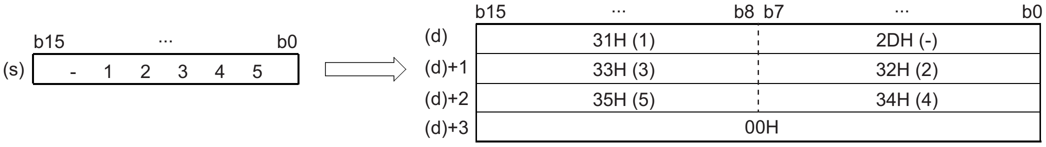
*3 When the length of the numeric part is 3 digits or less, 20H (space) is stored in ASCII 10^3.

*4 When the length of the numeric part is 2 digits or less, 20H (space) is stored in ASCII 10^2.

*5 When the length of the numeric part is 1 digit, 20H (space) is stored in ASCII 10^1.

Ex.

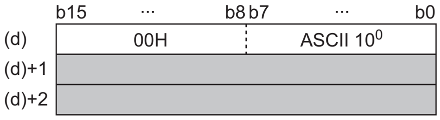
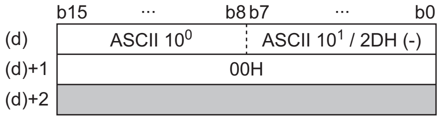
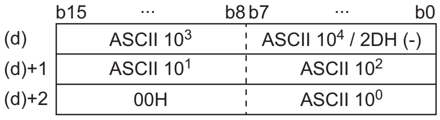
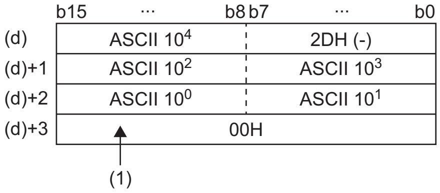
-12345 is set in (s) when the BINDA(P) instruction is used.



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+2	Value of (s)	Data of (d) to (d)+2
• 0 • Positive value (1 digit in numeric part)	• The upper byte of (d) is filled with 00H. • Data in (d)+1 and (d)+2 remains unchanged. 	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• (d)+1 is filled with 00H. • Data in (d)+2 remains unchanged. 
⋮			
• Positive value (5 digits in numeric part) • Negative value (4 digits in numeric part)	• The upper byte of (d)+2 is filled with 00H. 	• Negative value (5 digits in numeric part)	• (1): (d)+3 is filled with 00H only when SM701 (Number of output characters selection) is off. 

ASCII 10^4: Ten-thousands place of ASCII code

ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

ASCII 10^0: Ones place of ASCII code

- When the number of operation digits is less than the maximum number of digits (sign + 5 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of operation digits is equal to the maximum number of digits (a sign + 5 digits), 00H is stored in (d)+3 when SM701 is off. (d)+3 remains unchanged if SM701 is on.

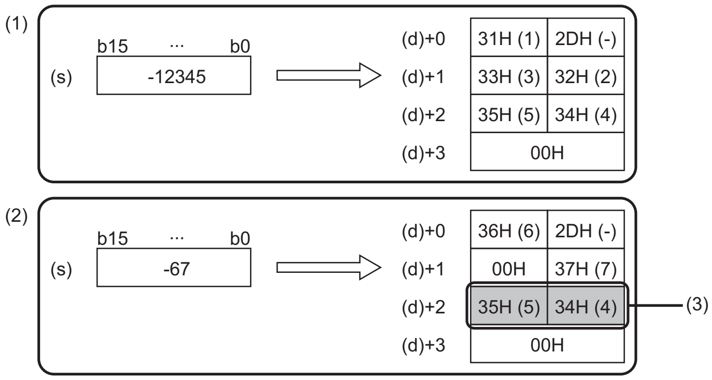
Operation error

There is no operation error.

Precautions

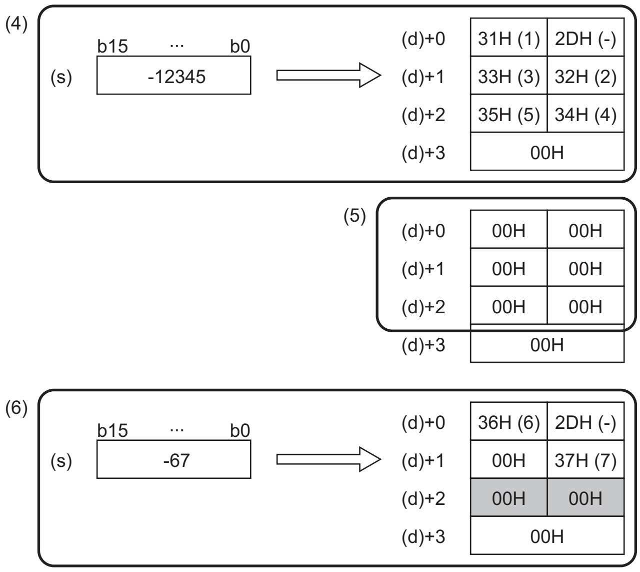
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BINDA(P) instruction when (s) is "-12345" and then executing another BINDA(P) instruction when (s) is "-67"



(1) "-12345" is converted into a string.
(2) "-67" is converted into a string.
(3) A part of the previous conversion result remains in (d)+2.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+2 before executing the BINDA(P)(_U) instruction.



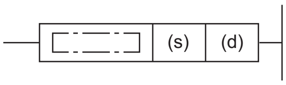
(4) "-12345" is converted into a string.
(5) (d)+0 to (d)+2 are cleared.
(6) "-67" is converted into a string.

Converting 32-bit binary data to decimal ASCII

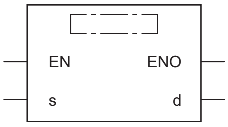
DBINDA(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

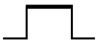
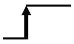
These instructions convert 32-bit binary data to the decimal ASCII code.

Ladder	ST	
	ENO:=DBINDA(EN,s,d); ENO:=DBINDAP(EN,s,d);	ENO:=DBINDA_U(EN,s,d); ENO:=DBINDAP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBINDA DBINDA_U	
DBINDAP DBINDAP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	DBINDA(P)	Binary data used for ASCII conversion	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBINDA(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)		Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in decimal notation of the 32-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (a sign + 10 digits)	Page 648 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 649 Operation of when SM705 (Number of conversion digits selection) is on

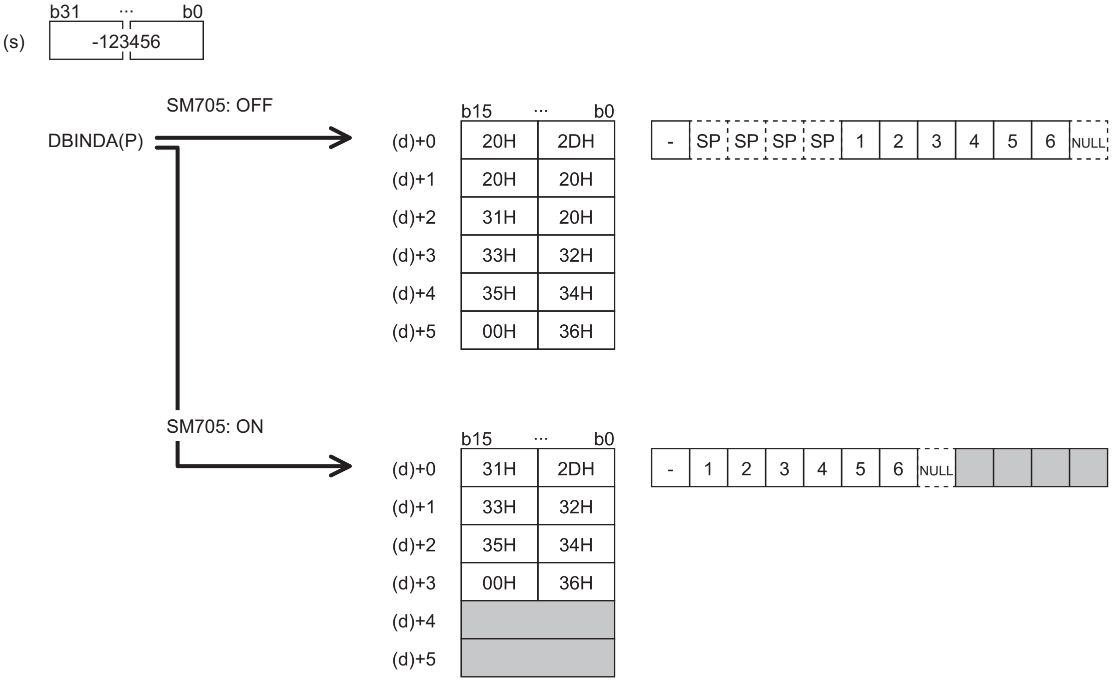
[1] For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

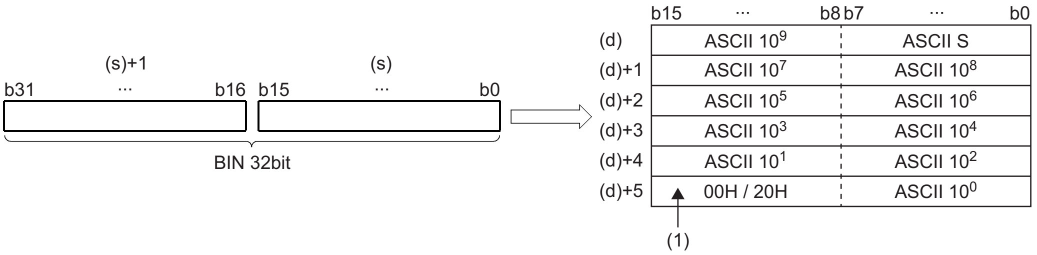
When the DBINDA(P) instruction is executed with a numeric value "-123456" stored in (s)



- When SM705 is off, the number of digits is fixed. The first character is a sign and it is 2DH(-) in the above example. (If (s) is 0 or positive, the first character is 20H (space).) The numeric part is right-justified. When the length of the numeric part is less than 10 digits, the ASCII code 20H (space) is stored for the ASCII code of the upper digit(s).
- When SM705 is on, data is left-justified. When the length of the numeric part is less than 10 digits, 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits in (d) to (d)+5.



	b15 ··· b8	b7 ··· b0
(d)	ASCII 10^9	ASCII S
(d)+1	ASCII 10^7	ASCII 10^8
(d)+2	ASCII 10^5	ASCII 10^6
(d)+3	ASCII 10^3	ASCII 10^4
(d)+4	ASCII 10^1	ASCII 10^2
(d)+5	00H / 20H	ASCII 10^0

ASCII S: Sign data of ASCII code*1

ASCII 10^9: Billions place of ASCII code*2

ASCII 10^8: Hundred-millions place of ASCII code*3

⋮

ASCII 10^1: Tens place of ASCII code*4

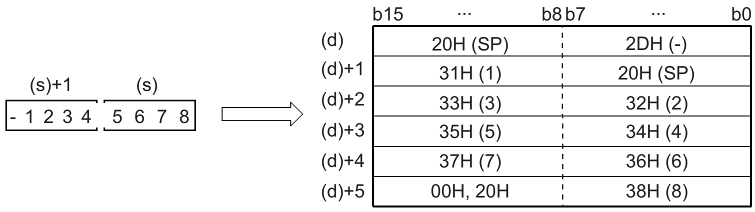
ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in the upper byte of (d)+5 only when SM701 (Number of output characters selection) is off, and 20H (space) is stored when it is on.

*1 When the value is 0 or positive, 20H (space) is stored. When the value is negative, 2DH (-) is stored.
*2 When the length of the numeric part is 9 digits or less, 20H (space) is stored in ASCII 10^9.
*3 When the length of the numeric part is 8 digits or less, 20H (space) is stored in ASCII 10^8.
*4 When the length of the numeric part is 1 digit, 20H (space) is stored in ASCII 10^1.

Ex.

When -12345678 (signed) is specified in (s)



	b15 ··· b8	b7 ··· b0
(d)	20H (SP)	2DH (-)
(d)+1	31H (1)	20H (SP)
(d)+2	33H (3)	32H (2)
(d)+3	35H (5)	34H (4)
(d)+4	37H (7)	36H (6)
(d)+5	00H, 20H	38H (8)

■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

<table>
<tr><th>Value of (s)</th><th>Data of (d) to (d)+2</th><th>Value of (s)</th><th>Data of (d) to (d)+2</th></tr>
<tr>
<td>• 0
• Positive value (1 digit in numeric part)</td>
<td>• The upper byte of (d) is filled with 00H.
• (d)+1 and later remain unchanged.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(d)</td><td>00H</td><td>ASCII 10^{0}</td></tr>
<tr><td>(d)+1</td><td colspan="2">▨</td></tr>
<tr><td>(d)+2</td><td colspan="2">▨</td></tr>
<tr><td>(d)+3</td><td colspan="2">▨</td></tr>
<tr><td>(d)+4</td><td colspan="2">▨</td></tr>
<tr><td>(d)+5</td><td colspan="2">▨</td></tr>
</table></td>
<td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td>
<td>• (d)+1 is filled with 00H.
• (d)+2 and later remain unchanged.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(d)</td><td>ASCII 10^{0}</td><td>ASCII 10^{1} / 2DH (-)</td></tr>
<tr><td>(d)+1</td><td colspan="2">00H</td></tr>
<tr><td>(d)+2</td><td colspan="2">▨</td></tr>
<tr><td>(d)+3</td><td colspan="2">▨</td></tr>
<tr><td>(d)+4</td><td colspan="2">▨</td></tr>
<tr><td>(d)+5</td><td colspan="2">▨</td></tr>
</table></td>
</tr>
<tr><td colspan="4">⋮</td></tr>
<tr>
<td>• Positive value (9 digits in numeric part)
• Negative value (8 digits in numeric part)</td>
<td>• The upper byte of (d)+4 is filled with 00H.
• (d)+5 and later remain unchanged.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(d)</td><td>ASCII 10^{7}</td><td>ASCII 10^{8} / 2DH (-)</td></tr>
<tr><td>(d)+1</td><td>ASCII 10^{5}</td><td>ASCII 10^{6}</td></tr>
<tr><td>(d)+2</td><td>ASCII 10^{3}</td><td>ASCII 10^{4}</td></tr>
<tr><td>(d)+3</td><td>ASCII 10^{1}</td><td>ASCII 10^{2}</td></tr>
<tr><td>(d)+4</td><td>00H</td><td>ASCII 10^{0}</td></tr>
<tr><td>(d)+5</td><td colspan="2">▨</td></tr>
</table></td>
<td>• Positive value (10 digits in numeric part)
• Negative value (9 digits in numeric part)</td>
<td>• (d)+5 is filled with 00H.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(d)</td><td>ASCII 10^{8}</td><td>ASCII 10^{9} / 2DH (-)</td></tr>
<tr><td>(d)+1</td><td>ASCII 10^{6}</td><td>ASCII 10^{7}</td></tr>
<tr><td>(d)+2</td><td>ASCII 10^{4}</td><td>ASCII 10^{5}</td></tr>
<tr><td>(d)+3</td><td>ASCII 10^{2}</td><td>ASCII 10^{3}</td></tr>
<tr><td>(d)+4</td><td>ASCII 10^{0}</td><td>ASCII 10^{1}</td></tr>
<tr><td>(d)+5</td><td colspan="2">00H</td></tr>
</table></td>
</tr>
<tr>
<td>• Negative value (10 digits in numeric part)</td>
<td>• 00H is stored in the upper byte (1) of (d)+5 when SM701 (Number of output characters selection) is off, and 20H (space) is stored when it is on.
<table>
<tr><th></th><th>b15 ··· b8</th><th>b7 ··· b0</th></tr>
<tr><td>(d)</td><td>ASCII 10^{9}</td><td>2DH (-)</td></tr>
<tr><td>(d)+1</td><td>ASCII 10^{7}</td><td>ASCII 10^{8}</td></tr>
<tr><td>(d)+2</td><td>ASCII 10^{5}</td><td>ASCII 10^{6}</td></tr>
<tr><td>(d)+3</td><td>ASCII 10^{3}</td><td>ASCII 10^{4}</td></tr>
<tr><td>(d)+4</td><td>ASCII 10^{1}</td><td>ASCII 10^{2}</td></tr>
<tr><td>(d)+5</td><td>↑ 00H / 20H</td><td>ASCII 10^{0}</td></tr>
</table>
(1)</td>
<td colspan="2">ASCII 10^{9}: Billions place of ASCII code
ASCII 10^{8}: Hundred-millions place of ASCII code
⋮
ASCII 10^{1}: Tens place of ASCII code
ASCII 10^{0}: Ones place of ASCII code</td>
</tr>
</table>

- When the number of operation digits is less than the maximum number of digits (a sign + 10 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of operation digits is equal to the maximum number of digits (a sign + 10 digits), 00H is stored in the upper byte of (d)+5 if SM701 is off. 20H (space) is stored in the upper byte of (d)+5 if SM701 is on.

Operation error

There is no operation error.

Precautions

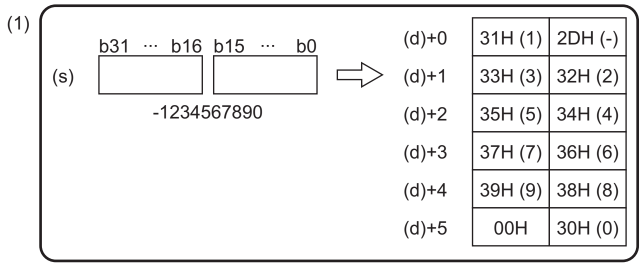
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the DBINDA(P) instruction when (s) is "-1234567890" and then executing another DBINDA(P) instruction when (s) is "-987654"

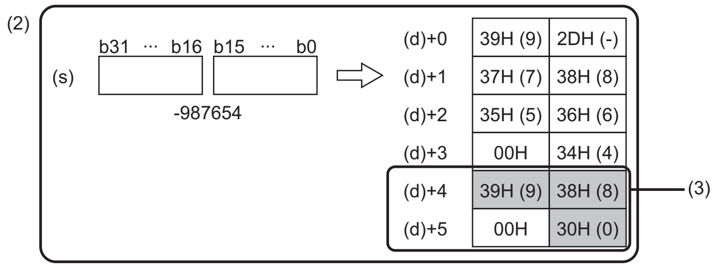
(1) "-1234567890" is converted into a string.

(2) "-987654" is converted into a string.

(3) A part of the previous conversion result remains in (d)+4 and (d)+5.



(d)+0	31H (1)	2DH (-)
(d)+1	33H (3)	32H (2)
(d)+2	35H (5)	34H (4)
(d)+3	37H (7)	36H (6)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)



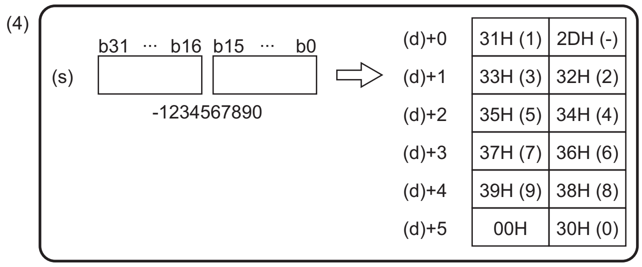
(d)+0	39H (9)	2DH (-)
(d)+1	37H (7)	38H (8)
(d)+2	35H (5)	36H (6)
(d)+3	00H	34H (4)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+5 before executing the DBINDA(P)(_U) instruction.

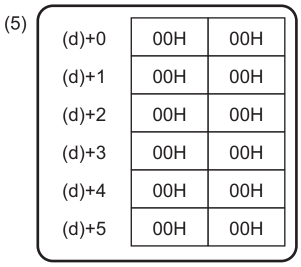
(4) "-1234567890" is converted into a string.

(5) (d)+0 to (d)+5 are cleared.

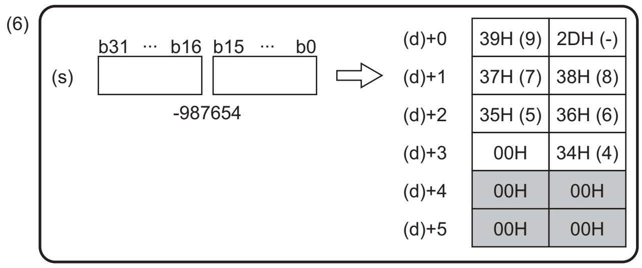
(6) "-987654" is converted into a string.



(d)+0	31H (1)	2DH (-)
(d)+1	33H (3)	32H (2)
(d)+2	35H (5)	34H (4)
(d)+3	37H (7)	36H (6)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)



(d)+0	00H	00H
(d)+1	00H	00H
(d)+2	00H	00H
(d)+3	00H	00H
(d)+4	00H	00H
(d)+5	00H	00H



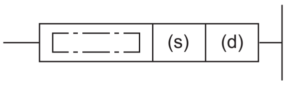
(d)+0	39H (9)	2DH (-)
(d)+1	37H (7)	38H (8)
(d)+2	35H (5)	36H (6)
(d)+3	00H	34H (4)
(d)+4	00H	00H
(d)+5	00H	00H

Converting 16-bit binary data to hexadecimal ASCII

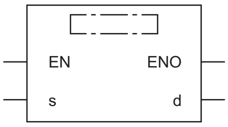
BINHA(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✕

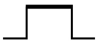
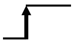
These instructions convert 16-bit binary data to the hexadecimal ASCII code.

Ladder	ST
 (s) (d)	ENO:=BINHA(EN,s,d); ENO:=BINHAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BINHA	
BINHAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data used for ASCII conversion	-32768 to 32767	16-bit signed binary	ANY16
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in hexadecimal notation of the 16-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the hexadecimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (4 digits)	Page 653 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 653 Operation of when SM705 (Number of conversion digits selection) is on

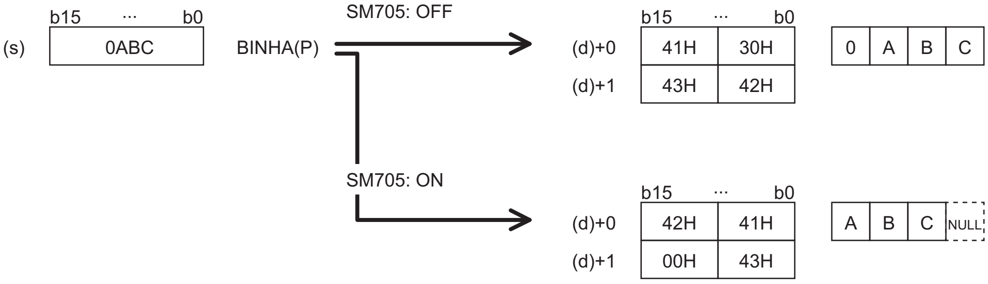
[1] For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

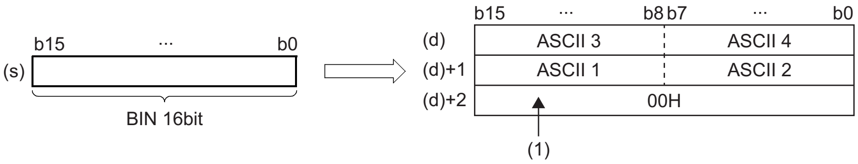
When the BINHA(P) instruction is executed with the 16-bit binary data "0ABCH" stored in (s)



- When SM705 is off, the number of digits is fixed. Four digits of "0ABC" are converted into ASCII data and stored.
- When SM705 is on, data is left-justified. "0ABC" with the leading "0" omitted ("ABC") are converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Hexadecimal ASCII data is stored in a fixed number of digits (4 digits) in (d).



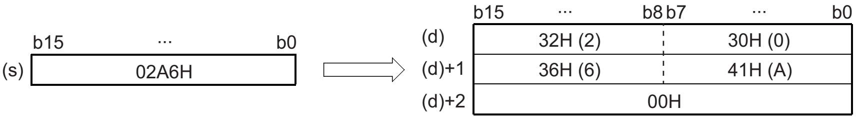
ASCII □: ASCII code (□th digit)

(1): 00H is stored in (d)+2 when SM701 (Number of output characters selection) is off. When it is on, the value in (d)+3 remains unchanged.

- The operation result to be stored in the device specified by (d) is processed as a 4-digit hexadecimal number. Therefore, 0 at the left side of the effective number of digits is processed as "0". (Zero padding)

Ex.

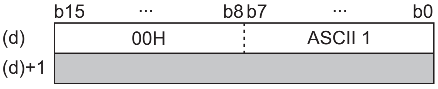
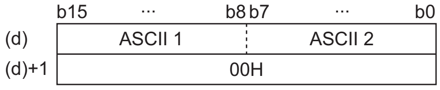
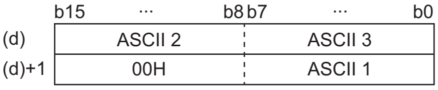
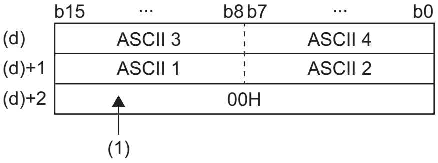
When 02A6H is stored in the device specified by (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Hexadecimal ASCII data for the number of digits (up to 4 digits) without the leftmost "0" in the effective digits is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+1	Value of (s)	Data of (d) to (d)+1
• 0H to FH	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to FFH	• (d)+1 is filled with 00H. 
• 100H to FFFH	• The upper byte of (d)+1 are filled with 00H. 	• 1000H to FFFFH	• (1): (d)+2 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+2 remains unchanged if SM701 is on. 

ASCII □: ASCII code (□th digit)

- When the number of digits is less than the maximum number of digits (4 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (4 digits), 00H is stored in (d)+2 when SM701 is off. (d)+2 remains unchanged if SM701 is on.

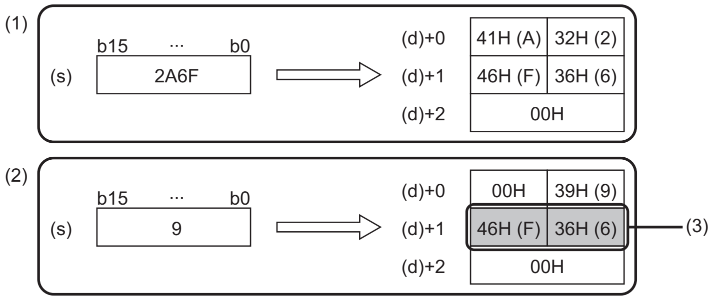
Operation error

There is no operation error.

Precautions

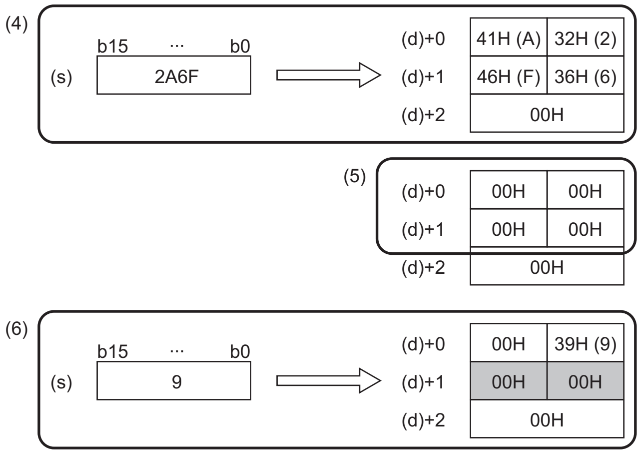
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BINHA(P) instruction when (s) is "2A6F" and then executing another BINHA(P) instruction when (s) is "9"



(1) "2A6F" is converted into a string.
(2) "9" is converted into a string.
(3) A part of the previous conversion result remains in (d)+1.

To avoid this, create a program to clear the entire data storage areas (d)+0 and (d)+1 before executing the BINHA(P) instruction.



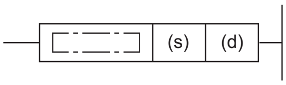
(4) "2A6F" is converted into a string.
(5) (d)+0 to (d)+1 are cleared.
(6) "9" is converted into a string.

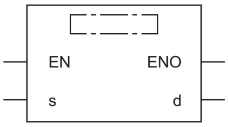
Converting 32-bit binary data to hexadecimal ASCII

DBINHA(P)

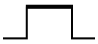
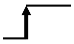
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions convert 32-bit binary data to the hexadecimal ASCII code.

Ladder	ST
	ENO:=DBINHA(EN,s,d); ENO:=DBINHAP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DBINHA	
DBINHAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data used for ASCII conversion	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit at each place in hexadecimal notation of the 32-bit binary data specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (8 digits)	Page 657 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 657 Operation of when SM705 (Number of conversion digits selection) is on

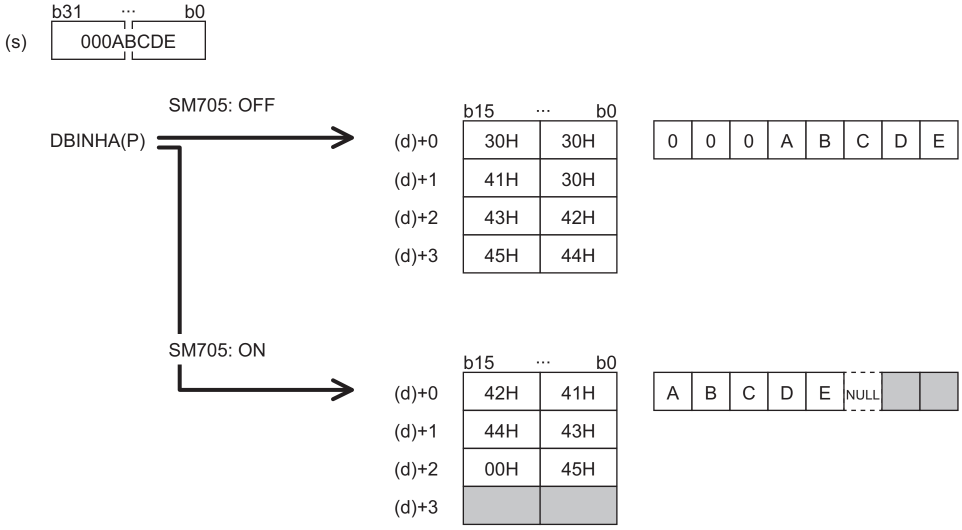
[1] For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

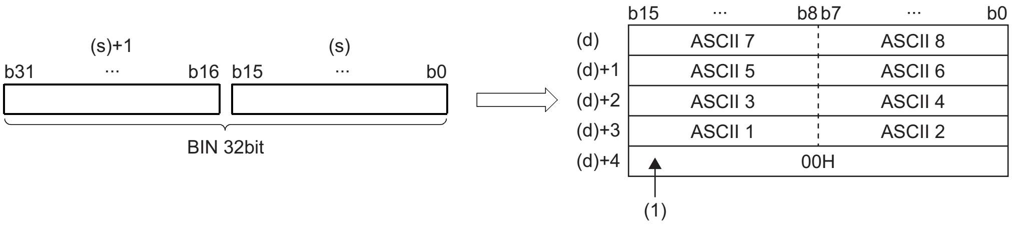
When the DBINHA(P) instruction is executed with a 32-bit binary data "000ABCDEH" stored in (s)



- When SM705 is off, the number of digits is fixed. Eight digits of "000ABCDE" are converted into ASCII data and stored.
- When SM705 is on, data is left-justified. The effective digits "000ABCDE" with the leading "0"s omitted ("ABCDE") are converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Hexadecimal ASCII data is stored in a fixed number of digits (8 digits) in (d) to (d)+3.



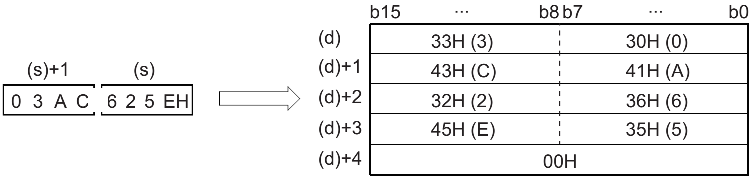
ASCII □: ASCII code (□th digit)

(1): 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. When it is on, the value in (d)+3 remains unchanged.

- The operation result to be stored in the device specified by (d) is processed as a 8-digit hexadecimal number. Therefore, 0 at the left side of the effective number of digits is processed as "0". (Zero padding)

Ex.

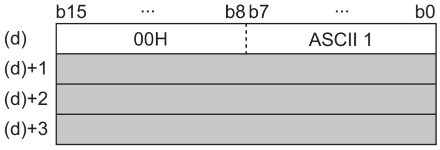
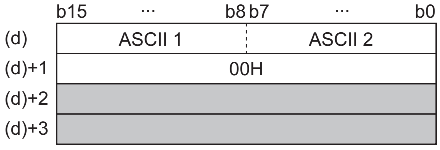
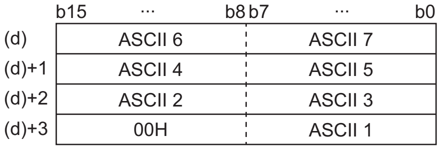
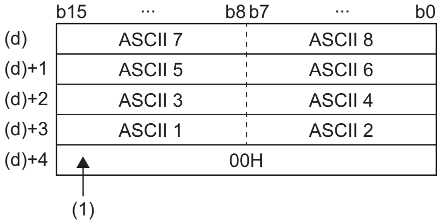
When 03AC625EH is stored in the device specified by (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Hexadecimal ASCII data for the number of digits (up to 8 digits) without the leftmost "0" in the effective digits is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+3	Value of (s)	Data of (d) to (d)+3
• 0H to FH	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to FFH	• (d)+1 is filled with 00H. • (d)+2 and later remain unchanged. 
⋮			
• 1000000H to FFFFFFFH	• The upper byte of (d)+3 is filled with 00H. 	• 10000000H to FFFFFFFFH	• (1): (d)+4 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on. 

ASCII □: ASCII code (□th digit)

- When the number of digits is less than the maximum number of digits (8 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (8 digits), 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on.

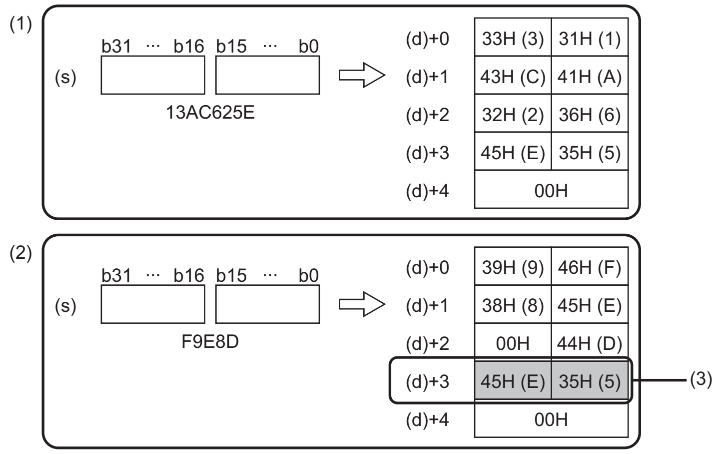
Operation error

There is no operation error.

Precautions

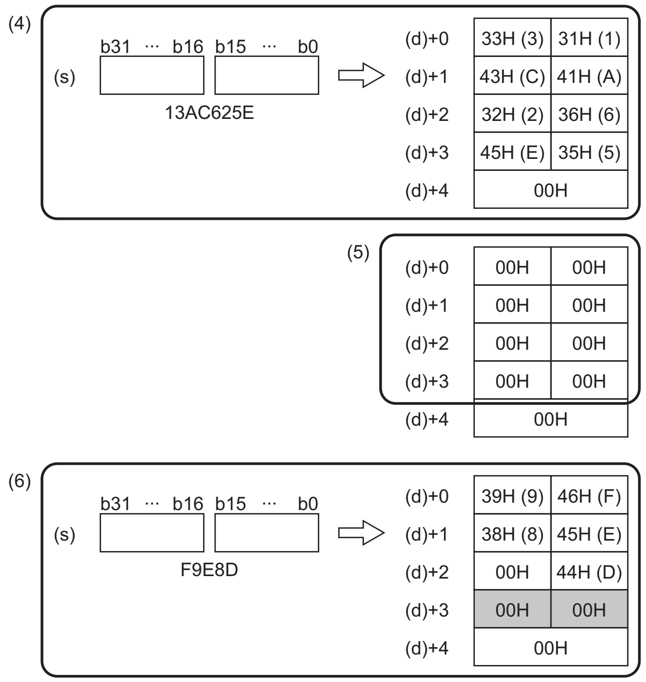
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the DBINHA(P) instruction when (s) is "13AC625E" and then executing another DBINHA(P) instruction when (s) is "F9E8D"



(1) "13AC625E" is converted into a string.
(2) "F9E8D" is converted into a string.
(3) A part of the previous conversion result remains in (d)+3.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+3 before executing the DBINHA(P) instruction.



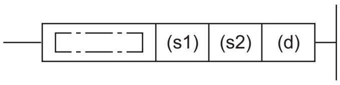
(4) "13AC625E" is converted into a string.
(5) (d)+0 to (d)+3 are cleared.
(6) "F9E8D" is converted into a string.

Converting 16-bit binary data to string data

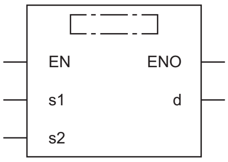
STR(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

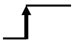
These instructions convert 16-bit binary data to a string by adding a decimal point to the specified place of the data.

Ladder	ST	
	ENO:=STR(EN,s1,s2,d); ENO:=STRP(EN,s1,s2,d);	ENO:=STR_U(EN,s1,s2,d); ENO:=STRP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STR STR_U	
STRP STRP_U	

Setting data

■Description, range, data type

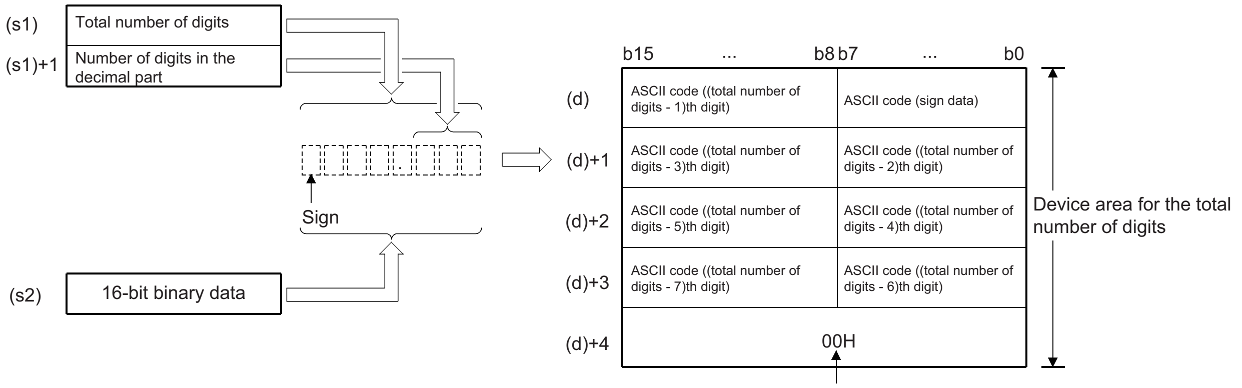
Operand		Description	Range	Data type	Data type (label)
(s1)	STR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	STR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	STR(P)	Conversion target data	-32768 to 32767	16-bit signed binary	ANY16_S
	STR(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

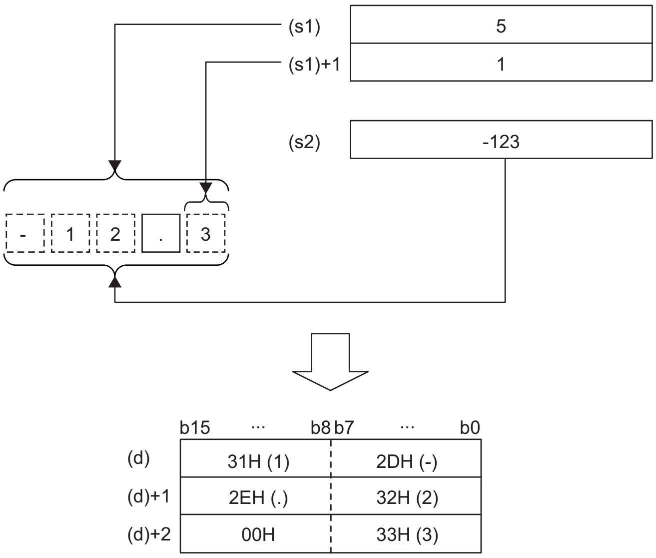
- These instructions add a decimal point to the 16-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.



A null character, 00H, is automatically stored at the end of string.

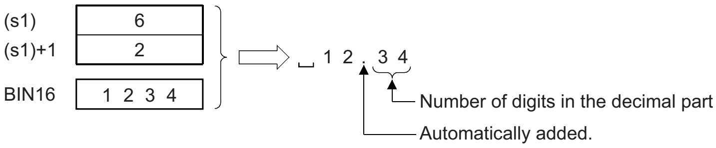
Ex.

When converting data "-123" in (s2) into a string assuming that the number of decimal places is one ("-12.3")

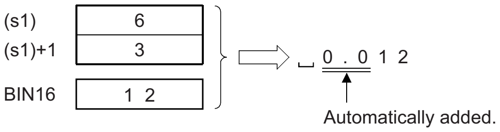


- The total number of digits that can be specified by (s1) is 2 to 8.
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 5. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.

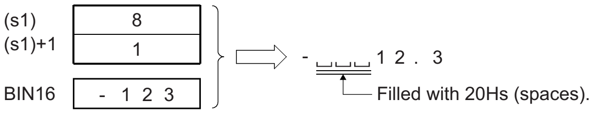
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the 16-bit binary data is positive, and "2DH" (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.



  - If the specified number of digits in the decimal part is greater than the number of digits of the 16-bit binary data, 0s are automatically added and the data is regarded as "0.□□□□".



  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 16-bit binary data, "20H" (space) is stored between the sign and the numeric value. If the number of digits of the 16-bit binary data is greater, an error occurs.



  - The value "00H" is automatically stored at the end of the converted character string.

Operation error

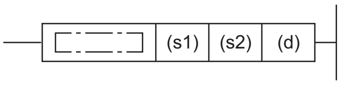
Error code (SD0)	Description
3401H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 8. • The specified number of digits in the decimal part is out of the range, 0 to 5. • The relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following. (Total number of digits)-3≥Number of digits in the decimal part • The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 16-bit binary data specified by (s2). [Number of digits in (s1)] < [Number of digits of 16-bit binary data excluding the sign in (s2) + Number of digits in the sign (+ or -) + Decimal point (.)]

Converting 32-bit binary data to string data

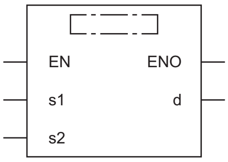
DSTR(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

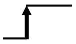
These instructions convert 32-bit binary data to a string by adding a decimal point to the specified place of the data.

Ladder	ST	
	ENO:=DSTR(EN,s1,s2,d); ENO:=DSTRP(EN,s1,s2,d);	ENO:=DSTR_U(EN,s1,s2,d); ENO:=DSTRP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSTR DSTR_U	
DSTRP DSTRP_U	

Setting data

■Description, range, data type

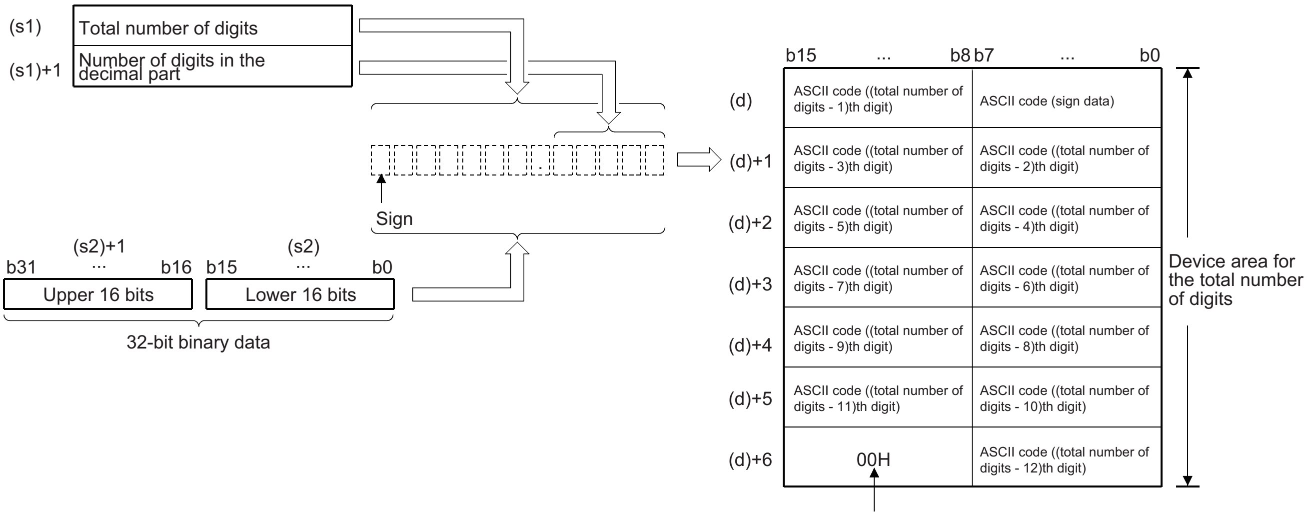
Operand		Description	Range	Data type	Data type (label)
(s1)	DSTR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	DSTR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	DSTR(P)	Conversion target data	-2147483648 to 2147483647	16-bit signed binary	ANY32_S
	DSTR(P)_U		0 to 4294967295	16-bit unsigned binary	ANY32_U
(d)		Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions add a decimal point to the 32-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to character string data, and store the converted data in the device areas specified by (d) and later.

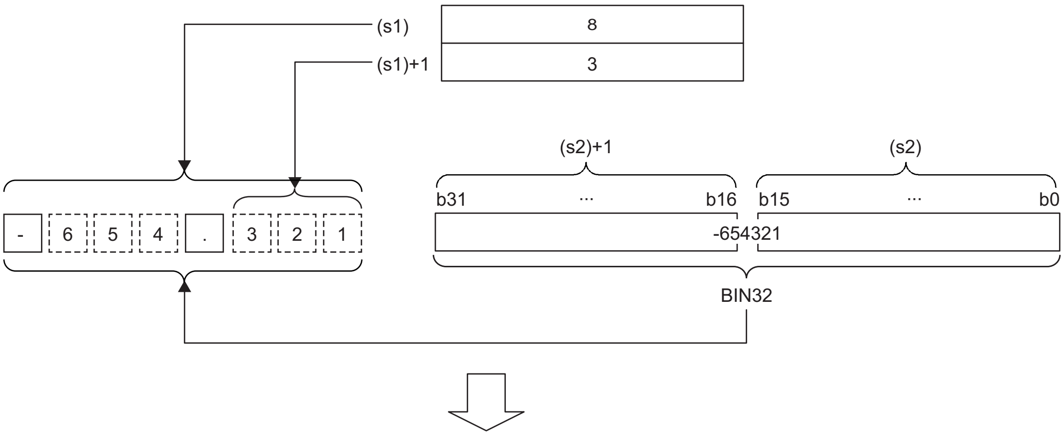


	b15 … b8	b7 … b0
(d)	ASCII code ((total number of digits - 1)th digit)	ASCII code (sign data)
(d)+1	ASCII code ((total number of digits - 3)th digit)	ASCII code ((total number of digits - 2)th digit)
(d)+2	ASCII code ((total number of digits - 5)th digit)	ASCII code ((total number of digits - 4)th digit)
(d)+3	ASCII code ((total number of digits - 7)th digit)	ASCII code ((total number of digits - 6)th digit)
(d)+4	ASCII code ((total number of digits - 9)th digit)	ASCII code ((total number of digits - 8)th digit)
(d)+5	ASCII code ((total number of digits - 11)th digit)	ASCII code ((total number of digits - 10)th digit)
(d)+6	00H	ASCII code ((total number of digits - 12)th digit)

A null character, 00H, is automatically stored at the end of string.

Ex.

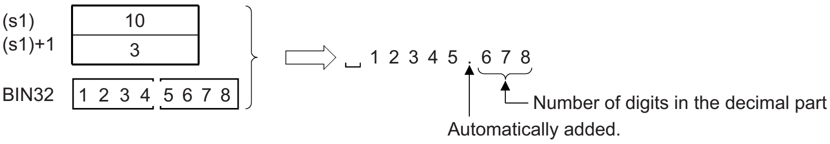
When converting data "-654321" in (s2) into a string assuming that the number of decimal places is three ("-654.321")



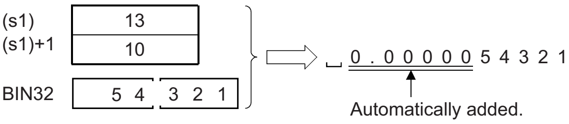
	b15 … b8	b7 … b0
(d)	36H (6)	2DH (-)
(d)+1	34H (4)	35H (5)
(d)+2	33H (3)	2EH (.)
(d)+3	31H (1)	32H (2)
(d)+4	00H	

- The total number of digits that can be specified by (s1) is 2 to 13.
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 10. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.

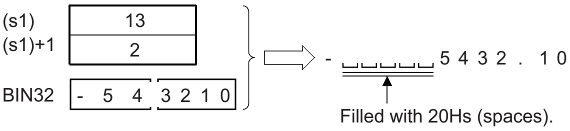
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, 20H (space) is stored if the 32-bit binary data is positive, and 2DH (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.



  - If the specified number of digits in the decimal part is greater than the number of digits of the 32-bit binary data, 0s are automatically added and the data is regarded as "0.□□□□".



  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 32-bit binary data, 20H (space) is stored between the sign and the numeric value. If the number of digits of the 32-bit binary data is greater, an error occurs.



  - The value "00H" is automatically stored at the end of the converted character string.

Operation error

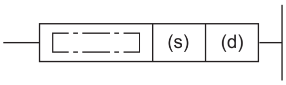
Error code (SD0)	Description
3401H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 13. • The specified number of digits in the decimal part is out of the range, 0 to 10. • The relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following. (Total number of digits)-3≥Number of digits in the decimal part • The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 32-bit binary data specified by (s2). [Number of digits in (s1)] < [Number of digits of 32-bit binary data excluding the sign in (s2) + Number of digits in the sign (+ or -) + Decimal point (.)]

Converting BCD 4-digit data to decimal ASCII code

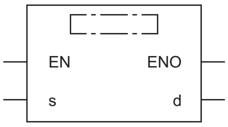
BCDDA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

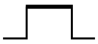
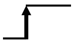
These instructions convert BCD 4-digit data to the ASCII code.

Ladder	ST
 (s) (d)	ENO:=BCDDA(EN,s,d); ENO:=BCDDAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BCDDA	
BCDDAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	BCD data used for ASCII conversion	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the numerical value of each digit of the BCD 4-digit data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the hexadecimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (4 digits)	Page 667 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 667 Operation of when SM705 (Number of conversion digits selection) is on

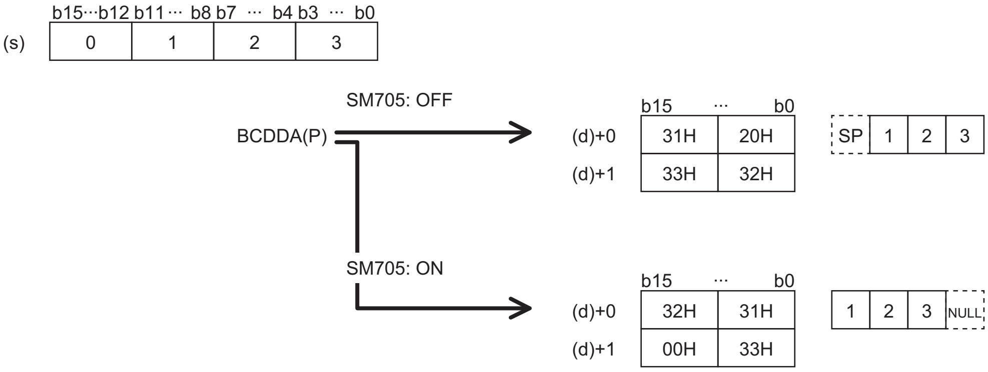
[*1] For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

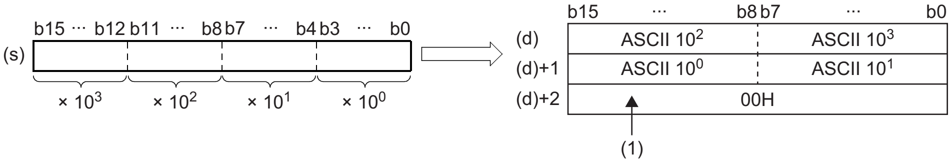
When the BCDDA(P) instruction is executed with BCD 4-digit data "0123" stored in (s)



- When SM705 is off, the number of digits is fixed. The leading "0" of "0123" is converted into 20H (space) and stored.
- When SM705 is on, data is left-justified. "0123" with the leading "0" omitted ("123") is converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits (4 digits) in (d) to (d)+1.



ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

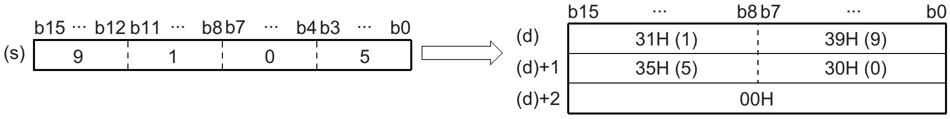
ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+2 when SM701 (Number of output characters selection) is off. When it is on, the value in (d)+3 remains unchanged.

- 20H (space) is stored for the leading "0"s at the left of the effective number of digits of the operation result stored in the device specified by (d). (Zero-suppression) For "0050", "00" becomes 20H (space) and "50" is the effective number of digits.

Ex.

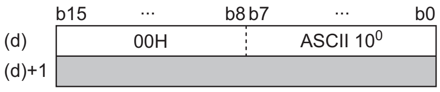
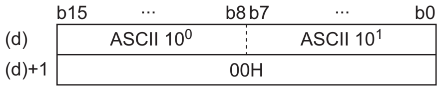
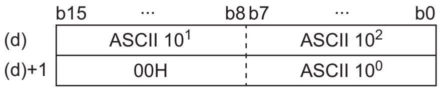
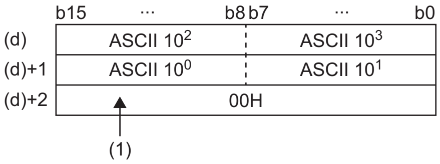
When 9105 is specified in (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data for the number of digits (up to 4 digits) without the leftmost "0" in the effective digits is stored in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+1	Value of (s)	Data of (d) to (d)+1
• 0H to 9H	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to 99H	• (d)+1 is filled with 00H. 
• 100H to 999H	• The upper byte of (d)+1 are filled with 00H. 	• 1000H to 9999H	• (1): (d)+2 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+2 remains unchanged if SM701 is on. 

ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

ASCII 10^0: Ones place of ASCII code

- When the number of digits is less than the maximum number of digits (4 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (4 digits), 00H is stored in (d)+2 when SM701 is off. (d)+2 remains unchanged if SM701 is on.

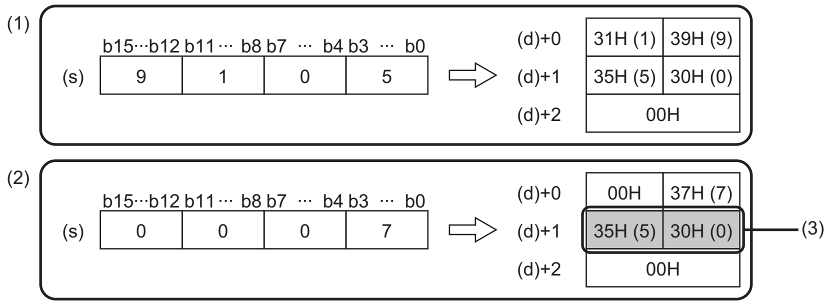
Operation error

Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 9999.

Precautions

When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BCDDA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BCDDA(P) instruction when (s) is "9105H" and then executing another BCDDA(P) instruction when (s) is "0007H"

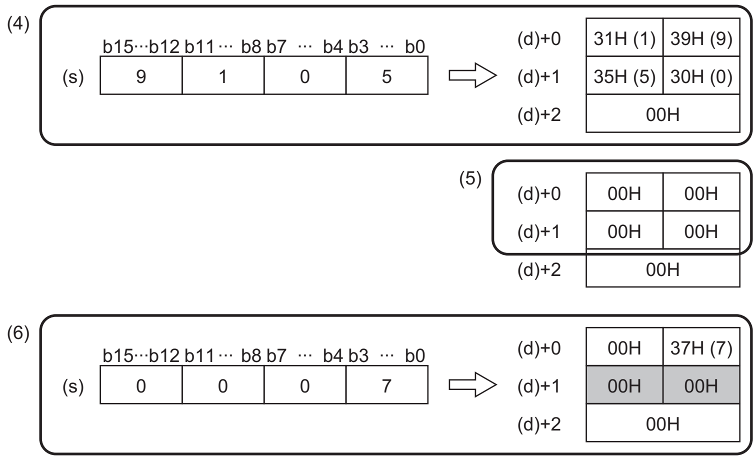


(1) "9105" is converted into a string.

(2) "7" is converted into a string.

(3) A part of the previous conversion result remains in (d)+1.

To avoid this, create a program to clear the entire data storage areas (d)+0 and (d)+1 before executing the BCDDA(P) instruction.



(4) "9105" is converted into a string.

(5) (d)+0 to (d)+1 are cleared.

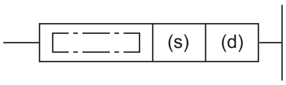
(6) "7" is converted into a string.

Converting BCD 8-digit data to decimal ASCII code

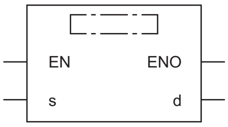
DBCDDA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

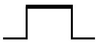
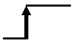
These instructions convert BCD 8-digit data to the ASCII code.

Ladder	ST
	ENO:=DBCDDA(EN,s,d); ENO:=DBCDDAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCDDA	
DBCDDAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	BCD data used for ASCII conversion	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the numerical value of each digit of the BCD 8-digit data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705[*1]	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (8 digits)	Page 671 Operation of when SM705 (Number of conversion digits selection) is off
ON	Each digit is stored left-justified depending on the value of (s).	Page 672 Operation of when SM705 (Number of conversion digits selection) is on

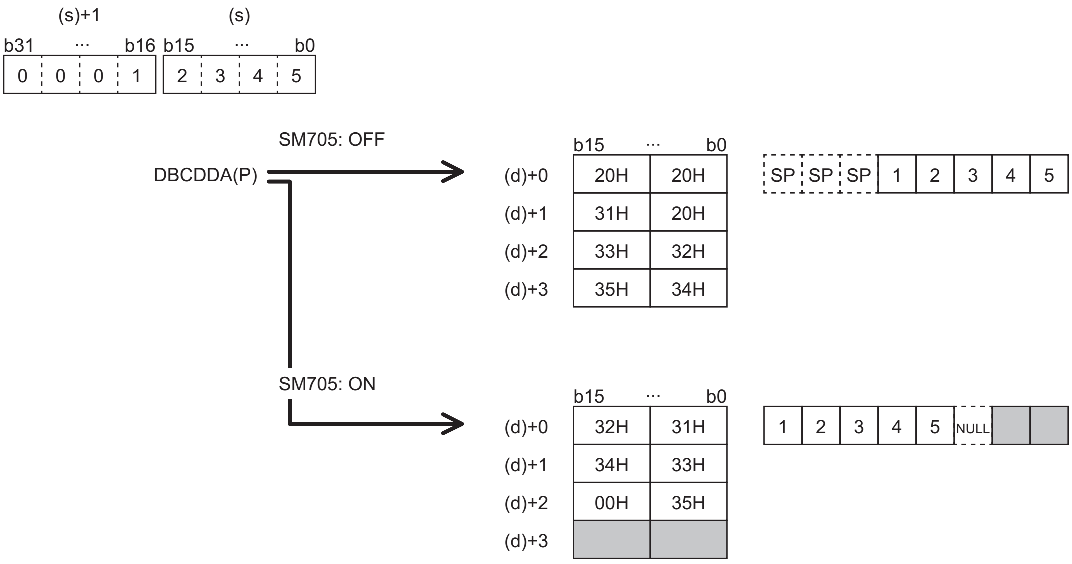
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

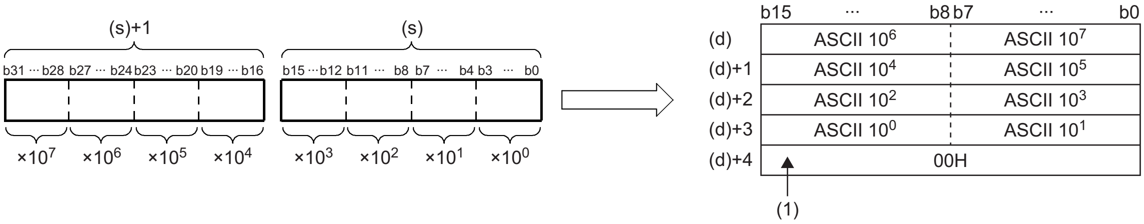
When the DBCDDA(P) instruction is executed with BCD 8-digit data "00012345" stored in (s)



- When SM705 is off, the number of digits is fixed. Each of the leading "0"s of "00012345" is converted into 20H (space) and stored.
- When SM705 is on, data is left-justified. "00012345" with the leading "0"s omitted ("12345") is converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits (8 digits) in (d) to (d)+3.



ASCII 10^7: Ten-millions place of ASCII code

ASCII 10^6: Millions place of ASCII code

⋮

ASCII 10^1: Tens place of ASCII code

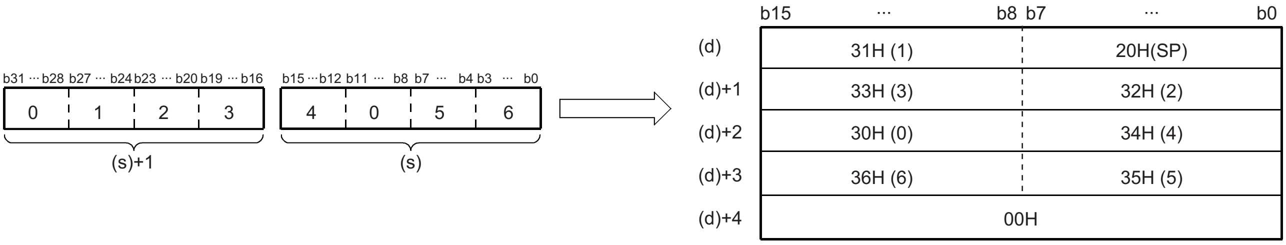
ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. When it is on, the value in (d)+3 remains unchanged.

- 20H (space) is stored for the leading "0"s at the left of the effective number of digits of the operation result stored in the device specified by (d). (Zero-suppression) For "00012098", "000" becomes 20H (space) and "12098" is the effective number of digits.

Ex.

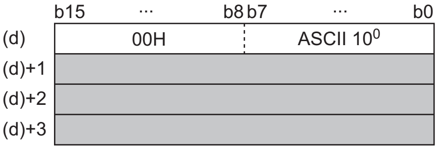
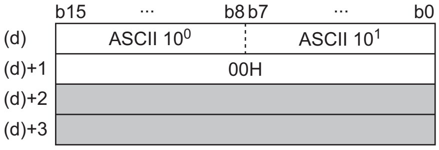
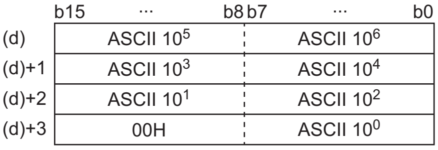
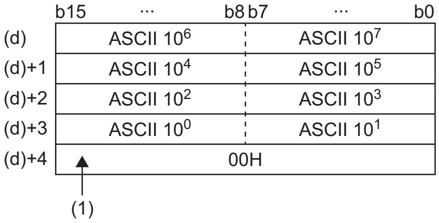
When 01234056 is specified in (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data for the number of digits (up to 8 digits) without the leftmost "0" in the effective digits is stored in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+3	Value of (s)	Data of (d) to (d)+3
• 0H to 9H	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to 99H	• (d)+1 is filled with 00H. • (d)+2 and later remain unchanged. 
⋮			
• 1000000H to 9999999H	• The upper byte of (d)+3 is filled with 00H. 	• 10000000H to 99999999H	• (1): (d)+4 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on. 

ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
⋮
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- When the number of digits is less than the maximum number of digits (8 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (8 digits), 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on.

Operation error

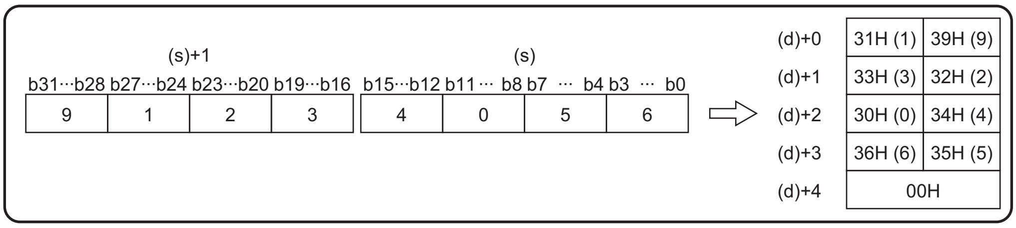
Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 99999999.

Precautions

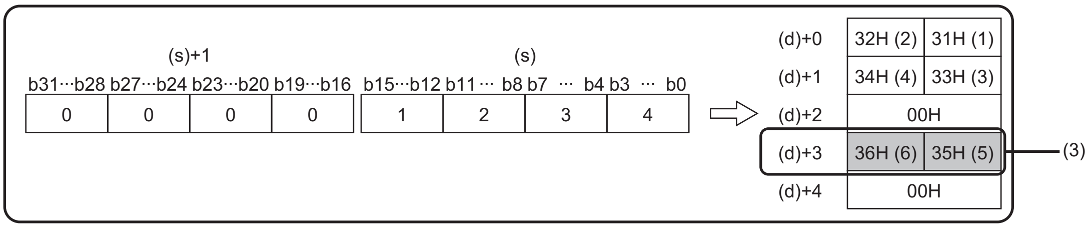
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBCDDA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the DBCDDA(P) instruction when (s) is "91234056H" and then executing another DBCDDA(P) instruction when (s) is "00001234H"

(1)



(2)



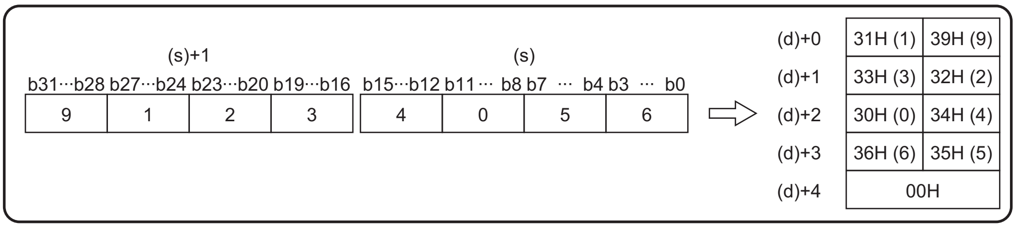
(1) "91234056" is converted into a string.

(2) "1234" is converted into a string.

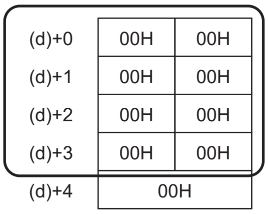
(3) A part of the previous conversion result remains in (d)+3.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+3 before executing the DBCDDA(P) instruction.

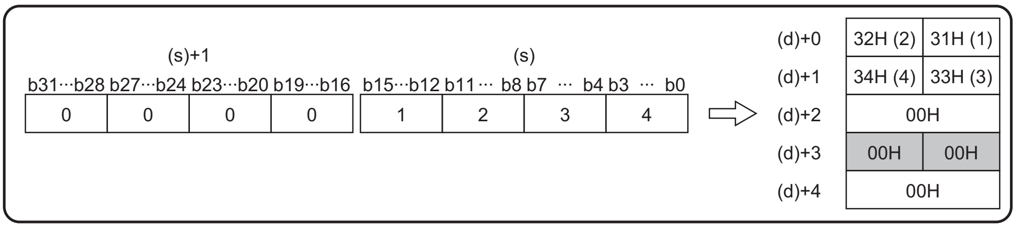
(4)



(5)



(6)



(4) "91234056" is converted into a string.

(5) (d)+0 to (d)+3 are cleared.

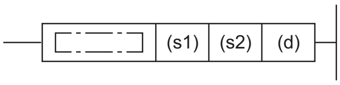
(6) "1234" is converted into a string.

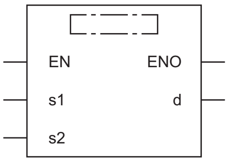
Converting single-precision real number to string data

ESTR(P)

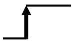
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions convert single-precision real number data to a string according to the display specification.

Ladder	ST
	ENO:=ESTR(EN,s1,s2,d); ENO:=ESTRP(EN,s1,s2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
ESTR	
ESTRP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Single-precision real number data to be converted, or the start device containing the data	0, $2^{-126}<\|(s1)\|<2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Start device containing the display specification of the real number to be converted	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

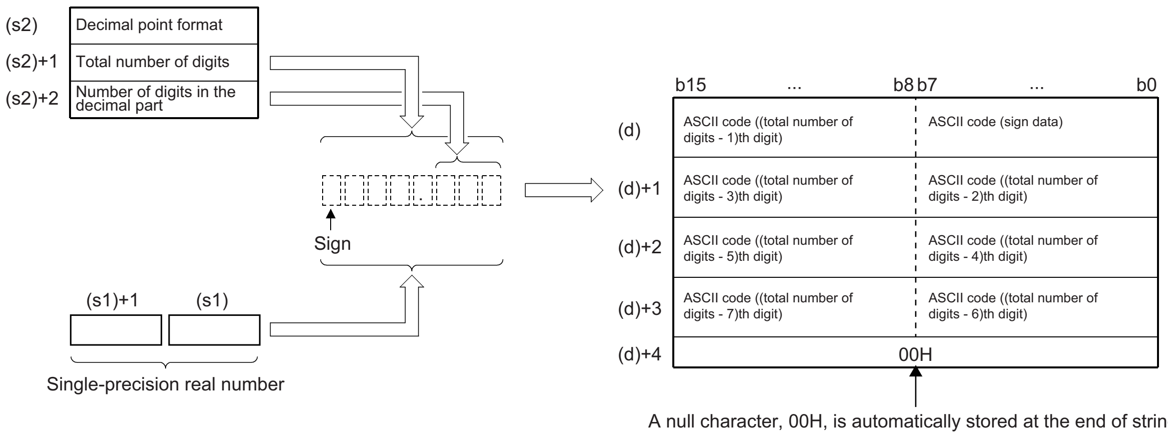
Processing details

- These instructions convert the single-precision real number data stored in the device specified by (s1) to a character string according to the display specification stored in the device number specified by (s2) and later, and store the converted data in the device number specified by (d) and later.
- The type of the converted data varies depending on the display specification stored in the device specified by (s2).

(s2)	0: Decimal point format 1: Exponent format
(s2)+1	Total number of digits
(s2)+2	Number of digits in the decimal part

■Decimal point format

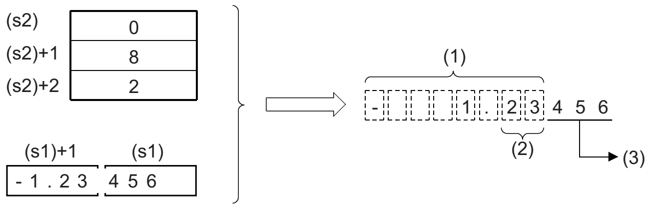
- When 0 is specified in (s2), the decimal point format is used.



- Total number of digits in the device specified by (s2)+1: When the number of digits in the decimal part is 0, total number of digits (maximum of 24)≥number of digits in the integral part[*1]+1. When it is a value other than 0, total number of digits (maximum of 24)≥(number of digits in the integral part[*1]+number of digits in the decimal part+2).

[*1] Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

- The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, 20H (space) is stored if the single-precision real number data is positive, and 2DH (-) is stored if the data is negative.
  - If the decimal part of the single-precision real number data is not stored within the range of the number of digits in the decimal part, the lower decimal digits are rounded off.

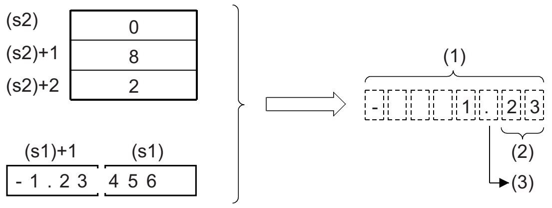


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Rounded off.

  - If the number of digits in the decimal part is set to a value other than 0, 2EH (.) is automatically stored at the position of the specified number of digits in the decimal part plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.

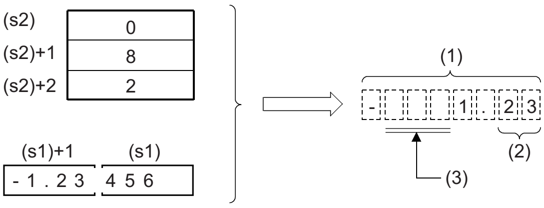


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Automatically added.

- If the number of digits excluding the sign, decimal point, and decimal part from the total number of digits is greater than the number of digits in the integral part of single-precision real number data, 20H (space) is stored between the sign and integral part.



(1) Total number of digits

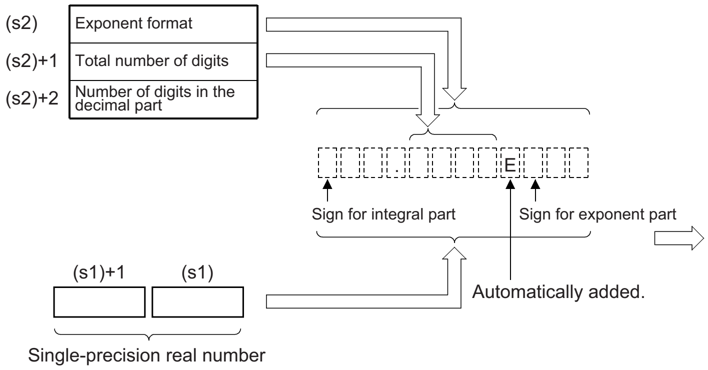
(2) Number of digits in the decimal part

(3) Filled with 20Hs (spaces).

- The value "00H" is automatically stored at the end of the converted character string.
- The number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1) can be 1 to 16.

■Exponential format

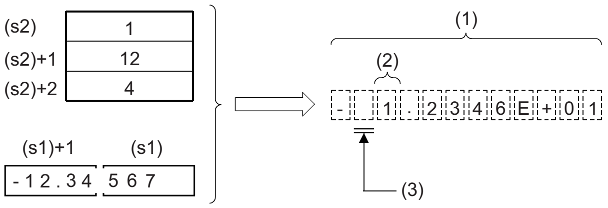
- When 1 is specified in (s2), the exponential format is used.



	b15 ... b8	b7 ... b0
(d)	ASCII code ((total number of digits - 1)th digit)	ASCII code (sign data for integral part)
(d)+1	ASCII code (decimal point (.)) (2EH)	ASCII code ((total number of digits - 2)th digit)
(d)+2	ASCII code ((total number of digits - 5)th digit)	ASCII code ((total number of digits - 4)th digit)
(d)+3	ASCII code ((total number of digits - 7)th digit)	ASCII code ((total number of digits - 6)th digit)
(d)+4	ASCII code (sign data for exponent part)	45H (E)
(d)+5	ASCII code ((total number of digits - 11)th digit)	ASCII code ((total number of digits - 10)th digit)
(d)+6	00H	

A null character, 00H, is automatically stored at the end of string.

- Total number of digits in the device specified by (s2)+1: When the number of digits in the decimal part is 0, total number of digits (maximum of 24)≥2. When it is a value other than 0, total number of digits (maximum of 24)≥(number of digits in the decimal part+7).
- The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 7.
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
- As sign data in the integral part, 20H (space) is stored if the single-precision real number data is positive, and 2DH (-) is stored if the data is negative.
- The integral part is fixed to one digit. 20H (space) is stored between the integral part and sign.

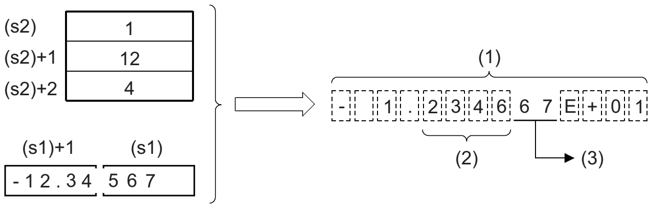


(1) Total number of digits

(2) Fixed to 1 digit

(3) Filled with 20Hs (spaces).

- If the decimal part of the single-precision real number data is not stored within the range of the number of digits in the decimal part, the lower decimal digits are rounded off.

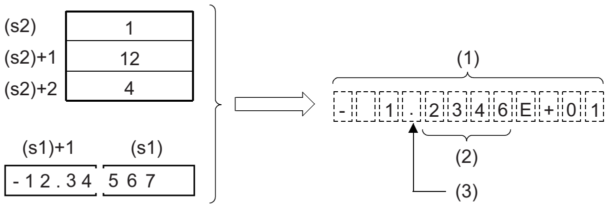


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Rounded down.

- If the number of digits in the decimal part is set to a value other than 0, 2EH (.) is automatically stored at the position of the specified number of digits in the decimal part plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.

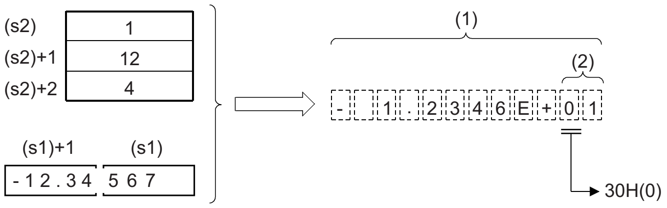


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Automatically added.

- As sign data in the exponent part, 2BH (+) is stored if the exponent is positive, and 2DH (-) is stored if it is negative.
- The exponent part is fixed to two digits. When the exponent part is one digit, 30H(0) is stored between the exponent part and sign.



(1) Total number of digits

(2) Fixed to 2 digits

- The value "00H" is automatically stored at the end of the converted character string.
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3401H	The value in the device specified by (s1) is not in the following range. $0, 2^{-126} \leq \|(s1)\| < 2^{128}$
	Invalid data that cannot be converted is set to (s2). • The format specification in the device specified by (s2) is a value other than 0 and 1. • In the decimal point format, the total number of digits specified by (s2)+1 is outside the following range. The number of digits in the decimal part is 0: Total number of digits≥number of digits in the integral part*1+1 The number of digits in the decimal part is not 0: Total number of digits≥number of digits in the integral part*1+ number of digits in the decimal point + 2 • In the exponential format, the total number of digits specified by (s2)+1 is outside the following range. The number of digits in the decimal part is 0: Total number of digits≥6 The number of digits in the decimal part is not 0: Total number of digits≥(number of digits in the decimal part+7) • In the exponential point format, the decimal part digit specification in the device specified by (s2)+2 is outside the following range. Decimal point format: Number of digits in the decimal part≤(total number of digits - 3) Exponential format: Number of digits in the decimal part≤(Total number of digits - 7) • In the decimal point format, the number of digits in the integral part of 32-bit floating point real number data in the device specified by (s1) exceeds 16. • The number of digits in the decimal part specified by (s2)+2 is out of the range from 0 to 7.
	The specified value consists of more than 24 digits in total.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.

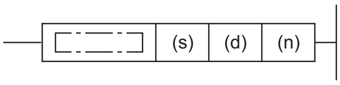
*1 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

Converting hexadecimal binary data to hexadecimal ASCII code

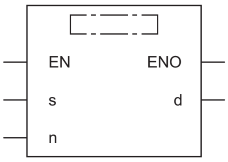
INT2ASC(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

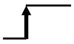
These instructions convert 16-bit binary data to the hexadecimal ASCII code and store it in any specified range.

Ladder	ST
	ENO:=INT2ASC(EN,s,n,d); ENO:=INT2ASCP(EN,s,n,d)

FBD/LD



■Execution condition

Instruction	Execution condition
INT2ASC	
INT2ASCP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the binary data to be exchanged to a character string	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be stored	0 to 16383	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

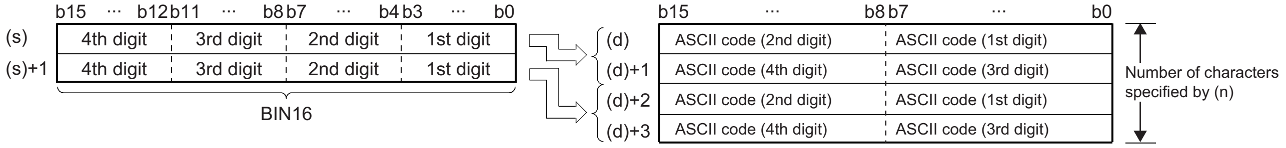
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

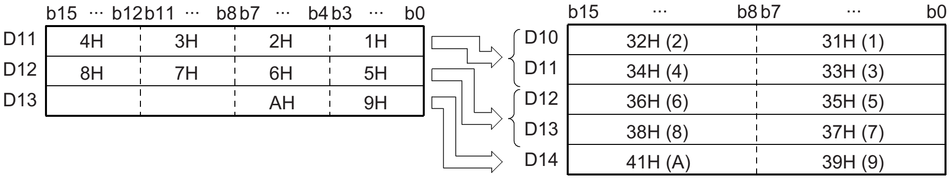
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

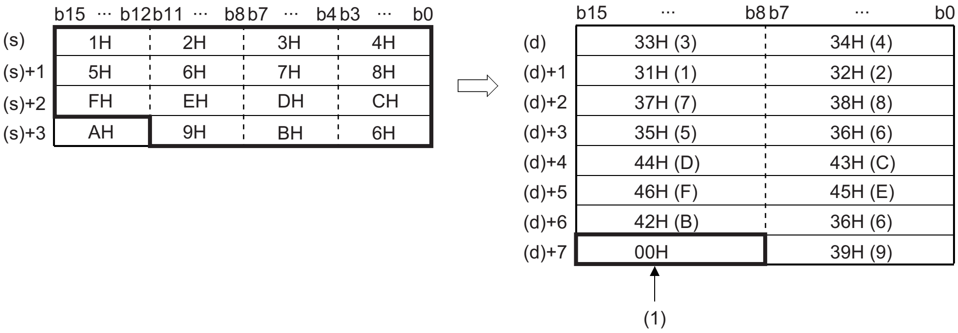
- Converts the 16-bit binary data in the device number specified by (s) and later to hexadecimal ASCII, and stores the converted data by the number of characters in the device specified by (n) in the device number specified by (d) and later.



- Setting the number of bytes by (n) automatically determines the range of binary data in the device specified by (s) and the range of the device specified by (d) for storing the character string data.
- Processing is performed normally even if the device range in which the binary data to be converted and the device range for storing the converted binary data are overlapping.



- If the number of characters in the device specified by (n) is an odd number, 00H is automatically stored in the upper 8 bits of the last device number among device numbers for storing the converted character string data.



(1) 00H is automatically stored.

- If the number of characters in the device specified by (n) is 0, no processing is performed.

Operation error

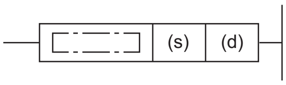
Error code (SD0)	Description
3405H	An out-of-range value is set to (n). • The specified number of characters is not between 0 and 16383.

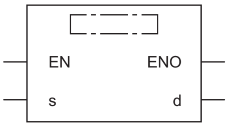
Converting Unicode character string to Shift JIS character string

WS2SJIS(P)

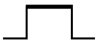
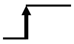
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions convert a Unicode character string to a Shift JIS character string.

Ladder	ST
	ENO:=WS2SJIS(EN,s,d); ENO:=WS2SJISP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
WS2SJIS	
WS2SJISP	

Setting data

■Description, range, data type

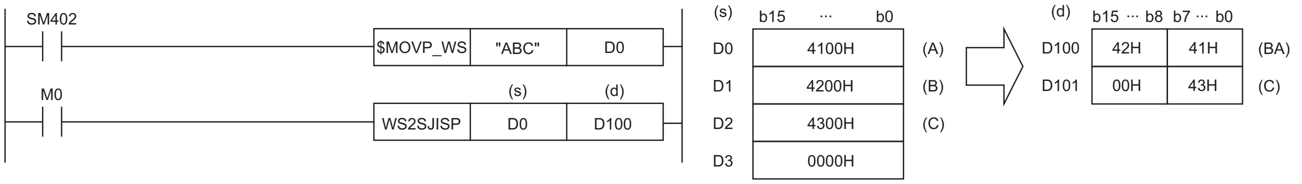
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the character string to be converted	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

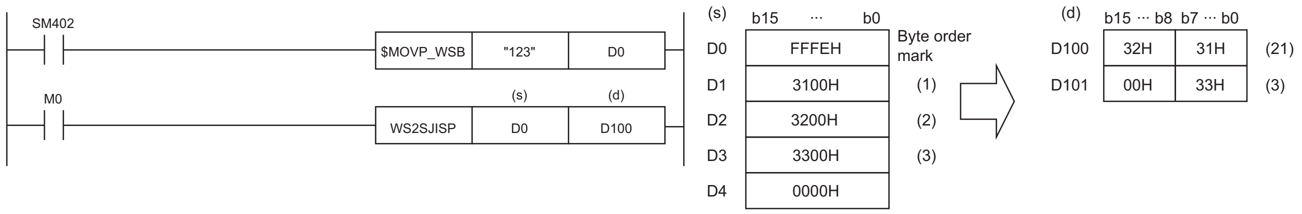
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instruction convert the Unicode character string in the device specified by (s) to the shift JIS character string, and stores the converted data in the device specified by (d).
- The Unicode character string in the device specified by (s) should be specified in little endian.
- When a byte order mark is not used, conversion from Unicode to shift JIS occurs as follows.



- When a byte order mark (FEFFH) is added, conversion from Unicode to shift JIS occurs as follows.



Operation error

Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3401H	Byte order mark FEFFH (big endian) is added to the character string in the device specified by (s).
	The range of data in the device specified by (s) includes a character code that cannot be converted.
3405H	The character string in the device specified by (s) exceeds 16383 characters.[*1]

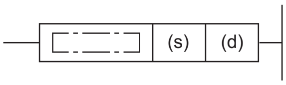
[*1] A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Converting shift JIS character string to Unicode character string (without byte order mark)

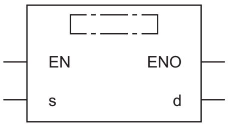
SJIS2WS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

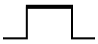
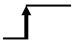
These instructions convert a Shift JIS character string to a Unicode character string.

Ladder	ST
	ENO:=SJIS2WS(EN,s,d); ENO:=SJIS2WSP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SJIS2WS	
SJIS2WSP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the character string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted character string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

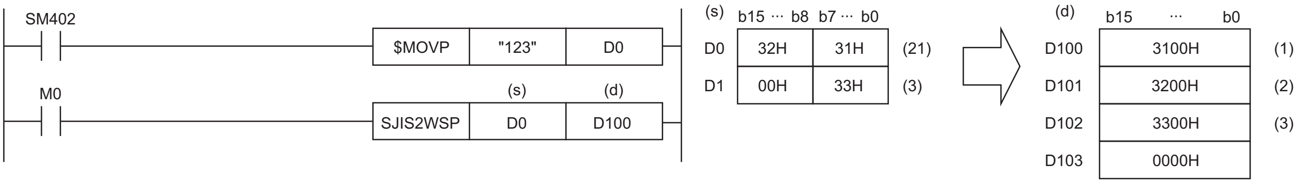
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS character string in the device specified by (s) to a Unicode character string, and store the converted data in the device specified by (d).
- The shift JIS character string in the device specified by (s) should be specified in little endian.
- The SJIS2WS(P) instruction does not add a byte order mark to the beginning of the data in the device specified by (d). To add a byte order mark, use the SJIS2WSB(P) instruction.

☞ Page 685 SJIS2WSB(P)

- The following figure shows the operation for converting shift JIS to Unicode.



Operation error

Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3401H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3405H	The character string in the device specified by (s) exceeds 16383 characters.[1]

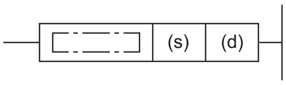
[1] A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Converting shift JIS character string to Unicode (with byte order mark)

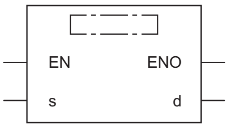
SJIS2WSB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

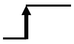
These instructions convert a shift JIS character string to a Unicode character string, and adds a byte order mark to the start of the converted data.

Ladder	ST
	ENO:=SJIS2WSB(EN,s,d); ENO:=SJIS2WSBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SJIS2WSB	
SJIS2WSBP	

Setting data

■Description, range, data type

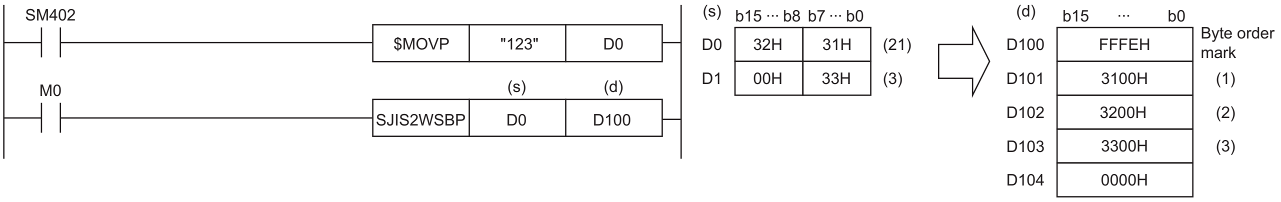
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the character string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted character string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS character string in the device specified by (s) to the Unicode character string, add a byte order mark to the start of the converted data, and store it in the device specified by (d).
- The shift JIS character string in the device specified by (s) should be specified in little endian.
- The following figure shows the operation for converting shift JIS to Unicode.



Operation error

Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3401H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3405H	The character string in the device specified by (s) exceeds 16383 characters.[1]

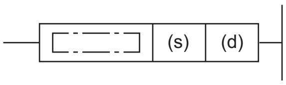
[1] A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Detecting a string length

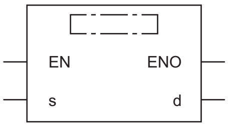
LEN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions detect the length of the specified string.

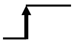
Ladder	ST[*1]
	ENO:=LENP(EN,s,d);

FBD/LD[*1]



*1 The LEN instruction does not support the ST and FBD/LD. Use the standard function, LEN.
☞ Page 1944 LEN(_E)

■Execution condition

Instruction	Execution condition
LEN	
LENP	

Setting data

■Description, range, data type

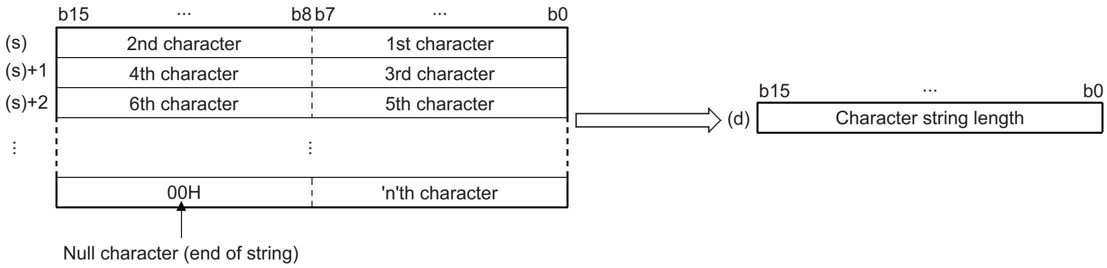
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Number of the device for storing the length of the detected character string	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

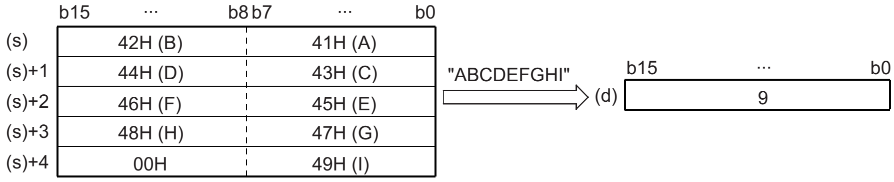
Processing details

- These instructions detect the length of the character string in the device specified by (s) and store it in the device number specified by (d) and later. The data stored in device numbers starting from the one specified by (s) to the one containing 00H is processed as a character string.



Ex.

When "ABCDEFGHI" is stored in the device specified by (s) and later



Operation error

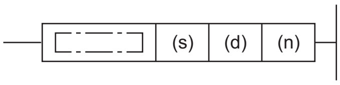
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

Extracting string data from the right

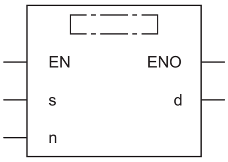
RIGHT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These instructions extract (n) characters of data from the right of string data.

Ladder	ST[1]
	ENO:=RIGHTP(EN,s,n,d);

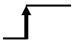
FBD/LD[1]



---

[1] The RIGHT instruction does not support the ST and FBD/LD. Use the standard function, RIGHT.
☞ Page 1946 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
RIGHT	
RIGHTP	

Setting data

■Descriptions, ranges, and data types

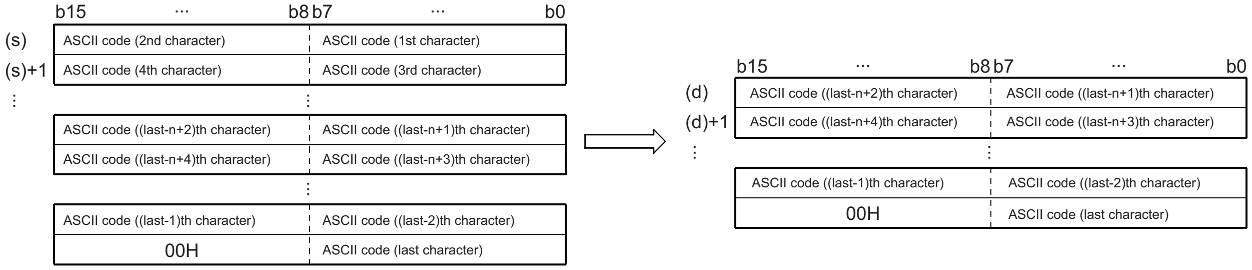
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of character string extracted from the right of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be extracted	1 to 16383	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

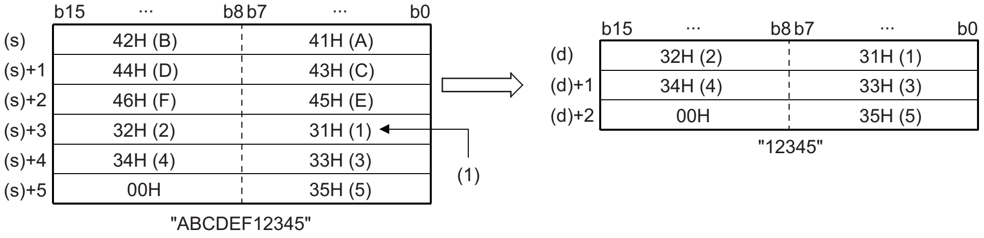
Processing details

- These instructions extract (n) characters of data from the right of the character string data (the end of the character string) stored in the device number specified by (s) and later, and store the extracted data in the device number specified by (d) and later.



Ex.

When (n)=5



(1) ASCII code (5th character)

- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character string data.
- When the number of characters in the device specified by (n) is 0, NULL code (00H) is stored in the device specified by (n).

Operation error

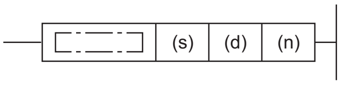
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	An out-of-range value is set to (s). • The number of characters of the character string in the device specified by (s) exceeds 16383. • The number of characters of the character string in the device specified by (s) is 0.
	The number of characters in the device specify by (n) exceeds that in the device specified by (s).

Extracting string data from the left

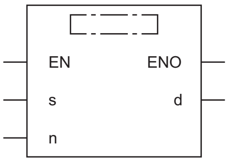
LEFT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions extract (n) characters of data from the left of the string data, and store the extracted data in the device number specified by (d) and later.

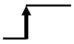
Ladder	ST[*1]
	ENO:=LEFTP(EN,s,n,d);

FBD/LD[*1]



[*1] The LEFT instruction does not support the ST and FBD/LD. Use the standard function, LEFT.
☞ Page 1946 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
LEFT	
LEFTP	

Setting data

■Descriptions, ranges, and data types

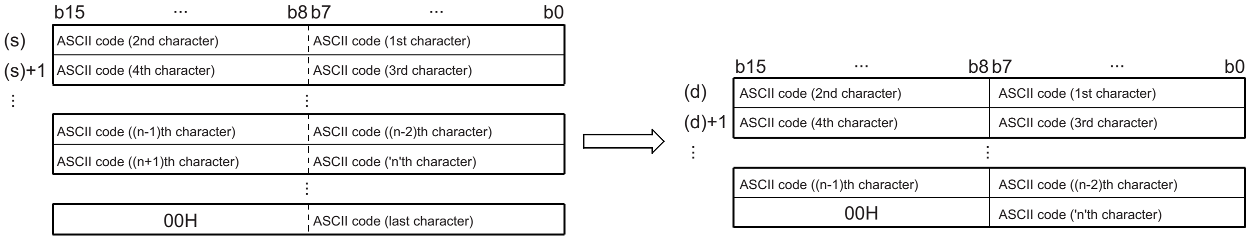
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of character string extracted from the left of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be extracted	1 to 16383	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

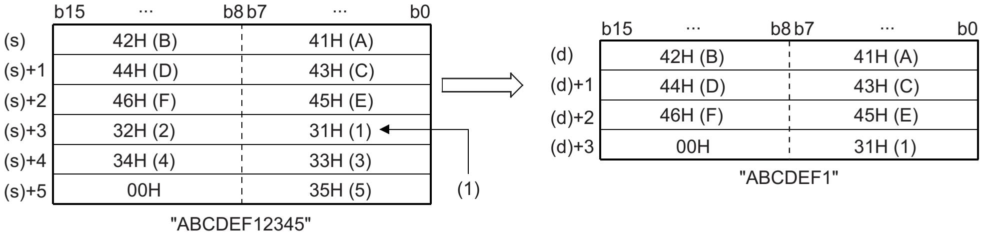
Processing details

- These instructions extract (n) characters of data from the left of the character string data (the start of the character string) stored in the device number specified by (s) and later, and store the extracted data in the device number specified by (d) and later.



Ex.

When (n)=7



(1) ASCII code (7th character)

- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character string data.
- When the number of characters in the device specified by (n) is 0, NULL code (00H) is stored in the device specified by (n).

Operation error

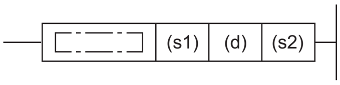
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	An out-of-range value is set to (s). • The number of characters of the character string in the device specified by (s) exceeds 16383. • The number of characters of the character string in the device specified by (s) is 0.
	The number of characters in the device specify by (n) exceeds that in the device specified by (s).

Extracting the specified string data

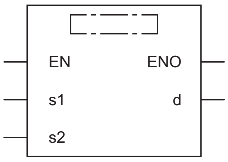
MIDR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

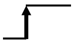
These instructions extract data at any position in string data, and store the extracted data in the device number specified by (d) and later.

Ladder	ST
	ENO:=MIDR(EN,s1,s2,d); ENO:=MIDRP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MIDR	
MIDRP	

Setting data

■Description, range, data type

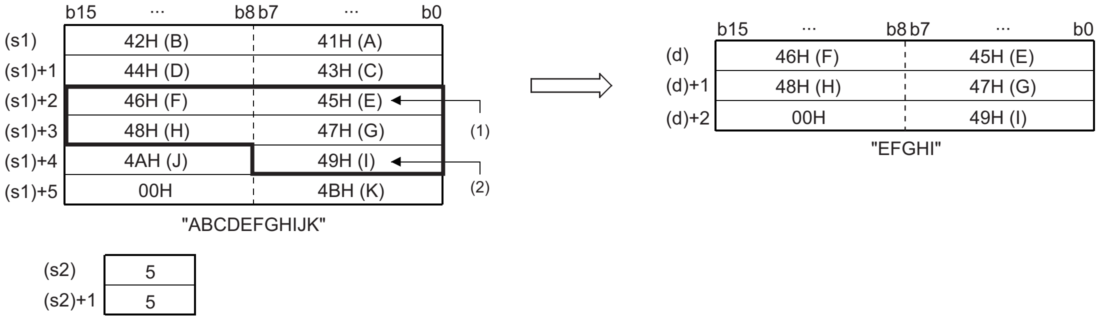
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the character string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location of the start character and the number of characters (s2): Location of start character, (s2)+1: Number of characters	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

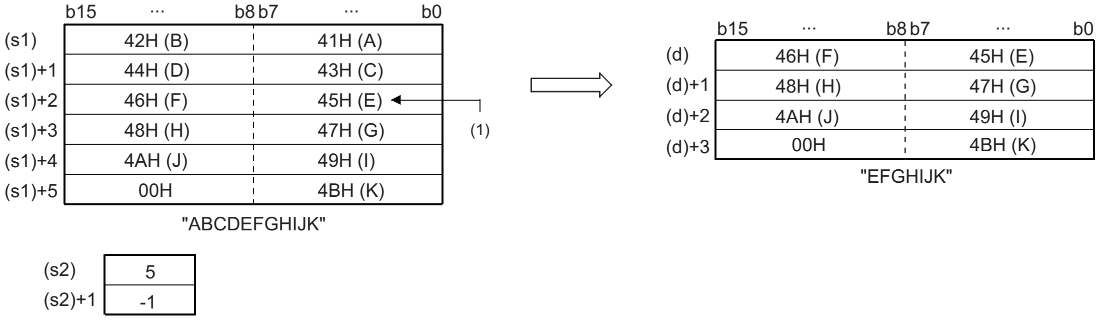
- These instructions extract the data by the number of characters specify by (s2)+1 from the location specified by (s2) in the character string data stored in the device number specified by (s1) and later, and store the extracted data in the device number specified by (d) and later.



(1) Position specified by (s2): 5th character

(2) ASCII code ((s2)+1)th character from the specified position)

- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character string data.
- If the number of characters in the device specified by (s2)+1 is 0, no processing is performed.
- When the number of characters of the data in the device specified by (s2)+1 is -1, the data till the last character data in the device specified by (s1) is stored in the device specified by (d) and later.



(1) Position specified by (s2): 5th character

Operation error

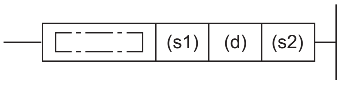
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
3405H	The number of characters of the character string in the device specified by (s1) exceeds 16383.
	An out-of-range value is set to (s2). • The value in the device specified by (s2) is 0 or less. • The value in the device specified by (s2)+1 is other than the valid values (-1, 0, 1 or bigger). • The value in the device specify by (s2) exceeds the number of characters in the device specified by (s1). • The value obtained by adding those in the devices specify by (s2) and (s2)+1 exceeds the number of characters in the device specified by (s1).

Replacing the specified string data

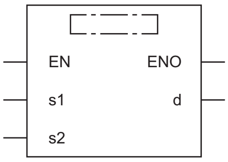
MIDW(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

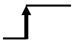
These instructions replace the data at the specified location in the string data with the specified string.

Ladder	ST
	ENO:=MIDW(EN,s1,s2,d); ENO:=MIDWP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MIDW	
MIDWP	

Setting data

■Description, range, data type

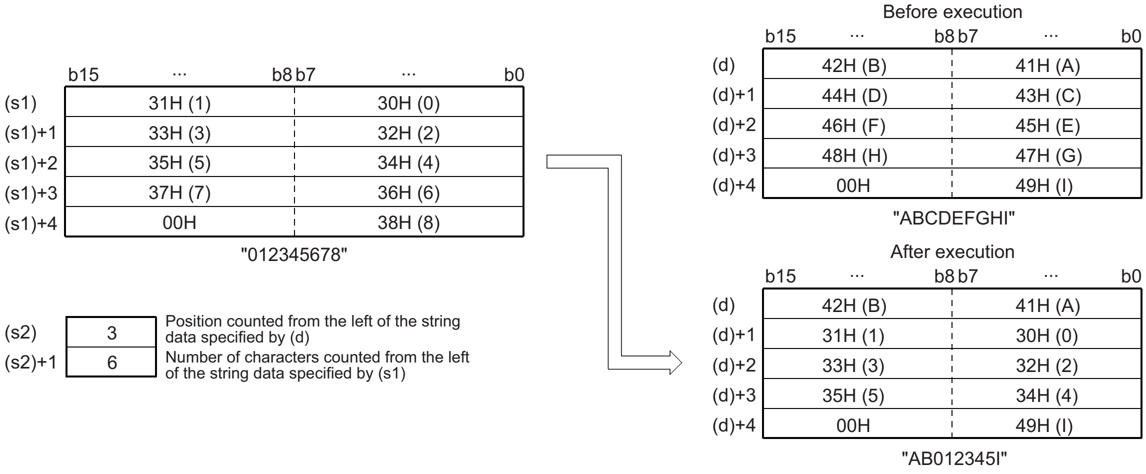
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the character string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location of the start character and the number of characters (s2): Location of start character, (s2)+1: Number of characters	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

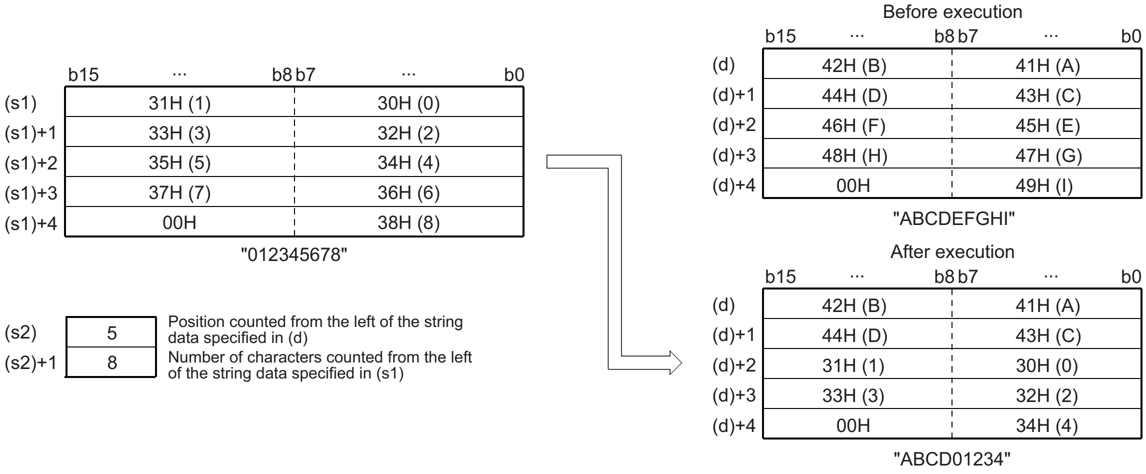
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions read data by the number of characters stored in the device specified by (s2)+1 from the character string data stored in the device number specified by (s1) and later, and store the read data at the location in the device specified by (s2) and later in the character string stored in the device number specified by (d) and later.

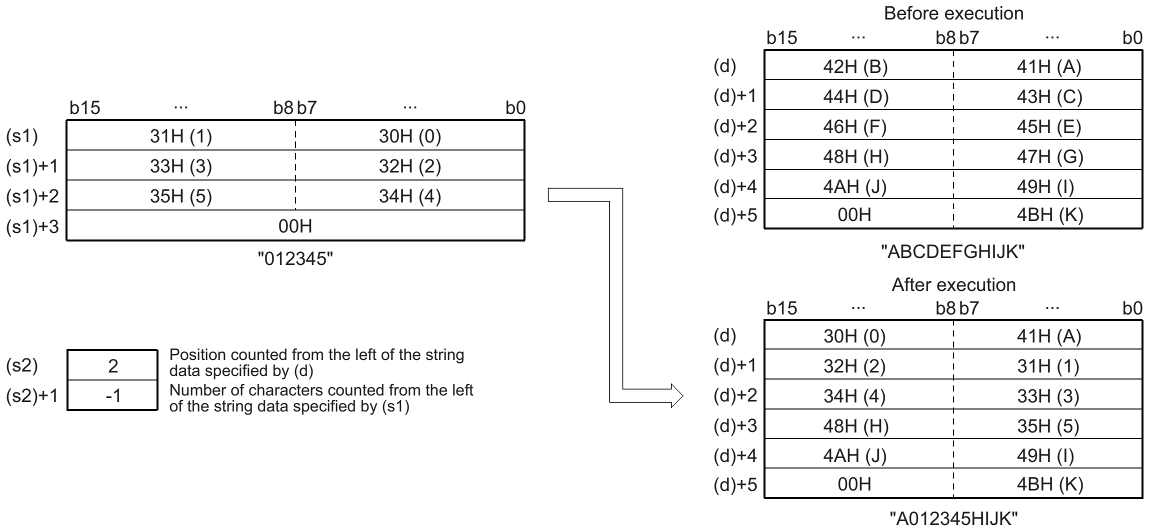


- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character string data.
- If the number of characters in the device specified by (s2)+1 is 0, no processing is performed.
- If the number of characters in the device specified by (s2)+1 exceeds the last character of the character string data in the device specified by (d), the data is stored up to the last character.



Data in 35H (5) to 37H (7) are not stored.

- When the number of characters of the data in the device specified by (s2)+1 is -1, the data till the last character data in the device specified by (s1) is stored in the device specified by (d) and later.



Operation error

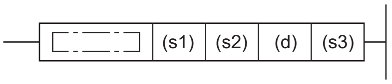
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
3405H	The number of characters of the character string in the device specified by (s1) exceeds 16383.
	The number of characters of the character string in the device specified by (d) exceeds 16383.
	An out-of-range value is set to (s2). • The value in the device specified by (s2) is 0 or less. • The value in the device specified by (s2)+1 is other than the valid values (-1, 0, 1 or bigger). • The value in the device specify by (s2) exceeds the number of characters in the device specified by (d). • The value in the device specify by (s2)+1 exceeds the number of characters in the device specified by (s1).

Searching string data

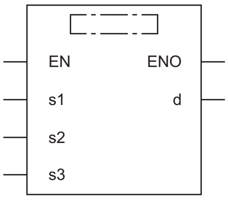
INSTR(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

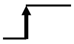
These instructions search string data for the specified string.

Ladder	ST
	ENO:=INSTR(EN,s1,s2,s3,d); ENO:=INSTRP(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INSTR	
INSTRP	

Setting data

■Descriptions, ranges, and data types

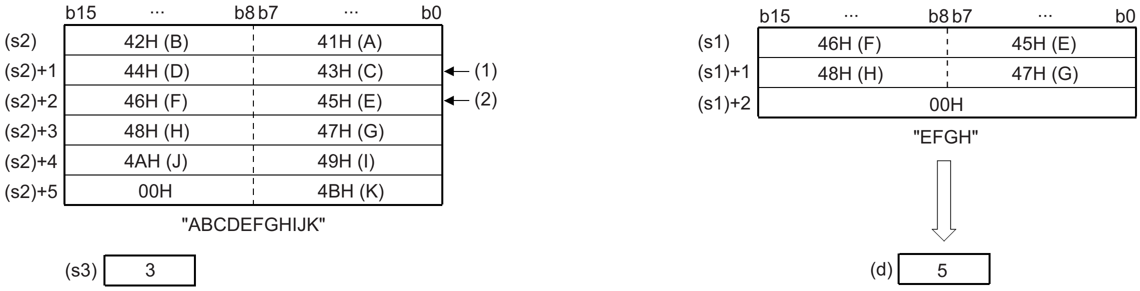
Operand	Description	Range	Data type	Data type (label)
(s1)	Character strings to be searched or the start device containing these character strings	—	String	ANYSTRING_SINGLE
(s2)	Character string to be searched for or the start device containing the character string to be searched for	—	String	ANYSTRING_SINGLE
(d)	Device for storing the search result	—	16-bit signed binary	ANY16
(s3)	Search start position	1 to 16383	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the character string data stored in the device number specified by (s2) and later starting from the (s3)th character from the left for the character string data stored in the device number specified by (s1) and later, and store the search result in the device specified by (d). The search result stored shows the number of characters from the start character of the character string data in the device specified by (s2).



(1) Starting position specified by (s3): 3rd character

(2) 5th character from the beginning of the data

- If no character string data is matching, 0 is stored in the device specified by (d).

Operation error

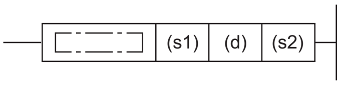
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s2) and later.
3405H	An out-of-range value is set to (s1). • The number of characters of the character string that has been set is 0. • The number of characters of the character string that has been set exceeds 16383.
	The number of characters of the character string that has been set in the device specified by (s2) exceeds 16383.
	An out-of-range value is set to (s3). • The value in the device specify by (s3) exceeds the number of characters in the device specified by (s2). • The value in the device specified by (s3) is negative or 0.

Inserting string data

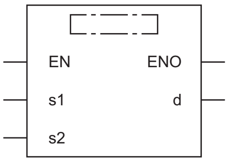
STRINS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

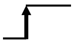
These instructions insert the specified string data into the specified position of the string data.

Ladder	ST
	ENO:=STRINS(EN,s1,s2,d); ENO:=STRINSP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STRINS	
STRINSP	

Setting data

■Descriptions, ranges, and data types

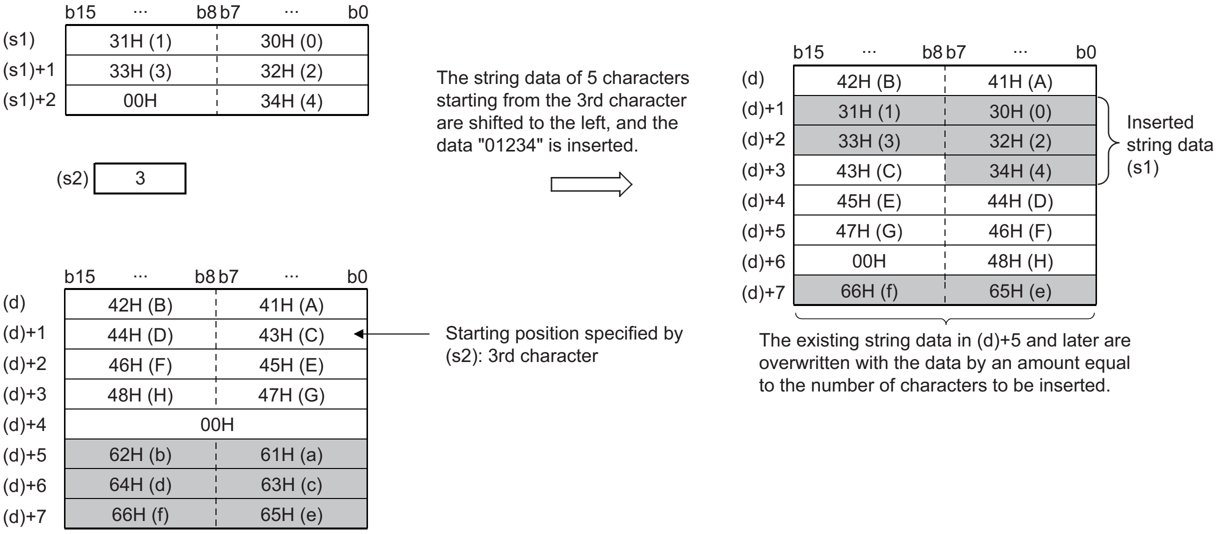
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string to be inserted or the start device containing the character string to be inserted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the insertion result character string	—	String	ANYSTRING_SINGLE
(s2)	Insertion position (bytes)	1 to 16383	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions insert the character string data in the device specified by (s1) to the (s2)th character (insertion position) from the start of the character string data in the device specified by (d).



- If the character string after insertion in the device specified by (s1)+(d) is even, the NULL code (00H) is stored in the device (1 word) next to the last one containing the character string.
- If the character string after insertion in the device specified by (s1)+(d) is odd, the NULL code (00H) is stored in the last device (upper 8 bits) of the character string.
- When the number of characters in the device specified by (d) plus 1 is specified in (s2), the character string in the device specified by (s1) is concatenated to the end of the character string in the device specified by (d).

Operation error

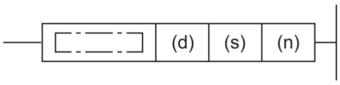
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The devices specified by (s1) and (d), both containing character strings, are overlapping even partly.
	The device specified by (s1)+(d) containing the character string after insertion overlaps with the character string storage device specified by (s1).
3405H	The number of characters of the character string in the device specified by (s1) exceeds 16383.
	An out-of-range value is set to (s2). • The specified value exceeds the number of characters plus 1 of the character string in the device specified by (d). • The specified value is not within the following range. 1≤(s2)≤16383
	The number of characters of the character string in the device specified by (d) exceeds 16383.
3406H	The character string after insertion stored in the device specified by (s1)+(d) becomes data outside the output enable range. • The number of characters of the character string after insertion in the device specified by (d) exceeds 16383. • The character string after insertion exceeds each setting area in the specified device/label memory.

Deleting string data

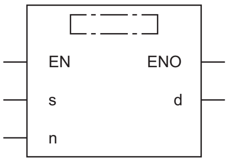
STRDEL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

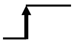
These instructions delete (n) characters starting from the specified position of string data.

Ladder	ST
	ENO:=STRDEL(EN,s,n,d); ENO:=STRDELP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STRDEL	
STRDELP	

Setting data

■Descriptions, ranges, and data types

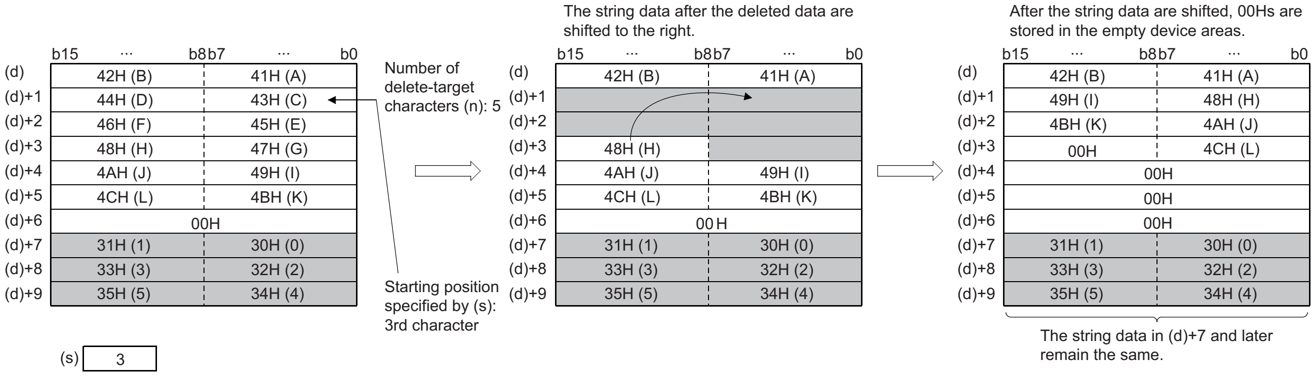
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the character string to be deleted	—	String	ANYSTRING_SINGLE
(s)	Deletion start position	1 to 16383	16-bit unsigned binary	ANY16
(n)	Number of characters to be deleted	0 to 16384-(s)	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s)	—	○	○	○	○	—	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions delete (n) characters starting from the position (deletion start position) specified by the (s)th character from the start of the character string data in the device specified by (d).



- If the character string after deletion in the device specified by (d) is even, the NULL code (00H) is stored in the device (1 word) next to the last one containing the character string.
- If the character string after deletion in the device specified by (d) is odd, the NULL code (00H) is stored in the last device (upper 8 bits) of the character string.
- The character string following the deleted one is shifted by (n) characters to the right, and the NULL code (00H) is stored in the device that has been emptied.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
3405H	The number of characters of the character string in the device specified by (d) exceeds 16383.
	An out-of-range value is set to (s). • The specified value is not within the following range. 1≤(s)≤16383 • The specified value exceeds the number of characters of the character string in the device specified by (d).
	An out-of-range value is set to (n). • The specified value exceeds the number of characters from the data in the device specified by (s) of the character string in the device specified by (d) to the last character. • The specified value is negative.

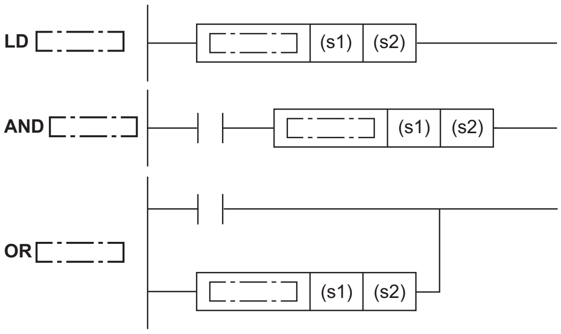
7.9 Real Number Instructions

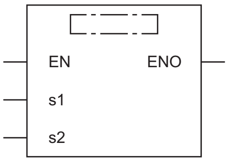
Comparing single-precision real numbers

LDE□, ANDE□, ORE□

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform a comparison operation of a single-precision real number. (Devices are used as a normally open contact.)

Ladder	ST[1]
 (□ is replaced by E=, E<>, E>, E<=, E<, E>=.)	ENO:=LDE_□(EN,s1,s2); ENO:=ANDE_□(EN,s1,s2); ENO:=ORE_□(EN,s1,s2); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[2]

FBD/LD
 (□ is replaced by a combination of LDE_, ANDE_, or ORE_ and EQ, NE, GT, LE, LT, or GE.)[2]

[1] The engineering tool with version "1.035M" or later supports the ST.

[2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDE□, ANDE□, ORE□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	0, $2^{-126} \leq \|(s1)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Comparison data or the start device where the comparison data is stored	0, $2^{-126} \leq \|(s2)\| < 2^{128}$	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—

Processing details

- These instructions perform a comparison operation between the single-precision real number in the device specified by (s1) and the single-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
E=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
E<>, NE	(s1)≠(s2)	
E>, GT	(s1)>(s2)	
E<=, LE	(s1)≤(s2)	
E<, LT	(s1)<(s2)	
E>=, GE	(s1)≥(s2)	
E=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
E<>, NE	(s1)=(s2)	
E>, GT	(s1)≤(s2)	
E<=, LE	(s1)>(s2)	
E<, LT	(s1)≥(s2)	
E>=, GE	(s1)<(s2)	

- If the data in the device specified by (s1) or (s2) is out of the range of setting data, the operation result will be non-continuity (ENO OFF).
- If the LDE_□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORE_□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.
- 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.

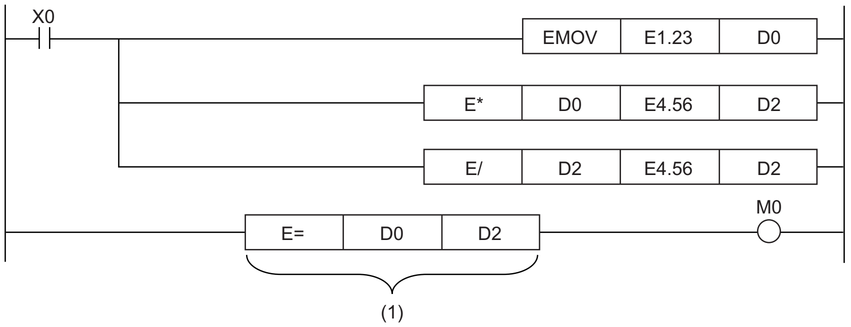
☞ Page 49 Precautions

Operation error

There is no operation error.

Point

Note that two values may not be equal due to an error when the E= instruction is used.



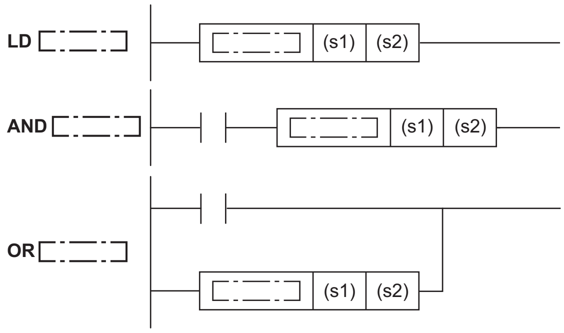
(1) Two values may not be equal.

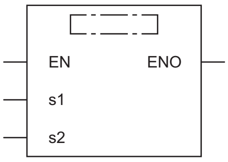
Comparing double-precision real numbers

LDED□, ANDED□, ORED□

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions perform a comparison operation of a double-precision real number. (Devices are used as a normally open contact.)

Ladder	ST*1
 (□ is replaced by ED=, ED<>, ED>, ED<=, ED<, ED>=.)	ENO:=LDED_□(EN,s1,s2); ENO:=ANDED_□(EN,s1,s2); ENO:=ORED_□(EN,s1,s2); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*2

FBD/LD
 (□ is replaced by a combination of LDED_, ANDED_, or ORED_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.

*2 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDED□, ANDED□, ORED□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	$0, 2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Comparison data or the start device where the comparison data is stored	$0, 2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—

Processing details

- These instructions perform a comparison operation between the double-precision real number in the device specified by (s1) and the double-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
ED=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
ED<>, NE	(s1)≠(s2)	
ED>, GT	(s1)>(s2)	
ED<=, LE	(s1)≤(s2)	
ED<, LT	(s1)<(s2)	
ED>=, GE	(s1)≥(s2)	
ED=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
ED<>, NE	(s1)=(s2)	
ED>, GT	(s1)≤(s2)	
ED<=, LE	(s1)>(s2)	
ED<, LT	(s1)≥(s2)	
ED>=, GE	(s1)<(s2)	

- If the data in the device specified by (s1) or (s2) is out of the range of setting data, the operation result will be non-continuity (ENO OFF).
- If the LDED_□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORED_□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.
- 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 49 Precautions

Operation error

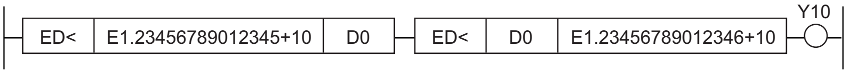
There is no operation error.

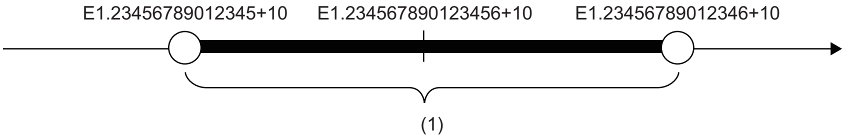
Precautions

The maximum number of digits of a real number which can be input using the engineering tool is 15, and therefore these instructions cannot perform comparison with a real number consisting of 16 or more effective digits. When these instructions are used to determine the match or mismatch with a real number consisting of 16 or more effective digits, the instructions need to compare the size with the approximate values before and after the real number to be compared.

Ex.

To determine the match between E1.234567890123456+10 (16 effective digits) and a double-precision real number

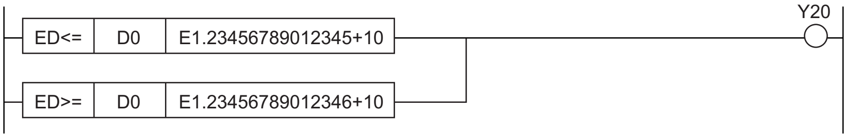


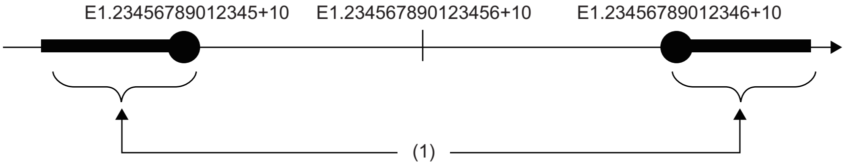


(1) Whether data in D0 to D3 are within this range is checked. (The boundary values are not included.)

Ex.

To determine the mismatch between E1.234567890123456+10 (16 effective digits) and a double-precision real number





(1) Whether data in D0 to D3 are within this range is checked. (The boundary values are included.)

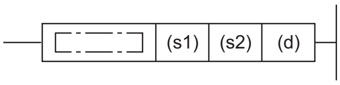
Outputting a comparison result of single-precision real numbers

ECMP(P)

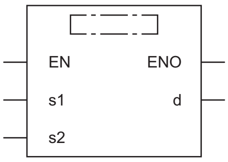
RnCPU RnENCPU ✕ RnPCPU (Process) ✕ RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

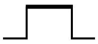
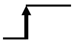
These instructions compare the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=ECMP(EN,s1,s2,d); ENO:=ECMPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ECMP	
ECMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device for storing the comparison data	0, $2^{-126}\leq\|(s1)\|<2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Comparison data or the start device for storing the comparison data	0, $2^{-126}\leq\|(s2)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

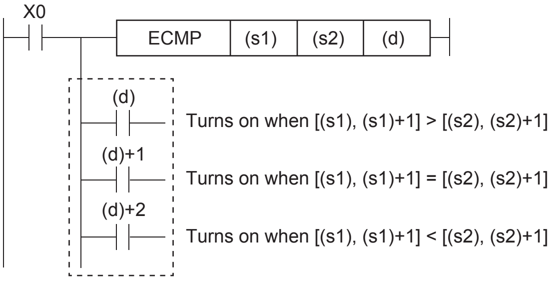
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	—	○	○	○	—	○	—	—
(s2)	—	—	○	○	—	○	○	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.



Operation error

Error code (SD0)	Description
3402H	The value input to (s1) and (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

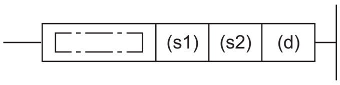
Outputting a comparison result of double-precision real numbers

EDCMP(P)

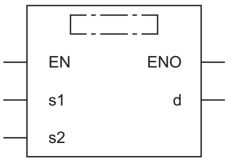
RnCPU **RnENCPU** ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

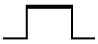
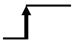
These instructions compare the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=EDCMP(EN,s1,s2,d); ENO:=EDCMPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDCMP	
EDCMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device for storing the comparison data	0, $2^{-1022} \leq \|(s1)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Comparison data or the start device for storing the comparison data	0, $2^{-1022} \leq \|(s2)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

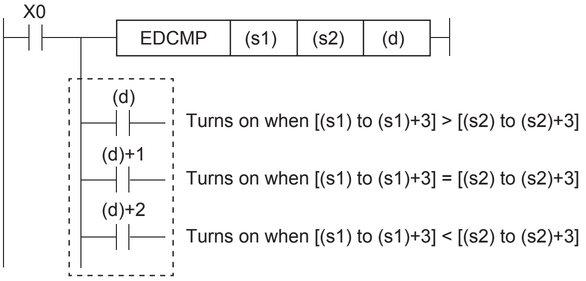
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○[1]	—	—	—	—	○	—	—	—	—

[1] T, ST, and C cannot be used.

Processing details

- These instructions compare the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.



Operation error

Error code (SD0)	Description
3402H	The value input to (s1) and (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

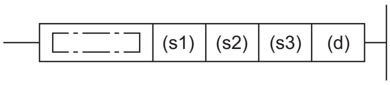
Outputting a band comparison result of single-precision real number

EZCP(P)

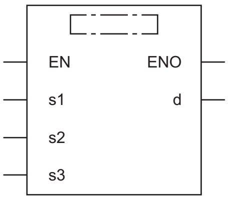
RnCPU **RnENCPU** ✗ **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

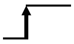
These instructions compare the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:= EZCP(EN, s1, s2, s3, d); ENO:= EZCPP(EN, s1, s2, s3, d);

FBD/LD



■Execution condition

Instruction	Execution condition
EZCP	
EZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Lower limit value or the start device for storing the lower limit value	0, $2^{-126}\leq\|(s1)\|<2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Upper limit value or the start device for storing the upper limit value	0, $2^{-126}\leq\|(s2)\|<2^{128}$	Single-precision real number	ANYREAL_32
(s3)	Comparison data or the start device for storing the comparison data	0, $2^{-126}\leq\|(s3)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

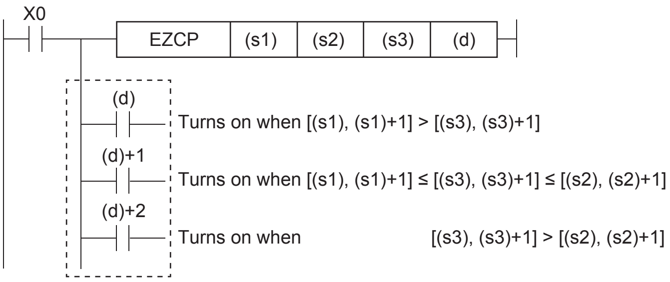
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	—	○	○	○	—	○	—	—
(s2)	—	—	○	○	—	○	○	○	—	○	—	—
(s3)	—	—	○	○	—	○	○	○	—	○	—	—
(d)	○	—	○[*1]	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

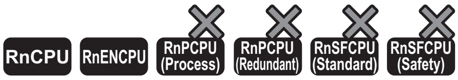
- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

Error code (SD0)	Description
3402H	The value input to (s1) to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.

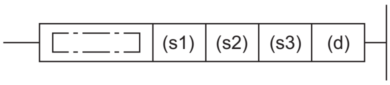
Outputting a band comparison result of double-precision real number

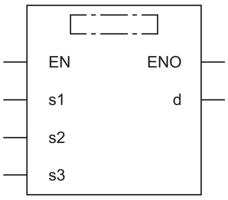
EDZCP(P)



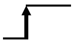
- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

These instructions compare the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:= EDZCP(EN, s1, s2, s3, d); ENO:= EDZCPP(EN, s1, s2, s3, d);

FBD/LD


■Execution condition

Instruction	Execution condition
EDZCP	
EDZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Lower limit value or the start device for storing the lower limit value	$0, 2^{-1022} \leq \|(s1)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Upper limit value or the start device for storing the upper limit value	$0, 2^{-1022} \leq \|(s2)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s3)	Comparison data or the start device for storing the comparison data	$0, 2^{-1022} \leq \|(s3)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

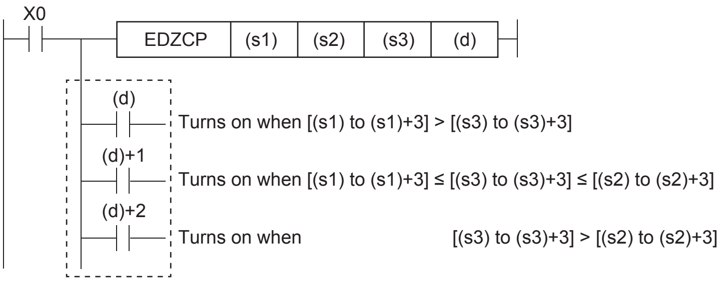
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(s3)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

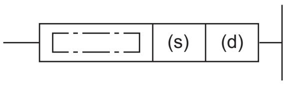
Error code (SD0)	Description
3402H	The value input to (s1) to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.

Adding single-precision real numbers

E+(P) [when two operands are set]

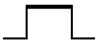
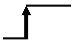
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 719 E+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 719 E+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E+	
E+P	

Setting data

■Description, range, data type

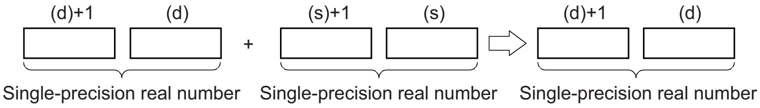
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0, $2^{-126}\leq\|(s)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where the first addend data is stored	0, $2^{-126}\leq\|(d)\|<2^{128}$	Single-precision real number	ANYREAL_32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, 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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the single-precision real number in the device specified by (s) to the single-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-126}\leq$|specified value (stored value)| $<2^{128}$ can be specified or stored in the devices specified by (s) 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 49 Precautions

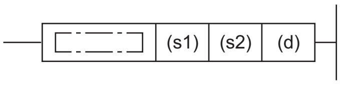
Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

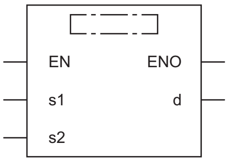
E+(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions add single-precision real numbers.

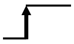
Ladder	ST
	ENO:=EPLUS(EN,s1,s2,d); ENO:=EPLUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: BPLUS, BPLUSP.)

■Execution condition

Instruction	Execution condition
E+	
E+P	

Setting data

■Description, range, data type

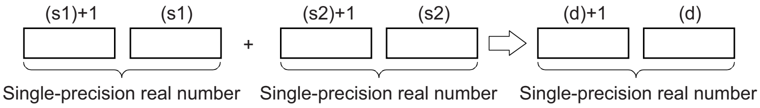
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	$0, 2^{-126} \le \|(s1)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Second addend data or the start device where the second addend data is stored	$0, 2^{-126}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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).



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

Operation error

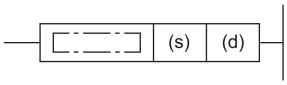
Error code (SD0)	**Description**
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

Subtracting single-precision real numbers

E-(P) [when two operands are set]

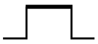
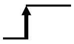
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions perform subtraction between single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 723 E-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 723 E-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E-	
E-P	

Setting data

■Description, range, data type

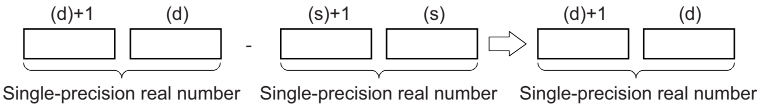
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	$0, 2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where minuend data is stored	$0, 2^{-126} \leq \|(d)\| < 2^{128}$	Single-precision real number	ANYREAL_32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, 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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions subtract the single-precision real number in the device specified by (s) from the single-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s) 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 49 Precautions

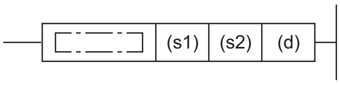
Operation error

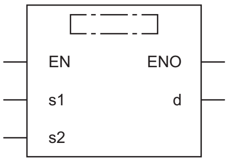
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

E-(P) [when three operands are set]

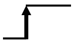
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions perform subtraction between single-precision real numbers.

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

FBD/LD

(□ is replaced by either of the following: BMINUS, BMINUSP.)

■Execution condition

Instruction	Execution condition
E-	
E-P	

Setting data

■Description, range, data type

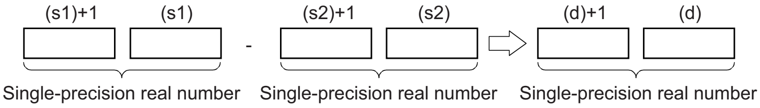
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	$0, 2^{-126} \le \|(s1)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Subtrahend data or the start device where subtrahend data is stored	$0, 2^{-126} \le \|(s2)\| < 2^{128}$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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq |\text{specified value (stored value)}| < 2^{128}$ 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 49 Precautions

Operation error

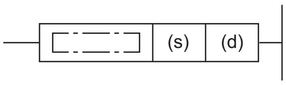
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

Adding double-precision real numbers

ED+(P) [when two operands are set]

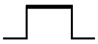
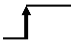
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 727 ED+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 727 ED+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED+	
ED+P	

Setting data

■Description, range, data type

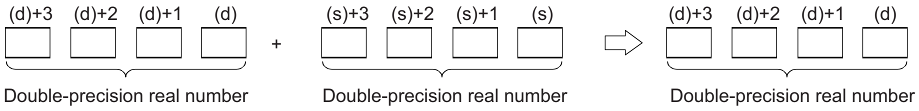
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	$0, 2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where the first addend data is stored	$0, 2^{-1022} \leq \|(d)\| < 2^{1024}$	Double-precision real number	ANYREAL_64

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions add the double-precision real number in the device specified by (d) to the double-precision real number in the device specified by (s), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s) 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 49 Precautions

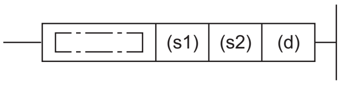
Operation error

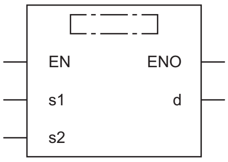
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

ED+(P) [when three operands are set]

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

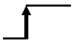
These instructions add double-precision real numbers.

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

FBD/LD


(□ is replaced by either of the following: EDPLUS, EDPLUSP.)

■Execution condition

Instruction	Execution condition
ED+	
ED+P	

Setting data

■Description, range, data type

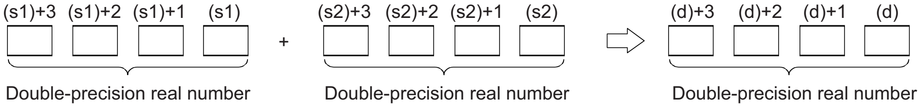
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0, $2^{-1022}\leq\|(s1)\|<2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Second addend data or the start device where the second addend data is stored	0, $2^{-1022}\leq\|(s2)\|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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions add 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).



- Value 0 or $2^{-1022} \leq |\text{specified value (stored value)}| < 2^{1024}$ 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 49 Precautions

Operation error

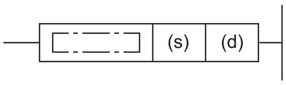
Error code (SD0)	**Description**
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Subtracting double-precision real numbers

ED-(P) [when two operands are set]

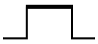
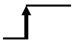
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ☒ **RnSFCPU (Safety)**

These instructions perform subtraction between double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 731 ED-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 731 ED-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED-	
ED-P	

Setting data

■Description, range, data type

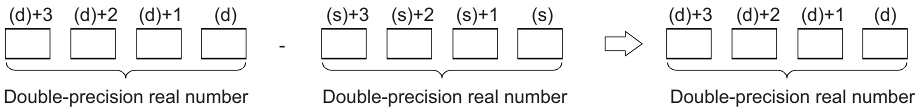
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	$0, 2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where minuend data is stored	$0, 2^{-1022} \leq \|(d)\| < 2^{1024}$	Double-precision real number	ANYREAL_64

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions subtract the double-precision real number in the device specified by (s) from the double-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $<2^{1024}$ can be specified or stored in the devices specified by (s) 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 49 Precautions

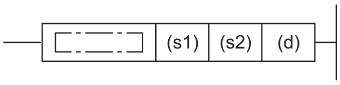
Operation error

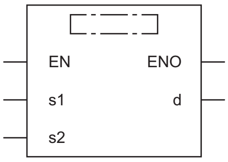
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

ED-(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

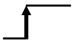
These instructions perform subtraction between double-precision real numbers.

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

FBD/LD


(□ is replaced by either of the following: EDMINUS, EDMINUSP.)

■Execution condition

Instruction	Execution condition
ED-	
ED-P	

Setting data

■Description, range, data type

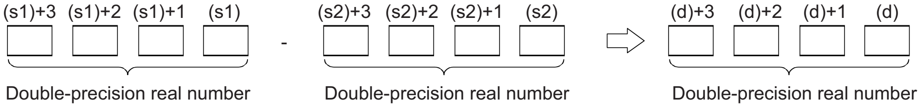
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-1022}\leq\|(s1)\|<2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022}\leq\|(s2)\|<2^{1024}$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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions subtract 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).



- Value 0 or $2^{-1022} \leq |\text{specified value (stored value)}| < 2^{1024}$ 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 49 Precautions

Operation error

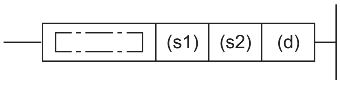
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Multiplying single-precision real numbers

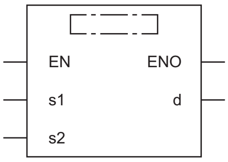
E*(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These instructions multiply single-precision real numbers.

Ladder	ST[*1]
	ENO:=EMULTI(EN,s1,s2,d); ENO:=EMULTIP(EN,s1,s2,d);

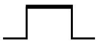
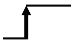
FBD/LD



(□ is replaced by either of the following: EMULTI, EMULTIP.)

[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
E*	
E*P	

Setting data

■Description, range, data type

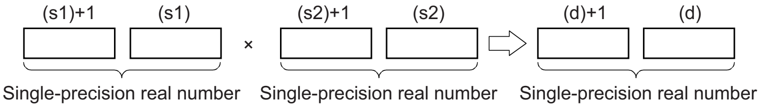
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-126}\leq$|(s1)|$<2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-126}\leq$|(s2)|$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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions multiply 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).



- Value 0 or $2^{-126} \leq |\text{specified value (stored value)}| < 2^{128}$ 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 49 Precautions

Operation error

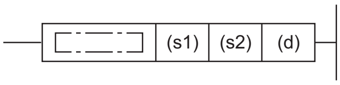
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert (d) \rvert < 2^{128}$

Dividing single-precision real numbers

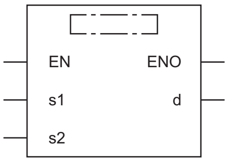
E/(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗RnSFCPU (Safety)

These instructions perform division between single-precision real numbers.

Ladder	ST[*1]
	ENO:=EDIVISION(EN,s1,s2,d); ENO:=EDIVISIONP(EN,s1,s2,d);

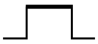
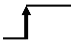
FBD/LD



(□ is replaced by either of the following: EDIVISION, EDIVISIONP.)

[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
E/	
E/P	

Setting data

■Description, range, data type

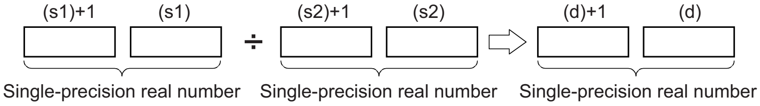
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	$0, 2^{-126} \leq \|(s1)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Divisor data or the start device where divisor data is stored	$0, 2^{-126} \leq \|(s2)\| < 2^{128}$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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the division result in the device specified by (d).



- Value 0 or $2^{-126} \leq |\text{specified value (stored value)}| < 2^{128}$ 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 49 Precautions

Operation error

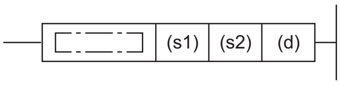
Error code (SD0)	Description
3400H	The data (divisor) in the device specified by (s2) is 0.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Multiplying double-precision real numbers

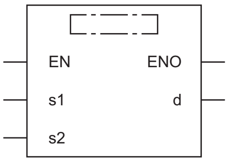
ED*(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions multiply double-precision real numbers.

Ladder	ST[*1]
	ENO:=EDMULTI(EN,s1,s2,d); ENO:=EDMULTIP(EN,s1,s2,d);

FBD/LD

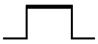
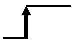


(□ is replaced by either of the following: EDMULTI, EDMULTIP.)

---

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
ED*	
ED*P	

Setting data

■Description, range, data type

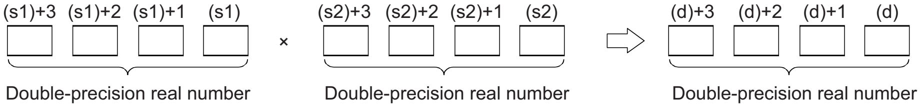
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-1022}\leq\|(s1)\|<2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-1022}\leq\|(s2)\|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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions multiply 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).



- Value 0 or $2^{-1022} \leq |\text{specified value (stored value)}| < 2^{1024}$ 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 49 Precautions

Operation error

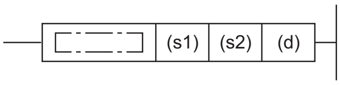
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)| < 2^{1024}$

Dividing double-precision real numbers

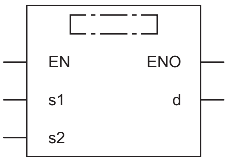
ED/(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform division between double-precision real numbers.

Ladder	ST[*1]
	ENO:=EDDIVISION(EN,s1,s2,d); ENO:=EDDIVISIONP(EN,s1,s2,d);

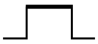
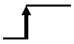
FBD/LD



(□ is replaced by either of the following: EDDIVISION, EDDIVISIONP.)

[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
ED/	
ED/P	

Setting data

■Description, range, data type

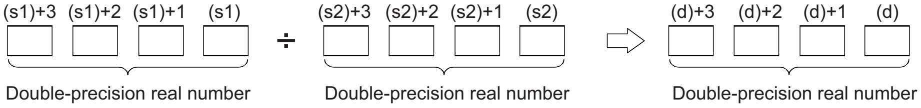
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-1022} \leq \|(s1)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-1022} \leq \|(s2)\| < 2^{1024}$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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions divide 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).



- Value 0 or $2^{-1022} \leq |\text{specified value (stored value)}| < 2^{1024}$ 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 49 Precautions

Operation error

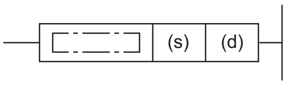
Error code (SD0)	Description
3400H	The data (divisor) in the device specified by (s2) is 0.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{1024}$

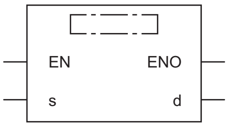
Converting 16-bit signed binary data to single-precision real number

INT2FLT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

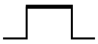
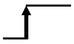
These instructions convert 16-bit signed binary data to a single-precision real number.

Ladder	ST[*1]
	ENO:=INT2FLT(EN,s,d); ENO:=INT2FLTP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2FLT	
INT2FLTP	

Setting data

■Description, range, data type

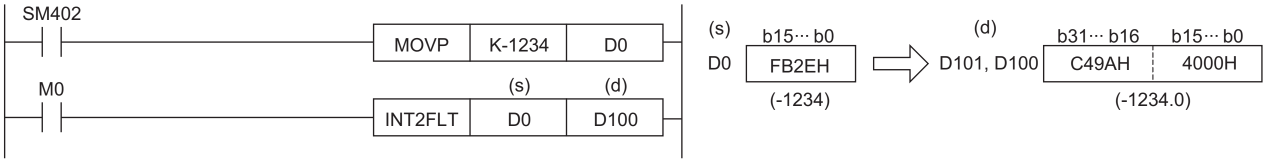
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to a single-precision real number, and store the converted data in the device specified by (d).



Operation error

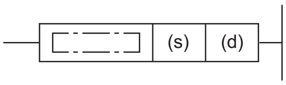
There is no operation error.

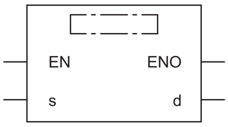
Converting 16-bit unsigned binary data to single-precision real number

UINT2FLT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

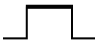
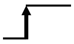
These instructions convert 16-bit unsigned binary data to a single-precision real number.

Ladder	ST[*1]
	ENO:=UINT2FLT(EN,s,d); ENO:=UINT2FLTP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2FLT	
UINT2FLTP	

Setting data

■Description, range, data type

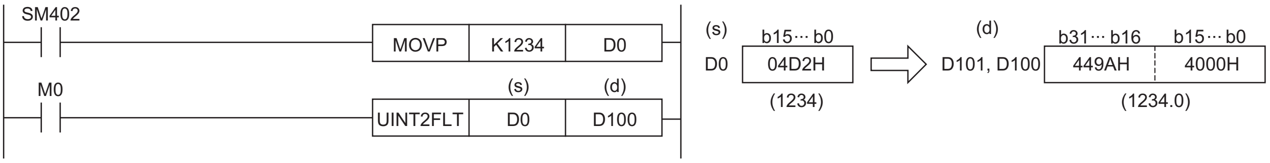
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to a single-precision real number, and store the real number in the device specified by (d).



Operation error

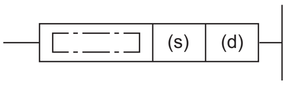
There is no operation error.

Converting 32-bit signed binary data to single-precision real number

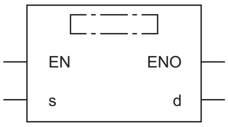
DINT2FLT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions convert 32-bit signed binary data to a single-precision real number.

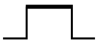
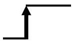
Ladder	ST[*1]
 (s) (d)	ENO:=DINT2FLT(EN,s,d); ENO:=DINT2FLTP(EN,s,d);

FBD/LD



[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2FLT	
DINT2FLTP	

Setting data

■Description, range, data type

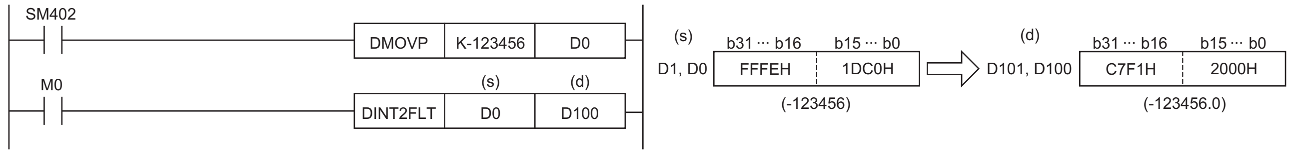
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

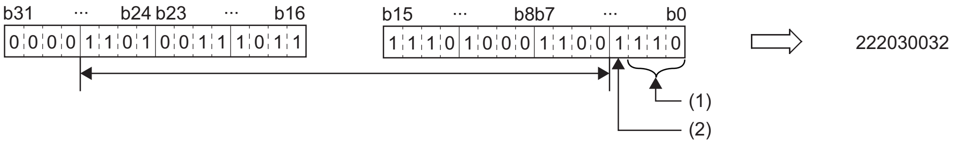
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

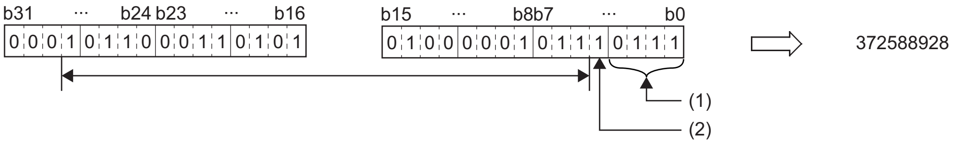
Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).



- A single-precision real number is processed in 32-bit single precision, and therefore the effective number of digits is 24 bits when it is represented in binary and is about 7 digits when represented in decimal. For this reason, if the integer value exceeds the range from -16777216 to 16777215 (24-bit binary value), an error occurs in the converted value. The operation result is an integer value in which the 25th bit from upper bits is rounded off.





(1) Rounded down.

(2) Rounded off.

Operation error

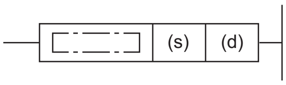
There is no operation error.

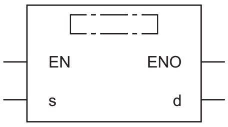
Converting 32-bit unsigned binary data to single-precision real number

UDINT2FLT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

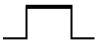
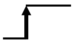
These instructions convert 32-bit unsigned binary data to a single-precision real number.

Ladder	ST[*1]
 (s) (d)	ENO:=UDINT2FLT(EN,s,d); ENO:=UDINT2FLTP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2FLT	
UDINT2FLTP	

Setting data

■Description, range, data type

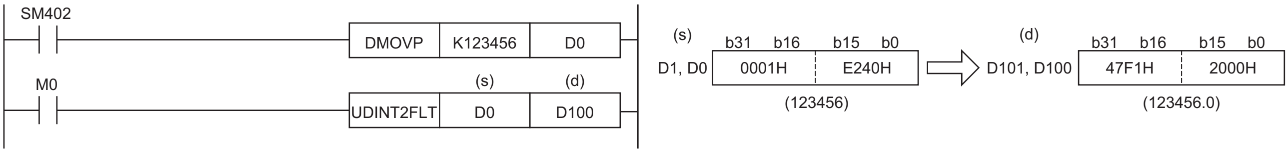
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32

■Applicable devices

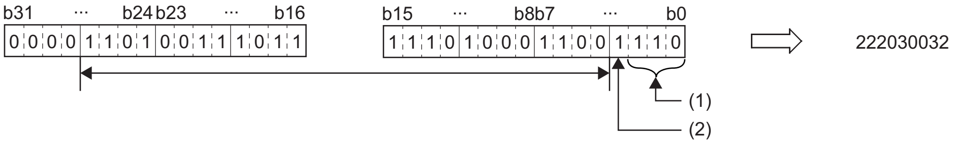
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

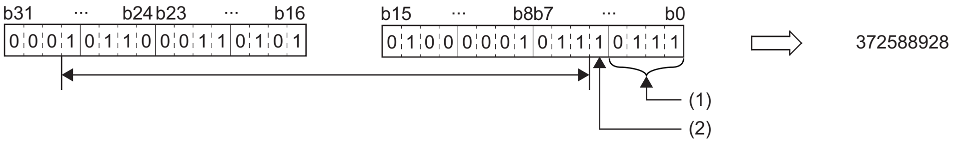
Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).



- A single-precision real number is processed in 32-bit single precision, and therefore the effective number of digits is 24 bits when it is represented in binary and is about 7 digits when represented in decimal. For this reason, if the integer value exceeds the range from 0 to 16777215 (24-bit binary value), an error occurs in the converted value. The operation result is an integer value in which the 25th bit from upper bits is rounded off.





(1) Rounded down.

(2) Rounded off.

Operation error

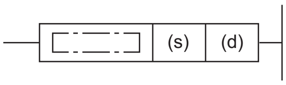
There is no operation error.

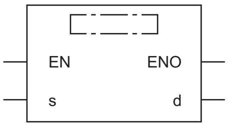
Converting double-precision real number to single-precision real number

DBL2FLT(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

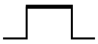
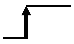
These instructions convert a double-precision real number to a single-precision real number.

Ladder	ST[*1]
	ENO:=DBL2FLT(EN,s,d); ENO:=DBL2FLTP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2FLT	
DBL2FLTP	

Setting data

■Description, range, data type

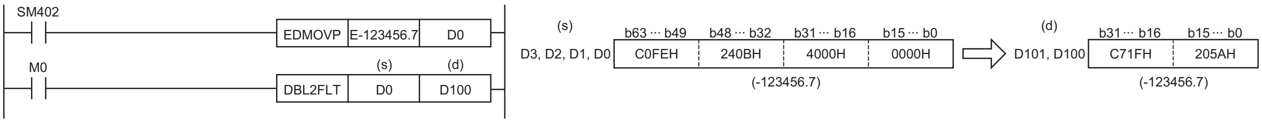
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-1022} \le \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the double-precision real number in the device specified by (s) to a single-precision real number, and store the real number in the device specified by (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 49 Precautions

Operation error

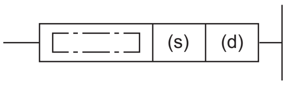
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

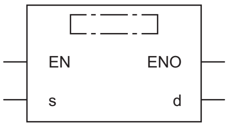
Converting 16-bit signed binary data to double-precision real number

INT2DBL(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

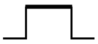
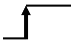
These instructions convert 16-bit signed binary data to a double-precision real number.

Ladder	ST[*1]
	ENO:=INT2DBL(EN,s,d); ENO:=INT2DBLP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2DBL	
INT2DBLP	

Setting data

■Description, range, data type

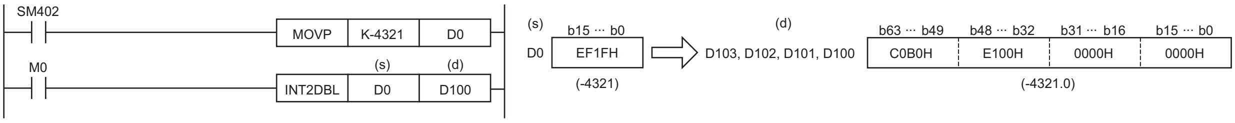
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



Operation error

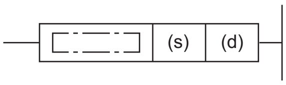
There is no operation error.

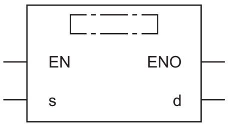
Converting 16-bit unsigned binary data to double-precision real number

UINT2DBL(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

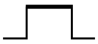
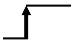
These instructions convert 16-bit unsigned binary data to a double-precision real number.

Ladder	ST[*1]
	ENO:=UINT2DBL(EN,s,d); ENO:=UINT2DBLP(EN,s,d);

FBD/LD


*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2DBL	
UINT2DBLP	

Setting data

■Description, range, data type

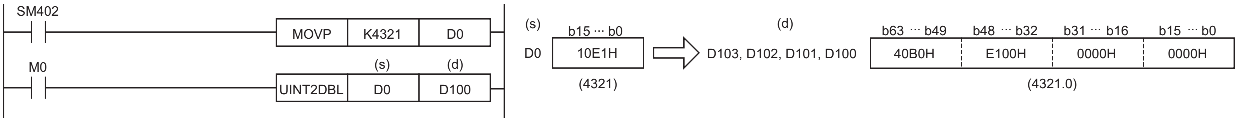
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



Operation error

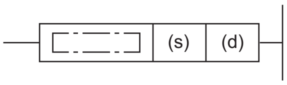
There is no operation error.

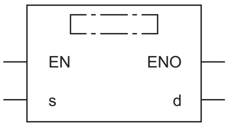
Converting 32-bit signed binary data to double-precision real number

DINT2DBL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

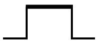
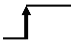
These instructions convert 32-bit signed binary data to a double-precision real number.

Ladder	ST[*1]
	ENO:=DINT2DBL(EN,s,d); ENO:=DINT2DBLP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2DBL	
DINT2DBLP	

Setting data

■Description, range, data type

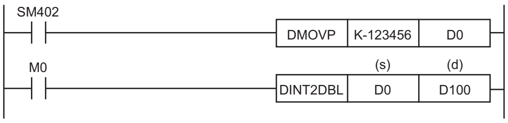
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

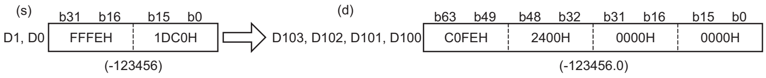
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).





Operation error

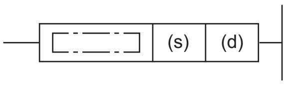
There is no operation error.

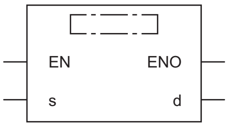
Converting 32-bit unsigned binary data to double-precision real number

UDINT2DBL(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

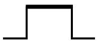
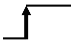
These instructions convert 32-bit unsigned binary data to a double-precision real number.

Ladder	ST[*1]
	ENO:=UDINT2DBL(EN,s,d); ENO:=UDINT2DBLP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2DBL	
UDINT2DBLP	

Setting data

■Description, range, data type

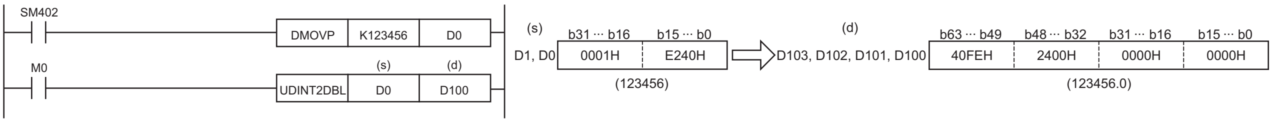
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



Operation error

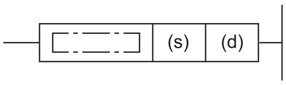
There is no operation error.

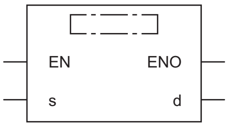
Converting single-precision real number to double-precision real number

FLT2DBL(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✗

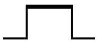
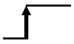
These instructions convert a single-precision real number to a double-precision real number.

Ladder	ST[*1]
	ENO:=FLT2DBL(EN,s,d); ENO:=FLT2DBLP(EN,s,d);

FBD/LD


[*1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2DBL	
FLT2DBLP	

Setting data

■Description, range, data type

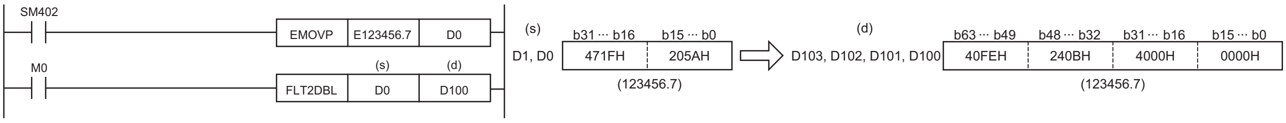
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	○	○	○	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

These instructions convert the single-precision real number in the device specified by (s) to a double-precision real number, and store the double-precision real number in the device specified by (d).



Operation error

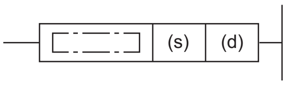
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

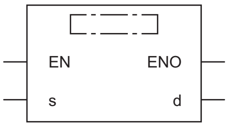
Converting string data to single-precision real number

EVAL(P)

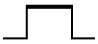
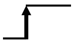
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions convert a string to a single-precision real number.

Ladder	ST
	ENO:=EVAL(EN,s,d); ENO:=EVALP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
EVAL	
EVALP	

Setting data

■Descriptions, ranges, and data types

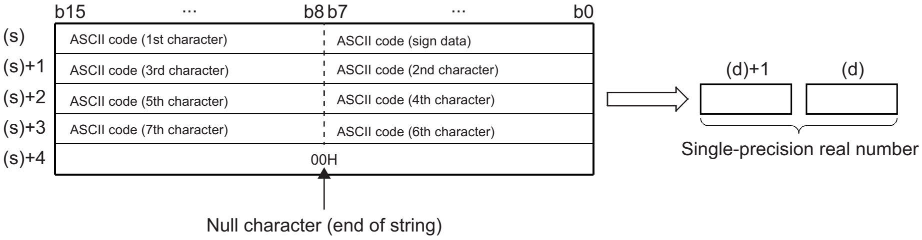
Operand	Description	Range	Data type	Data type (label)
(s)	Character string data to be converted into single-precision real number data, or the start device containing the character string data	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted single-precision real number data	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

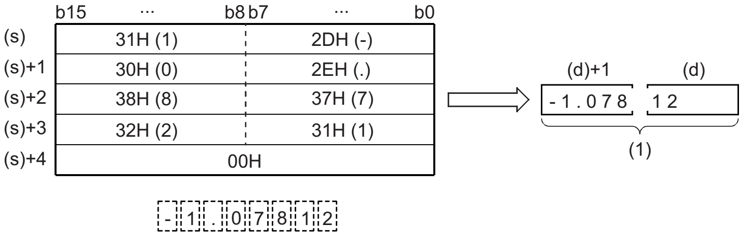
- These instructions convert the character string stored in the device number specified by (s) and later to single-precision real number data, and store the converted real number in the device specified by (d).
- The specified character string can be converted to a single-precision real number in either decimal point or exponential format.



- "20H" (space) that exists in the middle of the data is ignored.
- The character string can consist of up to 24 characters. "20H" (space) and "30H" (0) are counted as one character as well.

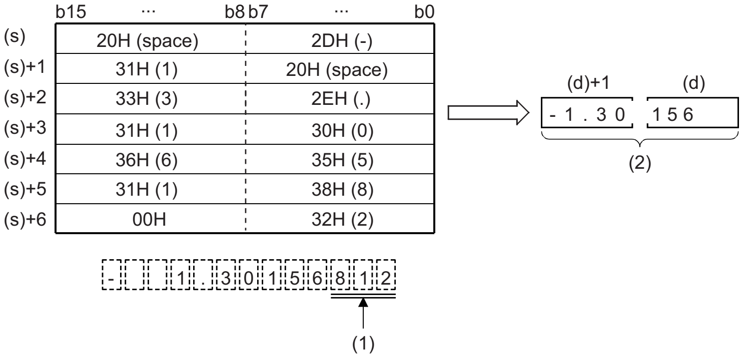
■Decimal point format

- When the character string in the device specified by (s) is in decimal point format, conversion is performed as shown below.



(1) Single-precision real number

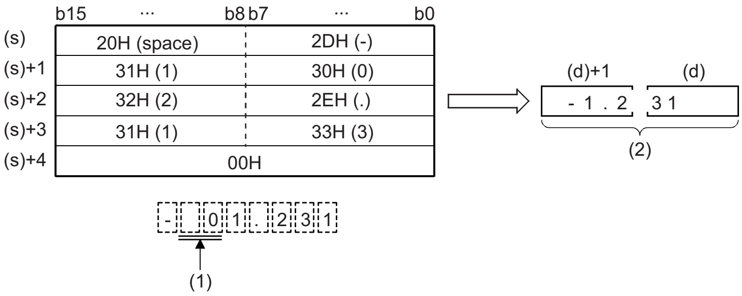
- The character string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(1) Rounded down.

(2) Single-precision real number

- If 2BH(+) is specified for the sign or the sign is omitted in decimal point format, the converted single-precision real number is treated as a positive value. If 2DH(-) is specified for the sign, it is treated as a negative value.
- If 20H (space) or 30H (0) exists between digits excluding the first 0 in the character string in the device specified by (s), the instruction performs conversion by ignoring 20H and 30H.

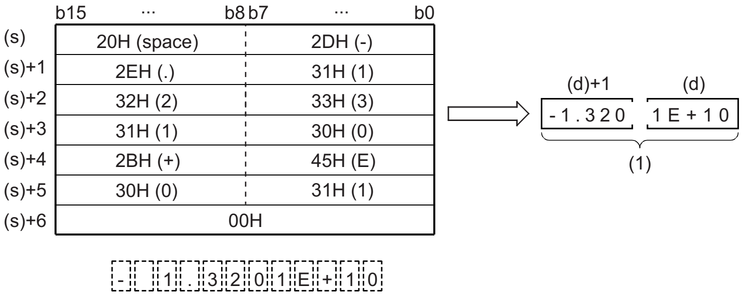


(1) Ignored.

(2) Single-precision real number

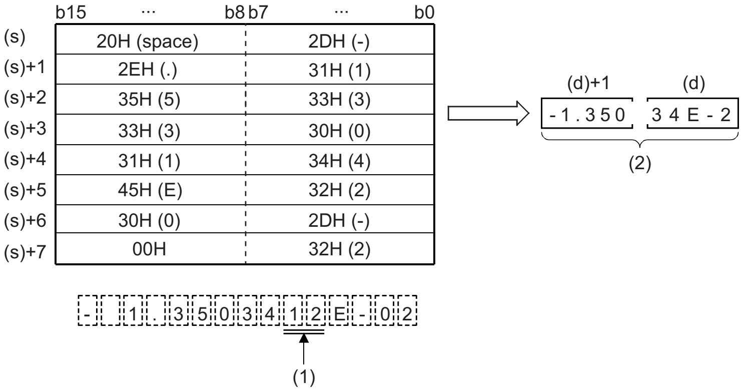
■Exponential format

- When the character string in the device specified by (s) is in exponential format, conversion is performed as shown below.



(1) Single-precision real number

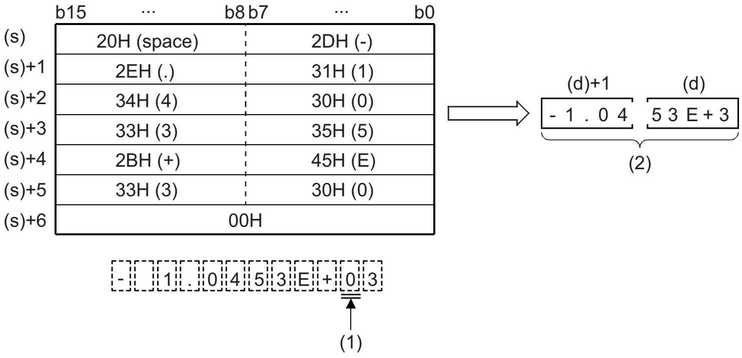
- The character string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(1) Rounded down.

(2) Single-precision real number

- If 2BH(+) is specified for the sign or the sign is omitted in exponential format, the converted single-precision real number is treated as a positive value. If 2DH(-) is specified for the sign in the exponent, it is treated as a negative value.
- If 20H (space) or 30H (0) exists between digits excluding the first 0 in the character string in the device specified by (s), the instruction performs conversion by ignoring 20H and 30H.
- If 30H (0) exists between "E" and a numerical value in the character string in exponential format, the instruction performs conversion by ignoring 30H.



	b15 ··· b8	b7 ··· b0
(s)	20H (space)	2DH (-)
(s)+1	2EH (.)	31H (1)
(s)+2	34H (4)	30H (0)
(s)+3	33H (3)	35H (5)
(s)+4	2BH (+)	45H (E)
(s)+5	33H (3)	30H (0)
(s)+6	00H	

(1) Ignored.

(2) Single-precision real number

Operation error

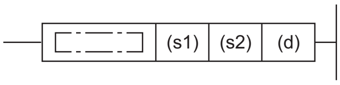
Error code (SD0)	Description
2820H	00H does not exist within the range of the relevant device specified by (s).
3401H	An out-of-range value is set to (s). • The integral part or decimal part contains a character other than 30H (0) to 39H (9). • The specified string contains two or more 2EH (.) • The exponent of the specified string contains a character other than 45H (E), 65H (e), 2BH (+), and 2DH (-). • The specified string contains more than one exponent 45H (E) or 65H (e). • The exponent in the specified string contains a numerical value consisting of three digits or more. • The exponent of the specified string contains more than one sign 2BH (+) or 2DH (-). • The specified string contains more than one sign 2BH (+) or 2DH (-) in the integral part of decimal point format or in the mantissa of exponential format. • The number of characters in the device specified by (s) and later is 0 or exceeds 24.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

Converting BCD format data to single-precision real number

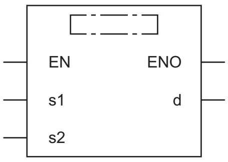
EREXP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

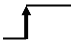
These instructions convert BCD floating point format data to single-precision real number data in accordance with the specified number of digits in the decimal part.

Ladder	ST
	ENO:=EREXP(EN,s1,s2,d); ENO:=EREXPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EREXP	
EREXPP	

Setting data

■Descriptions, ranges, and data types

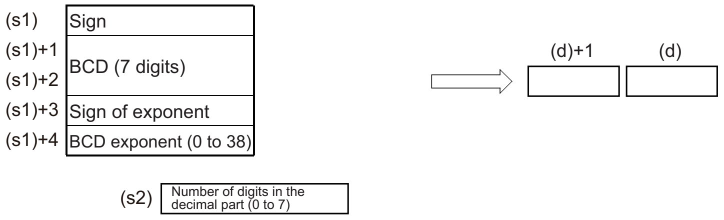
Operand	Description	Range	Data type	Data type (label)
(s1)	Start device containing BCD floating point format data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Number of digits in the decimal part	0 to 7	16-bit signed binary	ANY16
(d)	Start device for storing a single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the BCD floating-point format data stored in the device specified by (s1) and later to single-precision real number data in accordance with the number of decimal places stored in the device specified by (s2), and store the converted data in the device number specified by (d) and later.



(s1): Sign (Positive: 0, Negative: 1)

(s1)+1, (s1)+2: BCD (7 digits)

(s1)+3: Sign of exponent (Positive: 0, Negative: 1)

(s1)+4: BCD exponent (0 to 38)

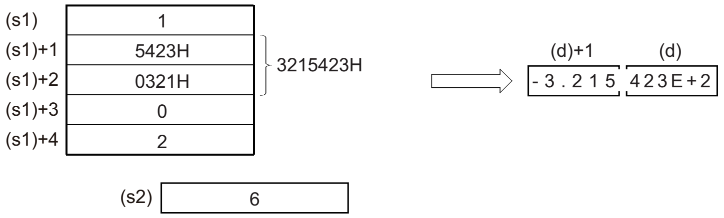
(s2): Number of digits in the decimal part (0 to 7)

(d)+1, (d): Single-precision real number

- For the sign in (s1) and exponent sign in (s1)+3, 0 is set for positive and 1 is set for negative.
- A value of 0 to 38 can be set for the BCD exponent in (s1)+4.
- A value of 0 to 7 can be set for the number of decimal part digits in the device specified by (s2).

Ex.

When 6 is specified in (s2)



Operation error

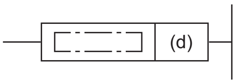
Error code (SD0)	Description
3401H	The value set to (s1) as sign data is not 0 or 1.
	A value other than 0 to 9 exists at any digit of data set to (s1)+1 and (s1)+2.
	The value set to (s1)+3 as sign data of exponent is not 0 or 1.
	The exponent data set to (s1)+4 is out of the range, 0 to 38.
	The value set to (s2) as the number of digits in the decimal part is out of the range, 0 to 7.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Inverting the sign of single-precision real number

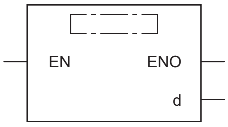
ENEG(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

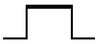
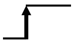
These instructions invert the sign of single-precision real number data.

Ladder	ST
 (d)	ENO:=ENEG(EN,d); ENO:=ENEGP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ENEG	
ENEGP	

Setting data

■Description, range, data type

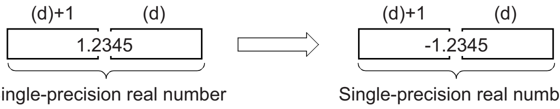
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the single-precision real number data subject to sign inversion	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions invert 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

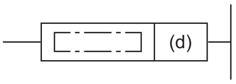
Error code (SD0)	Description
3402H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

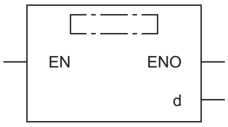
Inverting the sign of double-precision real number

EDNEG(P)

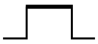
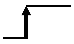
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions invert the sign of double-precision real number data.

Ladder	ST
	ENO:=EDNEG(EN,d); ENO:=EDNEGP(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
EDNEG	
EDNEGP	

Setting data

■Description, range, data type

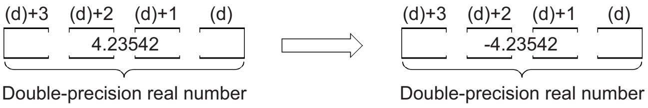
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the double-precision real number subject to sign inversion	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions invert 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

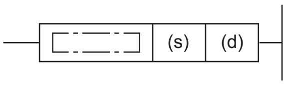
Error code (SD0)	Description
3402H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

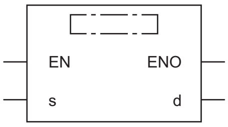
Transferring single-precision real number

EMOV(P)

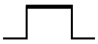
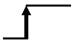
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions transfer single-precision real number data to the specified device.

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

FBD/LD


■Execution condition

Instruction	Execution condition
EMOV	
EMOVP	

Setting data

■Description, range, data type

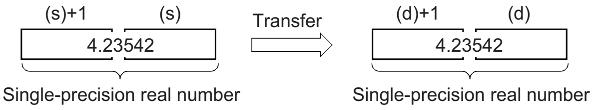
Operand	Description	Range	Data type	Data type (label)
(s)	Data to be transferred or start device containing the data to be transferred	$0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing transferred data	—	Single-precision real number	ANYREAL_32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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions transfer the single-precision real number data stored in the device specified by (s) to the device specified by (d).



Operation error

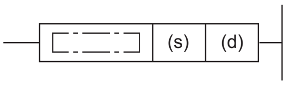
There is no operation error.

Transferring double-precision real number

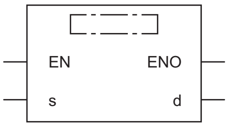
EDMOV(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

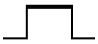
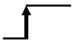
These instructions transfer double-precision real number data to the specified device.

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

FBD/LD



■Execution condition

Instruction	Execution condition
EDMOV	
EDMOVP	

Setting data

■Description, range, data type

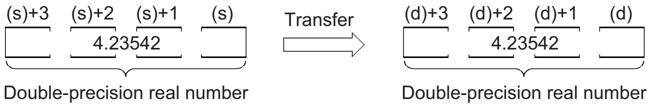
Operand	Description	Range	Data type	Data type (label)
(s)	Data to be transferred or start device containing the data to be transferred	0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing transferred data	—	Double-precision real number	ANYREAL_64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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the double-precision real number data stored in the device specified by (s) to the device specified by (d).



Operation error

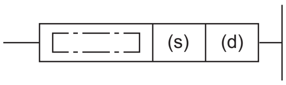
There is no operation error.

Calculating the sine of single-precision real number

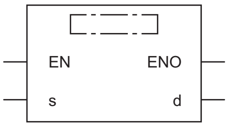
SIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)✕

These instructions calculate the sine of the angle specified by a single-precision real number.

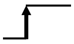
Ladder	ST[*1]
	ENO:=SINP(EN,s,d);

FBD/LD[*1]



*1 The SIN instruction does not support the ST and FBD/LD. Use the standard function, SIN.
☞ Page 1900 SIN(_E)

■Execution condition

Instruction	Execution condition
SIN	
SINP	

Setting data

■Description, range, data type

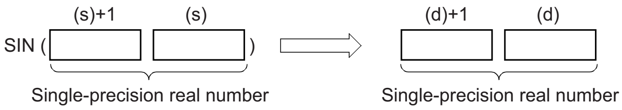
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	$0, 2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

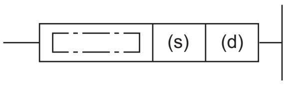
☞ Page 809 DEG(P)

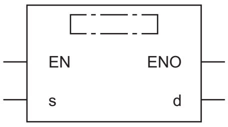
Calculating the cosine of single-precision real number

COS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

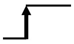
These instructions calculate the cosine of the angle specified by a single-precision real number.

Ladder	ST[1]
	ENO:=COSP(EN,s,d);

FBD/LD[1]


[1] The COS instruction does not support the ST and FBD/LD. Use the standard function, COS.
☞ Page 1901 COS(_E)

■Execution condition

Instruction	Execution condition
COS	
COSP	

Setting data

■Description, range, data type

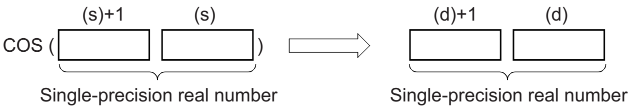
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	0, $2^{-126}\leq\|(s)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

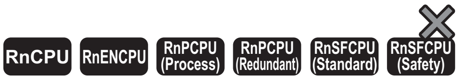
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

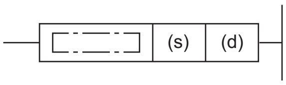
☞ Page 809 DEG(P)

Calculating the tangent of single-precision real number

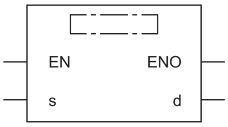
TAN(P)



These instructions calculate the tangent of the angle specified by a single-precision real number.

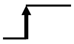
Ladder	ST[1]
	ENO:=TANP(EN,s,d);

FBD/LD[1]



[1] The TAN instruction does not support the ST and FBD/LD. Use the standard function, TAN.
☞ Page 1902 TAN(_E)

■Execution condition

Instruction	Execution condition
TAN	
TANP	

Setting data

■Description, range, data type

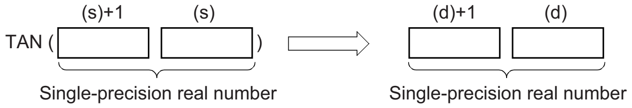
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	$0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Precautions

If the angle specified by (s) is π/2 radian or (3/2)π radian, no operation error will be issued because of the truncation error in the radian value.

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

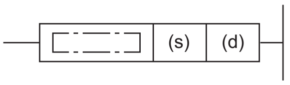
☞ Page 809 DEG(P)

Calculating the arc sine of single-precision real number

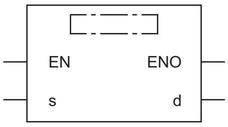
ASIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions calculate the angle from the sine specified by a single-precision real number.

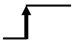
Ladder	ST[*1]
	ENO:=ASINP(EN,s,d);

FBD/LD[*1]



[*1] The ASIN instruction does not support the ST and FBD/LD. Use the standard function, ASIN.
☞ Page 1903 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIN	
ASINP	

Setting data

■Description, range, data type

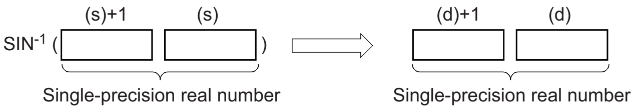
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the sine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The sine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

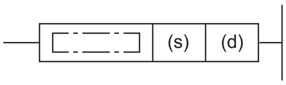
☞ Page 809 DEG(P)

Calculating the arc cosine of single-precision real number

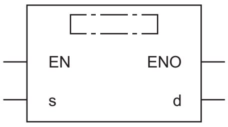
ACOS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions calculate the angle from the cosine specified by a single-precision real number.

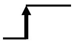
Ladder	ST[*1]
	ENO:=ACOSP(EN,s,d)

FBD/LD[*1]



[*1] The ACOS instruction does not support the ST and FBD/LD. Use the standard function, ACOS.
☞ Page 1904 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOS	
ACOSP	

Setting data

■Description, range, data type

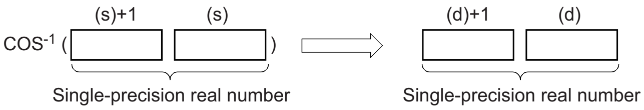
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the cosine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The cosine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

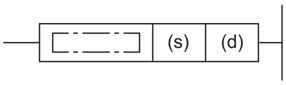
☞ Page 809 DEG(P)

Calculating the arc tangent of single-precision real number

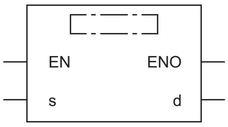
ATAN(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions calculate the angle from the tangent specified by a single-precision real number.

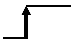
Ladder	ST[1]
	ENO:=ATANP(EN,s,d);

FBD/LD[1]



[1] The ATAN instruction does not support the ST and FBD/LD. Use the standard function, ATAN.
☞ Page 1905 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAN	
ATANP	

Setting data

■Description, range, data type

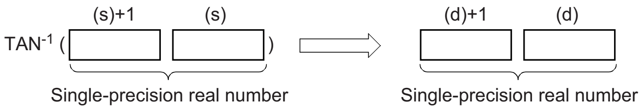
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	0, $2^{-126}\leq\|(s)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the tangent data in the device specified by (s), and store the operation result in the device number specified by (d).



- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 807 RAD(P)

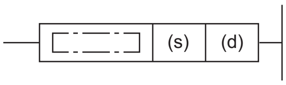
☞ Page 809 DEG(P)

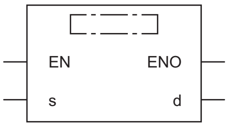
Calculating the sine of double-precision real number

SIND(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

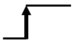
These instructions calculate the sine of the angle specified by a double-precision real number.

Ladder	ST[1]
	ENO:=SINDP(EN,s,d);

FBD/LD[1]


[1] The SIND instruction does not support the ST and FBD/LD. Use the standard function, SIN.
☞ Page 1900 SIN(_E)

■Execution condition

Instruction	Execution condition
SIND	
SINDP	

Setting data

■Description, range, data type

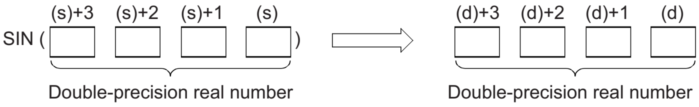
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	$0, 2^{-1022} \le \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

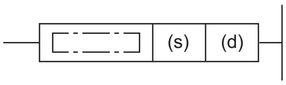
☞ Page 813 DEGD(P)

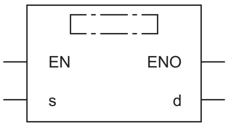
Calculating the cosine of double-precision real number

COSD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

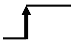
These instructions calculate the cosine of the angle specified by a double-precision real number.

Ladder	ST[*1]
	ENO:=COSDP(EN,s,d);

FBD/LD[*1]


*1 The COSD instruction does not support the ST and FBD/LD. Use the standard function, COS.
☞ Page 1901 COS(_E)

■Execution condition

Instruction	Execution condition
COSD	
COSDP	

Setting data

■Descriptions, ranges, and data types

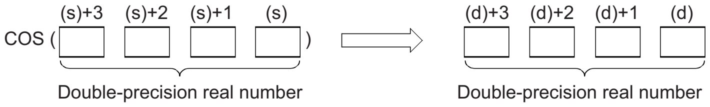
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	$0, 2^{-1022} \leq (s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

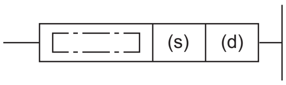
☞ Page 813 DEGD(P)

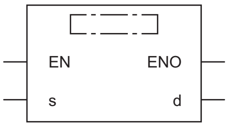
Calculating the tangent of double-precision real number

TAND(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

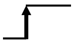
These instructions calculate the tangent of the angle specified by a double-precision real number.

Ladder	ST[*1]
	ENO:=TANDP(EN,s,d);

FBD/LD[*1]


*1 The TAND instruction does not support the ST and FBD/LD. Use the standard function, TAN.
☞ Page 1902 TAN(_E)

■Execution condition

Instruction	Execution condition
TAND	
TANDP	

Setting data

■Description, range, data type

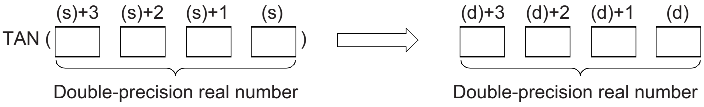
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	$0, 2^{-1022} \le \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- 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 49 Precautions

Precautions

If the angle specified by (s) is π/2 radian or (3/2)π radian, no operation error will be issued because of the truncation error in the radian value.

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

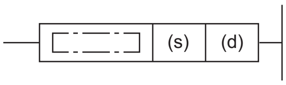
☞ Page 813 DEGD(P)

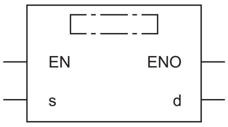
Calculating the arc sine of double-precision real number

ASIND(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

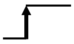
These instructions calculate the angle from the sine specified by a double-precision real number.

Ladder	ST[1]
	ENO:=ASINDP(EN,s,d)

FBD/LD[1]


[1] The ASIND instruction does not support the ST and FBD/LD. Use the standard function, ASIN.
☞ Page 1903 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIND	
ASINDP	

Setting data

■Description, range, data type

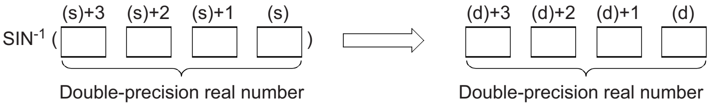
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the sine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The sine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

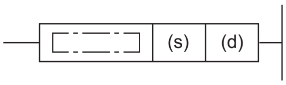
☞ Page 813 DEGD(P)

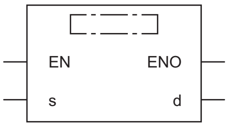
Calculating the arc cosine of double-precision real number

ACOSD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

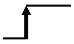
These instructions calculate the angle from the cosine specified by a double-precision real number.

Ladder	ST[1]
	ENO:=ACOSDP(EN,s,d);

FBD/LD[1]


[1] The ACOSD instruction does not support the ST and FBD/LD. Use the standard function, ACOS.
☞ Page 1904 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOSD	
ACOSDP	

Setting data

■Description, range, data type

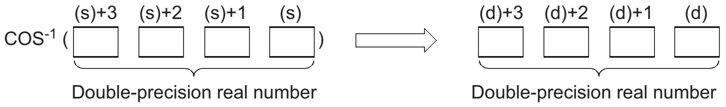
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the cosine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The cosine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

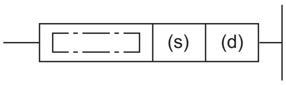
☞ Page 813 DEGD(P)

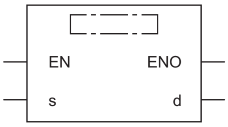
Calculating the arc tangent of double-precision real number

ATAND(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

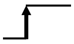
These instructions calculate the angle from the tangent specified by a double-precision real number.

Ladder	ST[*1]
	ENO:=ATANDP(EN,s,d);

FBD/LD[*1]


*1 The ATAND instruction does not support the ST and FBD/LD. Use the standard function, ATAN.
☞ Page 1905 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAND	
ATANDP	

Setting data

■Description, range, data type

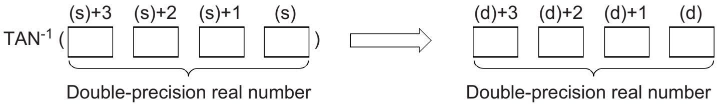
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	$0, 2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the tangent data in the device specified by (s), and store the operation result in the device number specified by (d).



- The angle (operation result) is stored in radians in the device specified by (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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 811 RADD(P)

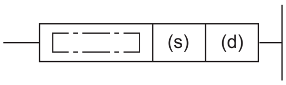
☞ Page 813 DEGD(P)

Calculating the sine of BCD data

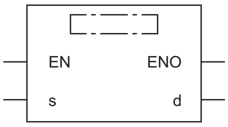
BSIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

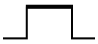
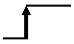
These instructions calculate the sine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BSIN(EN,s,d); ENO:=BSINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSIN	
BSINP	

Setting data

■Descriptions, ranges, and data types

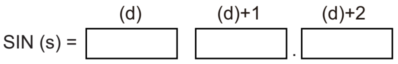
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for sine calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the sign of the operation result in the device specified by (d) and the operation result in the devices specified by (d)+1 and (d)+2.



(d): Sign

(d)+1: Integral part

(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360°(in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -1.000 to 1.000.
- The operation result is a value whose 5th decimal place is rounded off.

Operation error

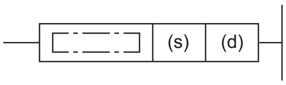
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

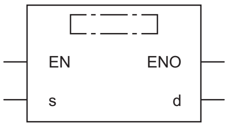
Calculating the cosine of BCD data

BCOS(P)

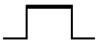
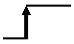
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions calculate the cosine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BCOS(EN,s,d); ENO:=BCOSP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BCOS	
BCOSP	

Setting data

■Descriptions, ranges, and data types

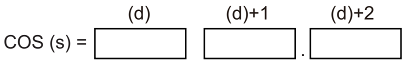
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for cosine calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the sign of the operation result in the word device specified by (d) and the operation result in the word devices specified by (d)+1 and (d)+2.



(d): Sign

(d)+1: Integral part

(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360°(in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -1.000 to 1.000.
- The operation result is a value whose 5th decimal place is rounded off.

Operation error

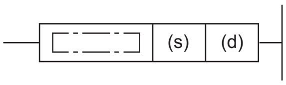
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

Calculating the tangent of BCD data

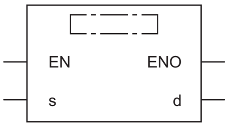
BTAN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

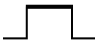
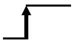
These instructions calculate the tangent of the angle specified by a BCD value.

Ladder	ST
	ENO:=BTAN(EN,s,d); ENO:=BTANP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BTAN	
BTANP	

Setting data

■Descriptions, ranges, and data types

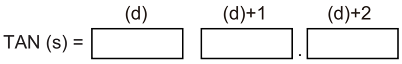
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for tangent calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the sign of the operation result in the device specified by (d) and the operation result in the devices specified by (d)+1 and (d)+2.



(d): Sign

(d)+1: Integral part

(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360°(in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -57.2901 to 57.2903.
- The operation result is a value whose 5th decimal place is rounded off.

Operation error

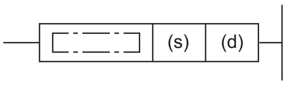
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360. • The specified data is 90° or 270°.

Calculating the arc sine of BCD data

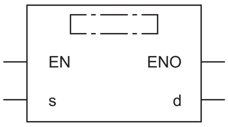
BASIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

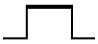
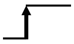
These instructions calculate the arc sine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BASIN(EN,s,d); ENO:=BASINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BASIN	
BASINP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc sine calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc sine of the value specified by (s), and store the operation result (angle) in the device specified by (d).

$$\mathrm{SIN}^{-1}\left(\overset{(s)}{\boxed{\qquad}}\ \overset{(s)+1}{\boxed{\qquad}}\,.\,\overset{(s)+2}{\boxed{\qquad}}\right) = (d)$$

(s): Sign

(s)+1: Integral part

(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 1.0000 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 90° or 270 to 360° (in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

Operation error

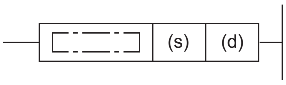
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

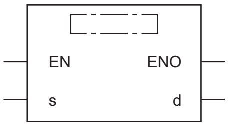
Calculating the arc cosine of BCD data

BACOS(P)

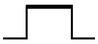
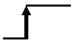
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions calculate the arc cosine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BACOS(EN,s,d); ENO:=BACOSP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BACOS	
BACOSP	

Setting data

■Descriptions, ranges, and data types

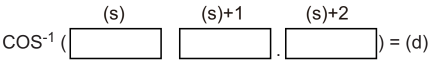
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc cosine calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc cosine of the value specified by (s), and store the operation result (angle) in the device specified by (d).



(s): Sign

(s)+1: Integral part

(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 1.0000 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 180°(in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

Operation error

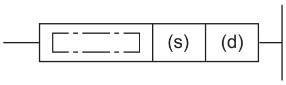
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

Calculating the arc tangent of BCD data

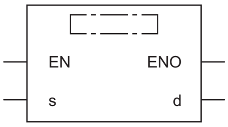
BATAN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

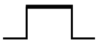
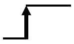
These instructions calculate the arc tangent of the angle specified by a BCD value.

Ladder	ST
	ENO:=BATAN(EN,s,d); ENO:=BATANP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BATAN	
BATANP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc tangent calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc tangent of the value specified by (s), and store the operation result (angle) in the device specified by (d).

$$\mathrm{TAN}^{-1}\ (\overset{(s)}{\boxed{\text{Sign}}}\ \overset{(s)+1}{\boxed{\text{Integral part}}}\,.\,\overset{(s)+2}{\boxed{\text{Decimal part}}}) = (d)$$

(s): Sign

(s)+1: Integral part

(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 9999.9999 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 90° or 270 to 360° (in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

Operation error

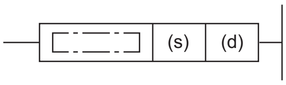
Error code (SD0)	**Description**
3405H	The data in the device specified by (s) is not a BCD value.

Converting single-precision real number angle to radian

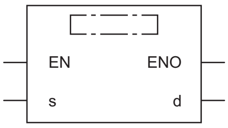
RAD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

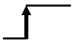
These instructions convert the unit of the measure of angle from the degree specified by a single-precision real number to radian.

Ladder	ST
	ENO:=RAD(EN,s,d); ENO:=RADP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RAD	
RADP	

Setting data

■Description, range, data type

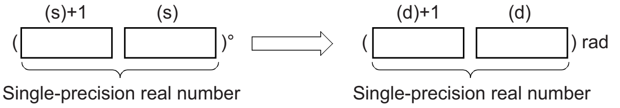
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the angle in radians	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the degree specified by (s) to the radian, and store the angle in radians in the device number specified by (d).



- Unit conversion from the degree to the radian is performed as follows.

$$\text{Radian} = \text{Degree} \times \frac{\pi}{180}$$

- 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 49 Precautions

Operation error

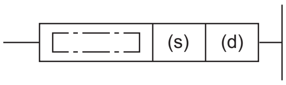
Error code (SD0)	**Description**
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Converting single-precision real number radian to angle

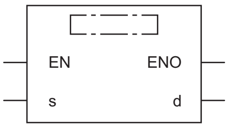
DEG(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

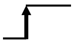
These instructions convert the unit of the measure of angle from the radian specified by a single-precision real number to the degree.

Ladder	ST
	ENO:=DEG(EN,s,d); ENO:=DEGP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DEG	
DEGP	

Setting data

■Description, range, data type

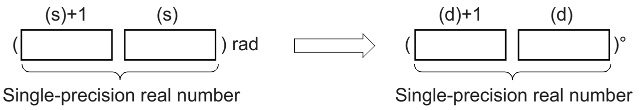
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	$0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the value converted in degrees	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the radian specified by (s) to the degree, and store the angle in degrees in the device number specified by (d).



- Unit conversion from the radian to the degree is performed as follows.

$$\text{Degree} = \text{Radian} \times \frac{\pi}{180}$$

- 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 49 Precautions

Operation error

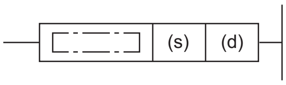
Error code (SD0)	**Description**
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

Converting double-precision real number angle to radian

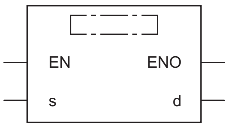
RADD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **✕ RnSFCPU (Safety)**

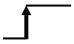
These instructions convert the unit of the measure of angle from the degree specified by a single-precision real number to radian.

Ladder	ST
	ENO:=RADD(EN,s,d); ENO:=RADDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RADD	
RADDP	

Setting data

■Description, range, data type

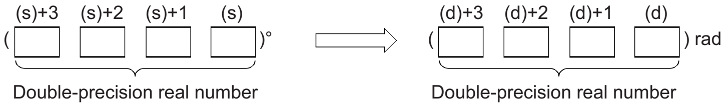
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-1022}\leq\|(s)\|<2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the angle in radians	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the degree specified by (s) to the radian, and store the angle in radians in the device number specified by (d).



- Unit conversion from the degree to the radian is performed as follows.

$$\text{Radian} = \text{Degree} \times \frac{\pi}{180}$$

- 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 49 Precautions

Operation error

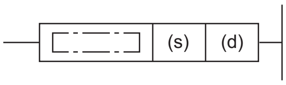
Error code (SD0)	**Description**
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Converting double-precision real number radian to angle

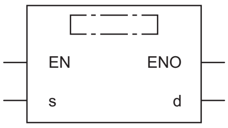
DEGD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

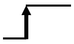
These instructions convert the unit of the measure of angle from the radian specified by a double-precision real number to the degree.

Ladder	ST
	ENO:=DEGD(EN,s,d); ENO:=DEGDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DEGD	
DEGDP	

Setting data

■Description, range, data type

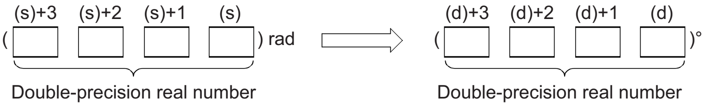
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	$0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the value converted in degrees	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the radian specified by (s) to the degree, and store the angle in degrees in the device number specified by (d).



- Unit conversion from the radian to the degree is performed as follows.

$$\text{Degree} = \text{Radian} \times \frac{180}{\pi}$$

- 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 49 Precautions

Operation error

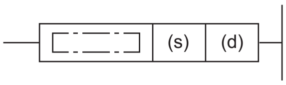
Error code (SD0)	**Description**
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Calculating the square root of single-precision real number

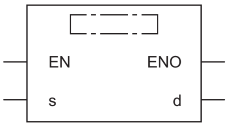
ESQRT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

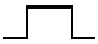
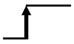
These instructions calculate the square root of the value specified by a single-precision real number.

Ladder	ST
	ENO:=ESQRT(EN,s,d); ENO:=ESQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESQRT	
ESQRTP	

Setting data

■Description, range, data type

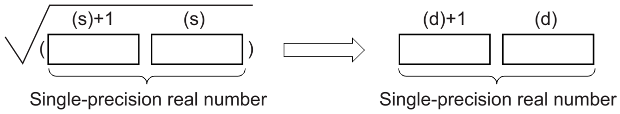
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0, $2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the square root of the value specified by (s), and store the operation result in the device specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

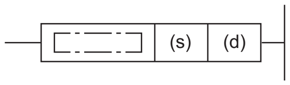
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is a negative number.

Calculating the square root of double-precision real number

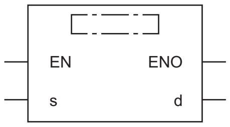
EDSQRT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

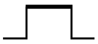
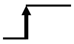
These instructions calculate the square root of the value specified by a double-precision real number.

Ladder	ST
	ENO:=EDSQRT(EN,s,d); ENO:=EDSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSQRT	
EDSQRTP	

Setting data

■Description, range, data type

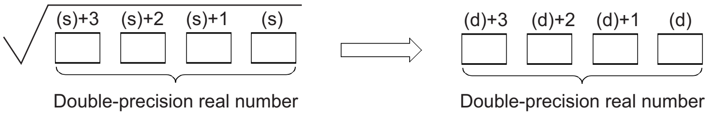
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	$0, 2^{-1022} \leq \lvert (s) \rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the value specified by (s), and store the operation result in the device specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

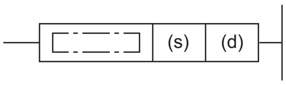
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is a negative number.

Calculating the exponent of single-precision real number

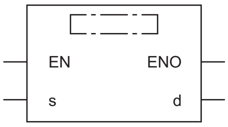
EXP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions calculate the exponent of the value specified by a single-precision real number.

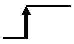
Ladder	ST[*1]
	ENO:=EXPP(EN,s,d);

FBD/LD[*1]



*1 The EXP instruction does not support the ST and FBD/LD. Use the standard function, EXP.
☞ Page 1899 EXP(_E)

■Execution condition

Instruction	Execution condition
EXP	
EXPP	

Setting data

■Description, range, data type

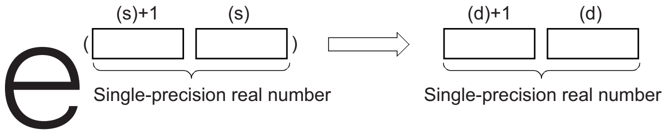
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	0, $2^{-126} \leq |(s)| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the exponent of the value specified by (s), and store the operation result in the device specified by (d).



- Exponent operation is performed with the base (e) set to "2.71828".
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Point

- The EXP(P) instruction performs operation using the natural logarithm. To determine a value using the common logarithm, determine a value by dividing the common logarithm value by 0.43429 and specify it in the device specified by (s).

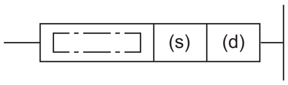
$$10^X = e^{\frac{X}{0.43429}}$$

Calculating the exponent of double-precision real number

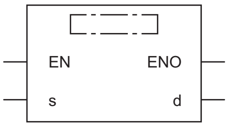
EXPD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions calculate the exponent of the value specified by a double-precision real number.

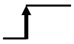
Ladder	ST[*1]
	ENO:=EXPDP(EN,s,d);

FBD/LD[*1]



[*1] The EXPD instruction does not support the ST and FBD/LD. Use the standard function, EXP.
☞ Page 1899 EXP(_E)

■Execution condition

Instruction	Execution condition
EXPD	
EXPDP	

Setting data

■Description, range, data type

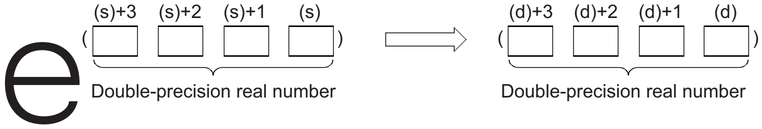
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	$0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the exponent of the value specified by (s), and store the operation result in the device specified by (d).



- Exponent operation is performed with the base (e) set to "2.71828".
- 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 49 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert (d) \rvert < 2^{1024}$

Point

- The EXPD(P) instruction performs operation using the natural logarithm. To determine a value using the common logarithm, determine a value by dividing the common logarithm value by 0.43429 and specify it in the device specified by (s).

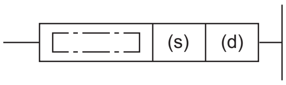
$$10^{X} = e^{\frac{X}{0.43429}}$$

Calculating the natural logarithm of single-precision real number

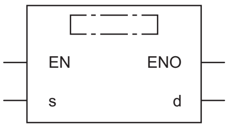
LOG(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions calculate the logarithm using the natural logarithm (e) of the value specified by a single-precision real number as the base.

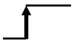
Ladder	ST[*1]
	ENO:=LOGP(EN,s,d);

FBD/LD[*1]



[*1] The LOG instruction does not support the ST and FBD/LD. Use the standard function, LOG.
☞ Page 1897 LOG(_E)

■Execution condition

Instruction	Execution condition
LOG	
LOGP	

Setting data

■Description, range, data type

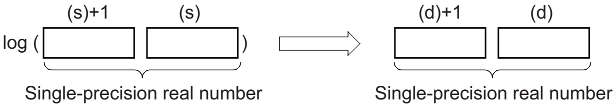
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-126} \le \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the logarithm using natural logarithm (e) of the value specified by (s), and store the operation result in the device specified by (d).



- Input a positive value only. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

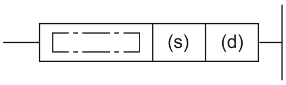
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Calculating the natural logarithm of double-precision real number

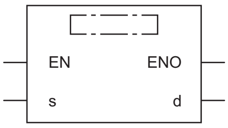
LOGD(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions calculate the logarithm using the natural logarithm (e) of the value specified by a double-precision real number as the base.

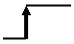
Ladder	ST[*1]
	ENO:=LOGDP(EN,s,d);

FBD/LD[*1]



*1 The LOGD instruction does not support the ST and FBD/LD. Use the standard function, LOG.
☞ Page 1897 LOG(_E)

■Execution condition

Instruction	Execution condition
LOGD	
LOGDP	

Setting data

■Description, range, data type

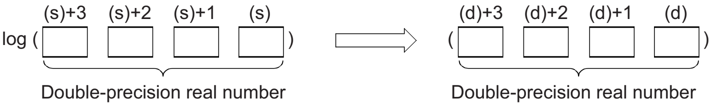
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-1022} \leq \vert(s)\vert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the logarithm using natural logarithm (e) of the value specified by (s), and store the operation result in the device specified by (d).



- Input a positive value only. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

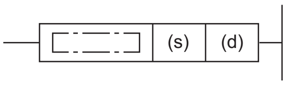
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

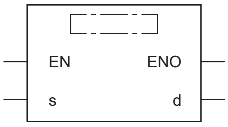
Calculating the square root of BCD 4-digit data

BSQRT(P)

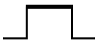
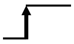
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions calculate the square root of the value specified by a BCD 4-digit data.

Ladder	ST
	ENO:=BSQRT(EN,s,d); ENO:=BSQRTP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BSQRT	
BSQRTP	

Setting data

■Descriptions, ranges, and data types

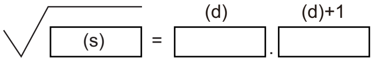
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the BCD 4-digit data specified by (s), and store the operation result in the device specified by (d).



(d): Integral part

(d)+1: Decimal part

- The value to be specified in (s) is a BCD value with a maximum of 4 digits (0 to 9999).
- A BCD value from 0 to 9999.9999 is stored as the operation result in the device specified by (d).
- The operation result is a value whose 5th decimal place is rounded down.

Operation error

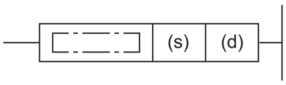
Error code (SD0)	Description
3405H	The data in the device specified by (s) is not a BCD value.

Calculating the square root of BCD 8-digit data

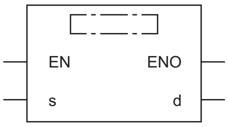
BDSQRT(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

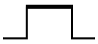
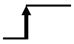
These instructions calculate the square root of the value specified by a BCD 8-digit data.

Ladder	ST
	ENO:=BDSQRT(EN,s,d); ENO:=BDSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BDSQRT	
BDSQRTP	

Setting data

■Description, range, data type

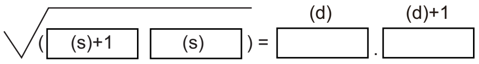
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the BCD 8-digit data specified by (s), and store the operation result in the device specified by (d).



(s)+1, (s): 2-word data

(d): Integral part

(d)+1: Decimal part

- The value to be specified in (s) is a BCD value with a maximum of 8 digits (0 to 99999999).
- A BCD value from 0 to 9999.9999 is stored as the operation result in the device specified by (d).
- The operation result is a value whose 5th decimal place is rounded down.

Operation error

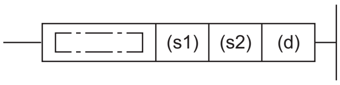
Error code (SD0)	Description
3405H	The data in the device specified by (s) is not a BCD value.

Calculating the exponentiation of single-precision real number

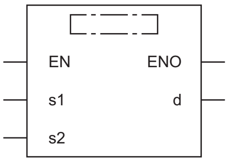
POW(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

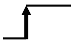
These instructions calculate the exponentiation of a single-precision real number.

Ladder	ST
	ENO:=POW(EN,s1,s2,d); ENO:=POWP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
POW	
POWP	

Setting data

■Description, range, data type

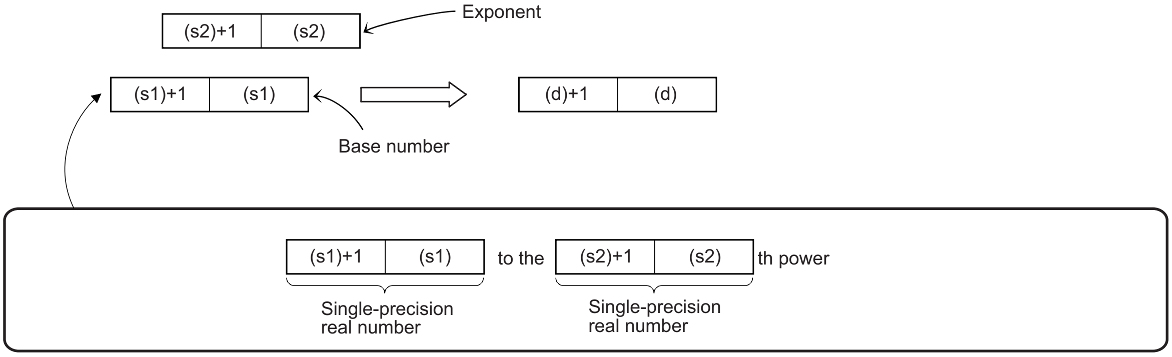
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	$0, 2^{-126}\leq|(s1)|<2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Exponentiation data or the start device containing the data	$0, 2^{-126}\leq|(s2)|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions raises the single-precision real number specified by (s1) to the power of the single-precision real number specified by (s2), and store the operation result in the device specified by (d).



- The values that can be specified by (s1) and (s2) and the value that can be stored are 0, 2^{-126}≤|setting value (stored value)|<2^{128}.
- 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 49 Precautions

Operation error

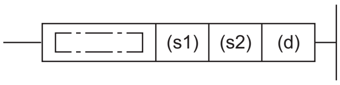
Error code (SD0)	Description
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

Calculating the exponentiation of double-precision real number

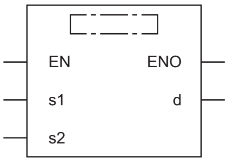
POWD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

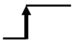
These instructions calculate the exponentiation of a double-precision real number.

Ladder	ST
	ENO:=POWD(EN,s1,s2,d); ENO:=POWDP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
POWD	
POWDP	

Setting data

■Description, range, data type

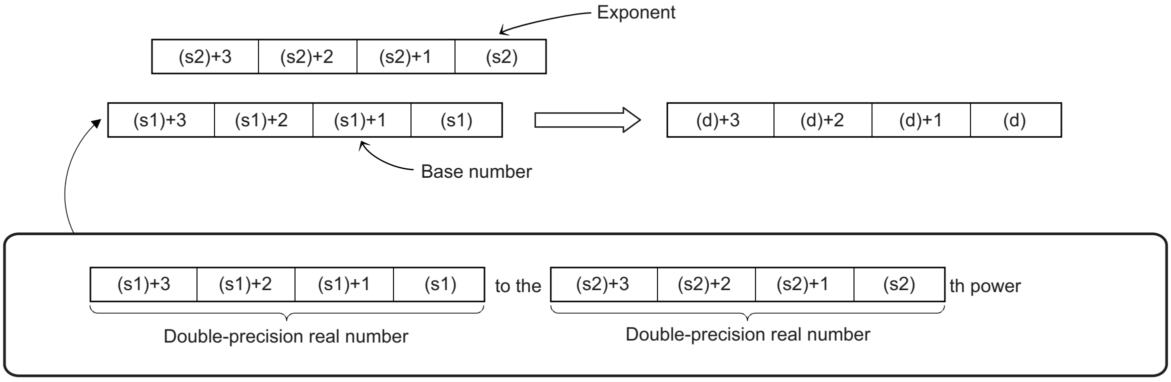
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	0, $2^{-1022} \leq \|(\text{s1})\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Exponentiation data or the start device containing the data	0, $2^{-1022} \leq \|(\text{s2})\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions raises the double-precision real number specified by (s1) to the power of the double-precision real number specified by (s2), and store the operation result in the device specified by (d).



- The values that can be specified by (s1) and (s2) and the value that can be stored are 0, $2^{-1022} \leq |\text{setting value (stored value)}| < 2^{1024}$.
- 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 49 Precautions

Operation error

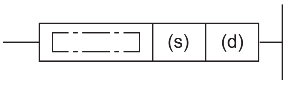
Error code (SD0)	Description
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

Calculating the common logarithm of single-precision real number

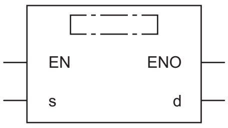
LOG10(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

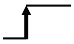
These instructions calculate the logarithm using the common logarithm (using 10 as the base) of the value specified by a single-precision real number.

Ladder	ST
	ENO:=LOG10(EN,s,d); ENO:=LOG10P(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
LOG10	
LOG10P	

Setting data

■Description, range, data type

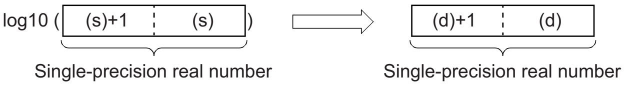
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for common logarithm operation, or the start device containing the data	0, $2^{-126}\leq\|(s)\|<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the common logarithm (using 10 as the base) of the value specified by (s), and store the operation result in the device number specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

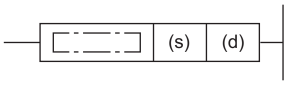
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Calculating the common logarithm of double-precision real number

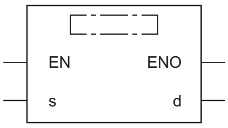
LOG10D(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

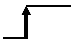
These instructions calculate the logarithm using the common logarithm (using 10 as the base) of the value specified by a double-precision real number.

Ladder	ST
	ENO:=LOG10D(EN,s,d); ENO:=LOG10DP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
LOG10D	
LOG10DP	

Setting data

■Description, range, data type

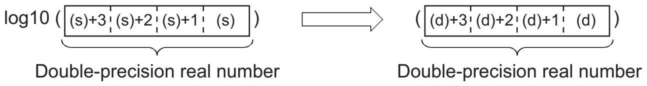
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for common logarithm operation, or the start device containing the data	0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the common logarithm (using 10 as the base) of the value specified by (s), and store the operation result in the device number specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- 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 49 Precautions

Operation error

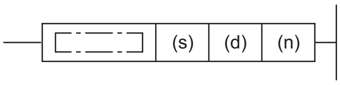
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Searching the maximum value of single-precision real number

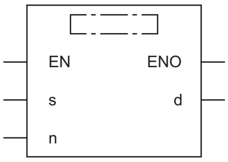
EMAX(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions search the block data of single-precision real numbers for the maximum value.

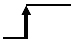
Ladder	ST[1]
	ENO:=EMAXP(EN,s,n,d);

FBD/LD[1]



*1 The EMAX instruction does not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EMAX	
EMAXP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number used for maximum value search, or the start device containing single-precision real numbers	$0, 2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32[1]
(d)	Start device for storing the search result (d) to (d)+1: Maximum value (d)+2: Position (d)+3: The number of search target data points	—	Single-precision real number	ANY_REAL_32_ARRAY (Number of elements: 4)
(n)	Number of single-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

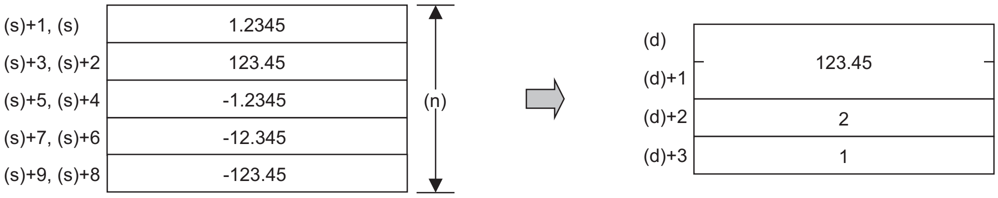
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the maximum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first maximum value by the number of points from (s) in the device specified by (d)+2 and the number of maximum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1: Maximum value

(d)+2: Location

(d)+3: Number of maximum values

- When (n) is 0, the processing is not performed.

Operation error

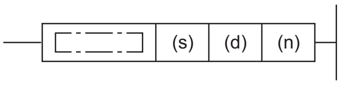
Error code (SD0)	**Description**
3402H	The block data in the device specified by (s) includes a value other than single-precision real number.

Searching the maximum value of double-precision real number

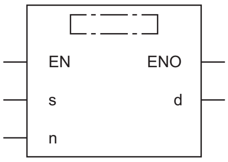
EDMAX(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions search the block data of double-precision real numbers for the maximum value.

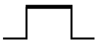
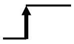
Ladder	ST[1]
	ENO:=EDMAXP(EN,s,n,d);

FBD/LD[1]



[1] The EDMAX instruction does not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EDMAX	
EDMAXP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number used for maximum value search, or the start device containing double-precision real numbers	0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64[1]
(d)	Start device for storing the search result (d) to (d)+3: Maximum value (d)+4: Position (d)+5: The number of search target data points	—	Double-precision real number	ANY_REAL_64_ARRAY (Number of elements: 6)
(n)	Number of double-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

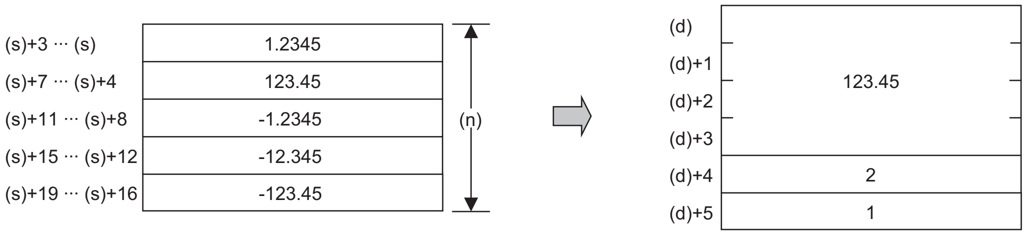
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the maximum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first maximum value by the number of points from (s) in the device specified by (d)+4 and the number of maximum values in the device specified by (d)+5.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1, (d)+2, (d)+3: Maximum value

(d)+4: Location

(d)+5: Number of maximum values

- When (n) is 0, the processing is not performed.

Operation error

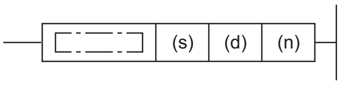
Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than double-precision real number.

Searching the minimum value of single-precision real number

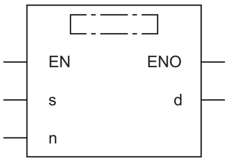
EMIN(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions search the block data of single-precision real numbers for the minimum value.

Ladder	ST[1]
	ENO:=EMINP(EN,s,n,d);

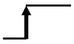
FBD/LD[1]



---

[1] The EMIN instruction does not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EMIN	
EMINP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number used for minimum value search, or the start device containing single-precision real numbers	0, $2^{-126} \leq \|(s)\| < 2^{128}$	Single-precision real number	ANYREAL_32[1]
(d)	Start device for storing the search result (d) to (d)+1: Minimum value (d)+2: Position (d)+3: The number of search target data points	—	Single-precision real number	ANY_REAL_32_ARRAY (Number of elements: 4)
(n)	Number of single-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

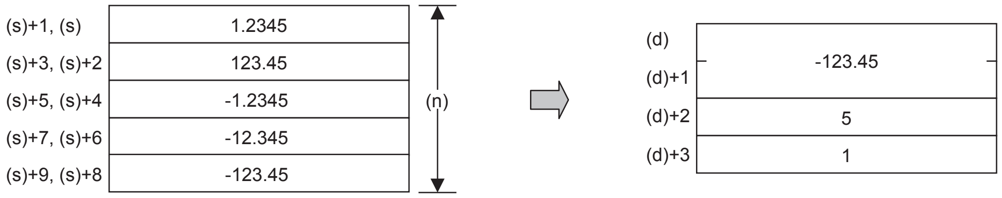
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the minimum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first minimum value by the number of points from (s) in the device specified by (d)+2 and the number of minimum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1: Minimum value

(d)+2: Location

(d)+3: Number of minimum values

- When (n) is 0, the processing is not performed.

Operation error

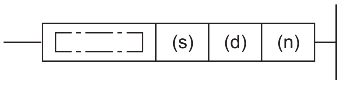
Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than single-precision real number.

Searching the minimum value of double-precision real number

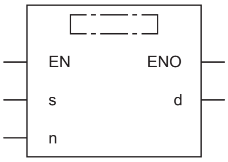
EDMIN(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions search the block data of double-precision real numbers for the minimum value.

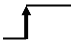
Ladder	ST[*1]
	ENO:=EDMINP(EN,s,n,d);

FBD/LD[*1]



*1 The EDMIN instruction does not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EDMIN	
EDMINP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number used for minimum value search, or the start device containing double-precision real numbers	$0, 2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64[*1]
(d)	Start device for storing the search result (d) to (d)+3: Minimum value (d)+4: Position (d)+5: The number of search target data points	—	Double-precision real number	ANY_REAL_64_ARRAY (Number of elements: 6)
(n)	Number of double-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

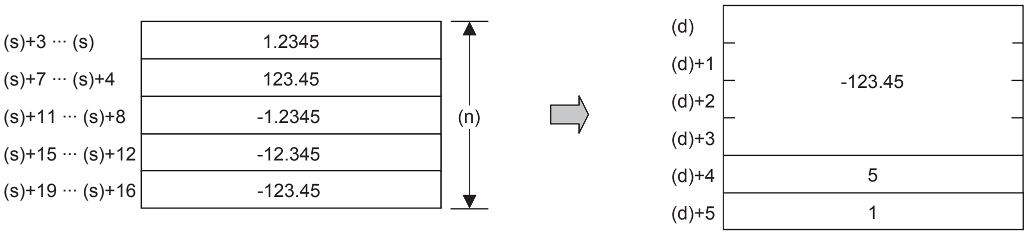
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the minimum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first minimum value by the number of points from (s) in the device specified by (d)+4 and the number of minimum values in the device specified by (d)+5.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1, (d)+2, (d)+3: Minimum value

(d)+4: Location

(d)+5: Number of minimum values

- When (n) is 0, the processing is not performed.

Operation error

Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than double-precision real number.

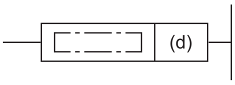
7.10 Random Number Instructions

Generating random number

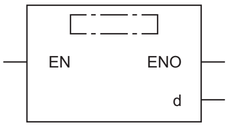
RND(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

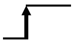
These instructions generate a random number between 0 and less than 32767, and store the random number in the specified device.

Ladder	ST
	ENO:=RND(EN,d); ENO:=RNDP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RND	
RNDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Device for storing the random number	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

These instructions generate a random number between 0 and 32767, and store the random number in the device specified by (d). The RND(P) instruction generates random numbers according to a certain calculation expression. The calculation expression uses the previous calculation result as a coefficient.

Operation error

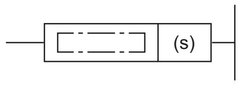
There is no operation error.

Changing random sequence

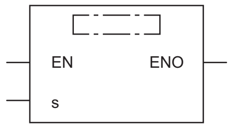
SRND(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

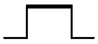
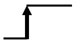
These instructions change the random number sequence according to the content of the 16-bit binary data stored in the specified device.

Ladder	ST
	ENO:=SRND(EN,s); ENO:=SRNDP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
SRND	
SRNDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Random number sequence data	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

These instructions change the random number sequence according to the content of the 16-bit binary data stored in the device specified by (s). The SRND(P) instruction can change the random number calculation pattern.

Operation error

There is no operation error.

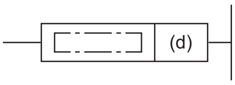
7.11 Index Register Instructions

Saving all data of the index register

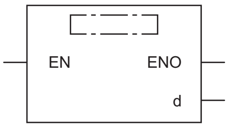
ZPUSH(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

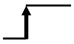
These instructions save the content of the index register to the specified area.

Ladder	ST
	ENO:=ZPUSH(EN,d); ENO:=ZPUSHP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZPUSH	
ZPUSHP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to which the index register will be saved	—	16-bit signed binary	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

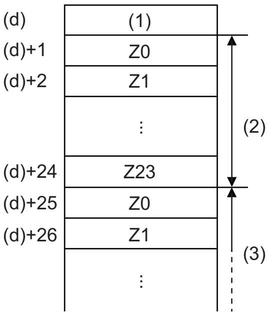
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions save the content of the index register to the device specified by (d) or the array label element and later.
- When the content of the index register is saved, the number of saves (d) is incremented by 1.
- Regardless of the number of points assigned to the index register and long index register, 24 words of data are saved. Accordingly, when 0 point is assigned to the index register, the long index register is saved by 12 points.
- The ZPOP(P) instructions can be used to restore data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs and can be nested to be used as a stack.

☞ Page 851 ZPOP(P)

- A nesting structure can be created by specifying the area specified by (d) of the ZPUSH(P) instruction in (d) of the ZPUSH(P) instruction again. The number of saves (d) is incremented by 1 every time the ZPUSH(P) instruction is executed.
- When another area is specified in (d) of the ZPUSH(P) instruction in the nesting structure, the content of the index register is saved to the specified another area.
- In the nesting structure, every time the ZPUSH(P) instruction is executed, saved data is added. Therefore, secure in advance the areas necessary for the number of times the instruction is executed.
- The following figure shows the configuration of the areas used after (d).



(1) Number of saves
(2) 1st nesting (24 words)
(3) 2nd nesting

Operation error

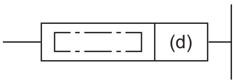
Error code (SD0)	Description
3405H	The number of saves in (d)+0 is FFFF.

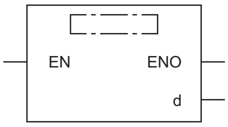
Returning all data of the index register

ZPOP(P)

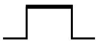
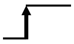
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions read the data, which has been saved to the specified area, into the index register.

Ladder	ST
	ENO:=ZPOP(EN,d); ENO:=ZPOPP(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
ZPOP	
ZPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to which the index register will be restored	—	16-bit signed binary	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the data, which has been saved to the device specified by (d) or the array label element and later, into the index register.
- When the content of the index register is read, the number of saves (d) is decremented by 1.
- Refer to the following for the configuration of the areas used after (d).

☞ Page 849 ZPUSH(P)

Operation error

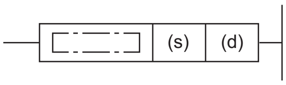
Error code (SD0)	Description
3405H	The number of saves in (d)+0 is 0.

Saving the selected data of the index register and long index register

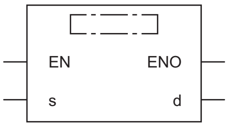
ZPUSH(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instruction save the contents of the index register and long index register to the specified area.

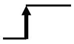
Ladder	ST
	ENO:=ZPUSH_2(EN,s,d); ENO:=ZPUSHP_2(EN,s,d);

FBD/LD



(□ is replaced by ZPUSH_2 or ZPUSHP_2.)

■Execution condition

Instruction	Execution condition
ZPUSH	
ZPUSHP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Type of the index register and long index register to be saved	1 to 3	16-bit unsigned binary	ANY16
(d)	Start device to which the index register and long index register will be saved	—	16-bit signed binary	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

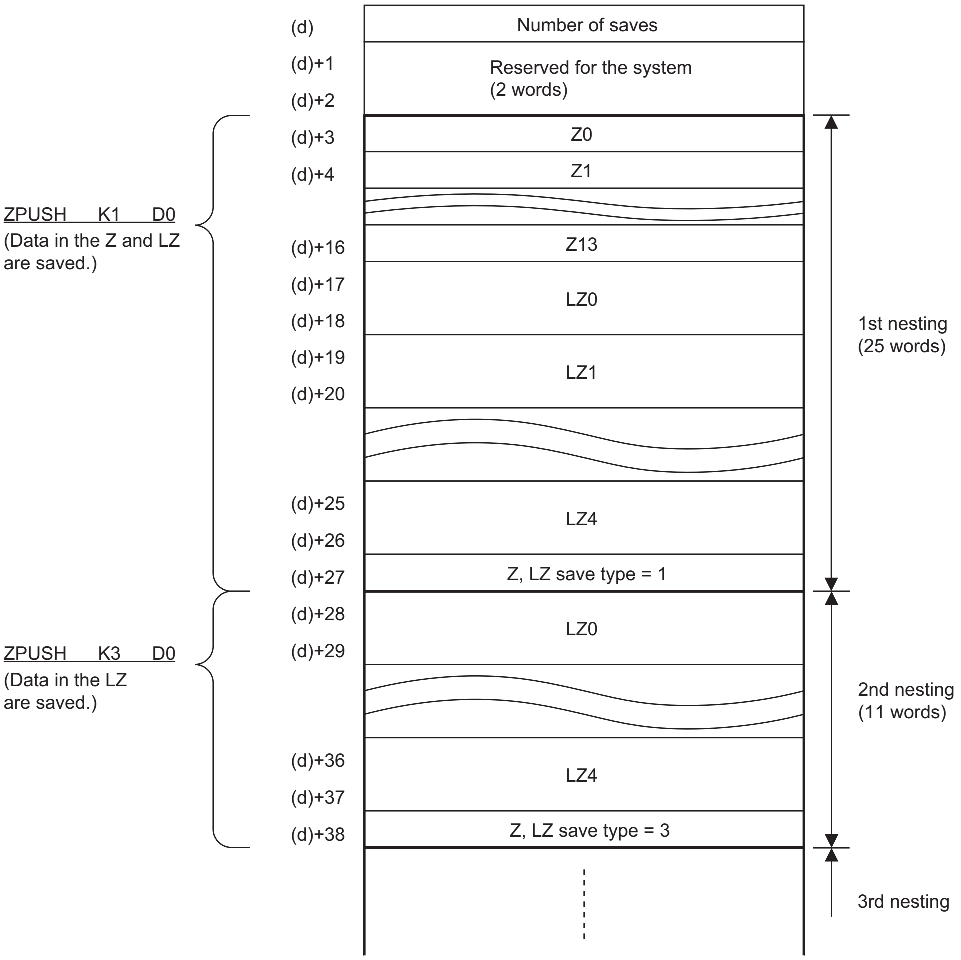
- These instructions save the contents of the index register and long index register in the device specified by (s) to the device specified by (d) or array label element and later. The instructions also save the type of the saved index register and long index register to the end of the saved data.
- When the contents of the index register and long index register are saved, the number of saves (d) is incremented by 1.
- The following table lists the values specified by (s) and the index registers and long index registers to be saved.

Value of (s)	Z and LZ saved
1	Total range of Z and LZ
2	Total range of Z
3	Total range of LZ

- The ZPOP(P) instructions (restoring the selected data of the index register and long index register) are used to restore data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs and can be nested to be used as a stack.

☞ Page 855 ZPOP(P)

- A nesting structure can be created by specifying the area specified by (d) of the ZPUSH(P) instruction in (d) of the ZPUSH(P) instruction again. The number of saves (d) is incremented by 1 every time the ZPUSH(P) instruction is executed.
- When another area is specified in (d) of the ZPUSH(P) instruction in the nesting structure, the content of the index register or long index register is saved to the specified another area.
- In the nesting structure, every time the ZPUSH(P) instruction is executed, saved data is added. Therefore, check the numbers of points assigned to the index register and long index register according to SD300 and SD302, and secure in advance the areas necessary for the number of times the instruction is executed.
- The following figure shows the configuration of the areas used after (d). (Z0 to Z23 and LZ0 to LZ4)



Precautions

(d)+1 and (d)+2 for the ZPUSH(P) instructions are used for the system. Do not change the values.
The Z and LZ save types stored in the area specified by (d) and later are also used for the system. Do not change the values. Changing the values may cause malfunction of the module.

Operation error

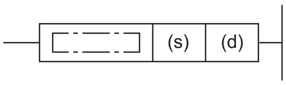
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified value is other than 1 to 3. • When the number of index register points is 0, 2 is specified. • When the number of long index register points is 0, 3 is specified.
	The value stored in the system-reserved area in the area specified by (d) has been changed.

Returning the selected data of the index register and long index register

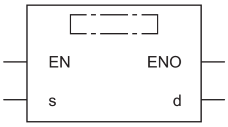
ZPOP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions read the data, which has been saved to the specified area, into the index register and long index register.

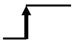
Ladder	ST
	ENO:=ZPOP_2(EN,s,d); ENO:=ZPOPP_2(EN,s,d);

FBD/LD



(□ is replaced by ZPOP_2 or ZPOPP_2.)

■Execution condition

Instruction	Execution condition
ZPOP	
ZPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Dummy	—	16-bit unsigned binary	ANY16
(d)	Start device to which the index register will be restored	—	16-bit signed binary	ANY16[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the data, which has been saved to the device specified by (d) or the array label element and later, into the index register or long index register.
- When the data saved to the index register and long index register is read, the number of saves (d) is decremented by 1.
- The date data in the device specified by (s) is regarded as dummy data and ignored.
- Refer to the following for the configuration of the areas used after (d).

☞ Page 852 ZPUSH(P)

Precautions

(d)+1 and (d)+2 are used for the system. Do not change the values.
The Z and LZ save types stored in the area specified by (d) and later are also used for the system. Do not change the values. Changing the values may cause malfunction of the module.

Operation error

Error code (SD0)	Description
3405H	The number of saves in (d)+0 is 0.
	The value stored in the system-reserved area in the area specified by (d) has been changed.
	A value other than 1 to 3 is set to the Z, LZ save type.

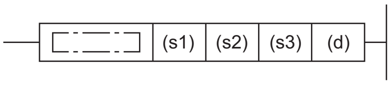
7.12 Data Control Instructions

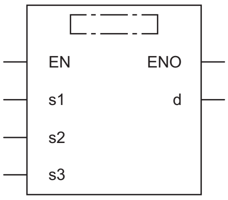
Upper and lower limit control of 16-bit binary data

LIMIT(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

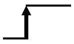
These instructions control the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits.

Ladder	ST[*1]	
	ENO:=LIMITP(EN,s1,s2,s3,d);	ENO:=LIMITP_U(EN,s1,s2,s3,d);

FBD/LD[*1]


[*1] The LIMIT and LIMIT_U instructions do not support the ST and FBD/LD. Use the standard function, LIMIT.
☞ Page 1935 LIMIT(_E)

■Execution condition

Instruction	Execution condition
LIMIT LIMIT_U	
LIMITP LIMITP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	LIMIT(P)	Lower limit value (minimum output threshold value)	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	LIMIT(P)	Upper limit value (maximum output threshold value)	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	LIMIT(P)	Input value to be controlled with the upper and lower limits	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	LIMIT(P)	Device for storing the output value controlled by upper/lower limit control	—	16-bit signed binary	ANY16_S
	LIMIT(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

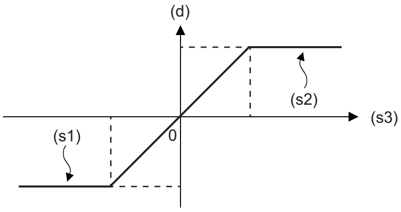
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Lower limit value (s1) > Input value (s3)	Lower limit value (s1)
Upper limit value (s2) < Input value (s3)	Upper limit value (s2)
Lower limit value (s1) ≤ Input value (s3) ≤ Upper limit value (s2)	Input value (s3)



- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

Operation error

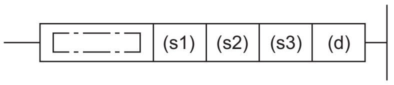
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

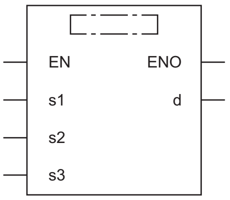
Upper and lower limit control of 32-bit binary data

DLIMIT(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

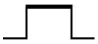
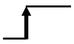
These instructions control the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits.

Ladder	ST*1	
	ENO:=DLIMITP(EN,s1,s2,s3,d);	ENO:=DLIMITP_U(EN,s1,s2,s3,d);

FBD/LD*1


*1 The DLIMIT and DLIMIT_U instructions do not support the ST and FBD/LD. Use the standard function, LIMIT.
☞ Page 1935 LIMIT(_E)

■Execution condition

Instruction	Execution condition
DLIMIT DLIMIT_U	
DLIMITP DLIMITP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	DLIMIT(P)	Device for storing the lower limit value (minimum output threshold value)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DLIMIT(P)	Device for storing the upper limit value (maximum output threshold value)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DLIMIT(P)	Device for storing the input value controlled by upper/lower limit control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DLIMIT(P)	Start device for storing the controlled output value	—	32-bit signed binary	ANY32_S
	DLIMIT(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

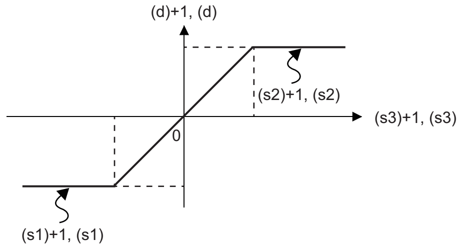
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Lower limit value ((s1), (s1)+1) > Input value ((s3), (s3)+1)	Lower limit value ((s1), (s1)+1)
Lower limit value ((s2), (s2)+1) < Input value ((s3), (s3)+1)	Upper limit value ((s2), (s2)+1)
Lower limit value ((s1), (s1)+1) ≤Input value ((s3), (s3)+1) ≤ Upper limit value ((s2), (s2)+1)	Input value ((s3), (s3)+1)



- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

Operation error

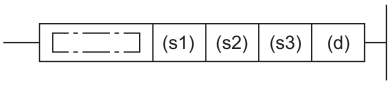
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Dead band control of 16-bit binary data

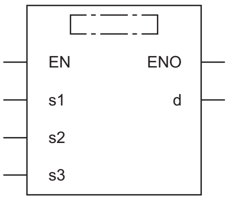
BAND(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

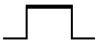
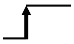
These instructions control the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits of the dead band.

Ladder	ST	
	ENO:=BAND(EN,s1,s2,s3,d); ENO:=BANDP(EN,s1,s2,s3,d);	ENO:=BAND_U(EN,s1,s2,s3,d); ENO:=BANDP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BAND BAND_U	
BANDP BANDP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	BAND(P)	Lower limit of dead band (no-output band)	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	BAND(P)	Upper limit of dead band (no-output band)	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	BAND(P)	Input value to be controlled by dead band control	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	BAND(P)	Device for storing the output value controlled by dead band control	—	16-bit signed binary	ANY16_S
	BAND(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

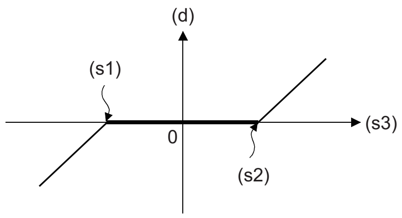
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking whether the input value (16-bit binary data) in the device specified by (s3) is within range of the upper and lower limits of the dead band in the devices specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Dead band lower limit value (s1) > input value (s3)	Input value (s3) - dead band lower limit value (s1)
Dead band upper limit value (s2) < input value (s3)	Input value (s3) - dead band upper limit value (s2)
Dead band lower limit value (s1) ≤ input value (s3) ≤ dead band upper limit (s2)	0



- The following example shows the case where the operation result of the BAND(P) instruction is out of the range from -32768 to 32767.

Ex.

When (s1) is 10 and (s3) is -32768, output value is -32768-10 = 8000H-000AH = 7FF6H = 32758.

- The following example shows the case when the operation result of the BAND(P)_U instruction is out of the range from 0 to 65535.

Ex.

When (s1) is 100 and (s3) is 50, output value is 50-100 = 0032H-0064H = FFCEH = 65486.

Operation error

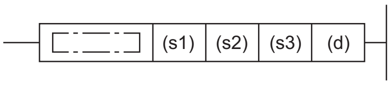
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Dead band control of 32-bit binary data

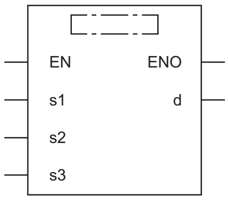
DBAND(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

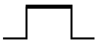
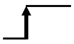
These instructions control the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits of the dead band.

Ladder	ST	
	ENO:=DBAND(EN,s1,s2,s3,d); ENO:=DBANDP(EN,s1,s2,s3,d);	ENO:=DBAND_U(EN,s1,s2,s3,d); ENO:=DBANDP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBAND DBAND_U	
DBANDP DBANDP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	DBAND(P)	Device for storing the lower limit value of dead band (no-output band)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DBAND(P)	Device for storing the upper limit value of dead band (no-output band)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DBAND(P)	Device for storing the input value controlled by dead band control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DBAND(P)	Start device for storing the output value controlled by dead band control	—	32-bit signed binary	ANY32_S
	DBAND(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

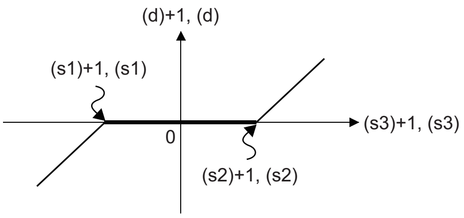
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking whether the input value (32-bit binary data) in the device specified by (s3) is within range of the upper and lower limits of the dead band in the devices specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Dead band lower limit value ((s1), (s1)+1) > Input value ((s3), (s3)+1)	Input value ((s3), (s3)+1) - Dead band lower limit value ((s1), (s1)+1)
Dead band upper limit value ((s2), (s2)+1) < Input value ((s3), (s3)+1)	Input value ((s3), (s3)+1) - Dead band upper limit value ((s2), (s2)+1)
Dead band lower limit value ((s1), (s1)+1) ≤ Input value ((s3), (s3)+1) ≤ Dead band upper limit ((s2), (s2)+1)	0



- The following example shows the case when the operation result of the DBAND(P) instruction is out of the range from -2147483648 to 2147483647.

Ex.

When ((s1), (s1)+1) is 1000 and ((s3), (s3)+1) is -2147483648, output value is -2147483648-1000 = 80000000H-000003E8H = 7FFFFC18H = 2147482648.

- The following example shows the case when the operation result of the DBAND(P)_U instruction is out of the range from 0 to 4294967295.

Ex.

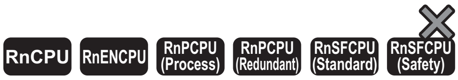
When ((s1), (s1)+1) is 100 and ((s3), (s3)+1) is 50, output value is 50-100 = 00000032H-00000064H = FFFFFFCEH = 4294967246.

Operation error

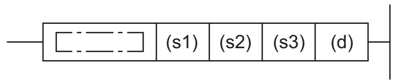
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Zone control of 16-bit binary data

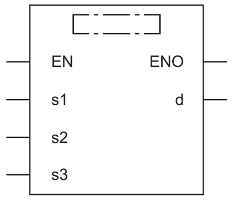
ZONE(P)(_U)



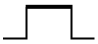
These instructions add a bias value to the specified input value (16-bit binary).

Ladder	ST	
	ENO:=ZONE(EN,s1,s2,s3,d); ENO:=ZONEP(EN,s1,s2,s3,d);	ENO:=ZONE_U(EN,s1,s2,s3,d); ENO:=ZONEP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZONE ZONE_U	
ZONEP ZONEP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	ZONE(P)	Negative bias value to be added to the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	ZONE(P)	Positive bias value to be added to the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	ZONE(P)	Input value used for zone control	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	ZONE(P)	Device for storing the output value controlled by zone control	—	16-bit signed binary	ANY16_S
	ZONE(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

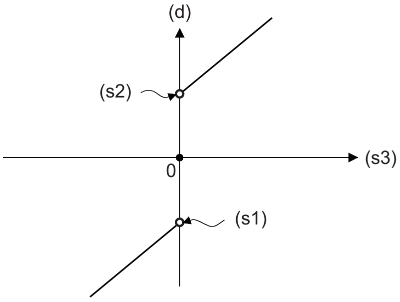
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (16-bit binary) specified by (s3), and store the result in the device number specified by (d). The bias value is controlled as follows.

Condition	Output value
Input value (s3) < 0	Input value (s3) + negative bias value (s1)
Input value (s3) < 0	0
Input value (s3) < 0	Input value (s3) + positive bias value (s2)



- The following example shows the case where the operation result of the ZONE(P) instruction is out of the range from -32768 to 32767.

Ex.

When (s1) is -100 and (s3) is -32768, output value is -32768+(-100) = 8000H-FF9CH = 7F9CH = 32668.

- The following example shows the case where the operation result of the ZONE(P)_U instruction is out of the range from 0 to 65535.

Ex.

When (s2) is 100 and (s3) is 65535, output value is 65535+100 = FFFFH-0064H = 0063H = 99.

- The ZONE(P)_U instruction treats the data in the device specified by (s1) as dummy and does not use it.

Operation error

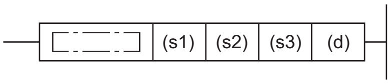
There is no operation error.

Zone control of 32-bit binary data

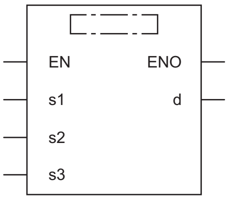
DZONE(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

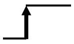
These instructions add a bias value to the specified input value (32-bit binary).

Ladder	ST	
	ENO:=DZONE(EN,s1,s2,s3,d); ENO:=DZONEP(EN,s1,s2,s3,d);	ENO:=DZONE_U(EN,s1,s2,s3,d); ENO:=DZONEP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DZONE DZONE_U	
DZONEP DZONEP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s1)	DZONE(P)	Device for storing the negative bias value to be added to the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DZONE(P)	Device for storing the positive bias value to be added to the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DZONE(P)	Device for storing the Input value used for zone control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DZONE(P)	Start device for storing the output value controlled by zone control	—	32-bit signed binary	ANY32_S
	DZONE(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

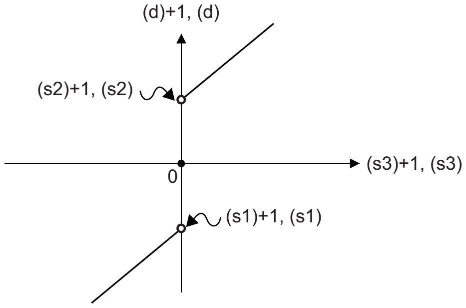
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (32-bit binary) specified by (s3), and store the result in the device number specified by (d). The bias value is controlled as follows.

Condition	Output value
Input value ((s3), (s3)+1) < 0	Input value ((s3), (s3)+1) + negative bias value (s1), (s1)+1
Input value ((s3), (s3)+1) = 0	0
Input value ((s3), (s3)+1) > 0	Input value ((s3), (s3)+1) + positive bias value (s2), (s2)+1



- The following example shows the case where the operation result of the DZONE(P) instruction is out of the range from -2147483648 to 2147483647.

Ex.

When ((s1), (s1)+1) is -1000 and ((s3), (s3)+1) is -2147483648, output value is -2147483648+(-1000) = 80000000H-FFFFFC18H = 7FFFFC18H = 2147482648.

- The following example shows the case where the operation result of the DZONE(P)_U instruction is out of the range from 0 to 4294967295.

Ex.

When ((s2), (s2)+1) is 1000 and ((s3), (s3)+1) is 4294967295, output value is 4294967295+1000 = FFFFFFFFH-00003E8H = 000003E7H = 999.

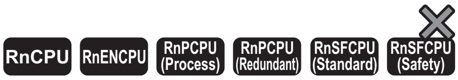
- The DZONE(P)_U instruction treats the data in the device specified by (s1) and (s1)+1 as dummy and does not use them.

Operation error

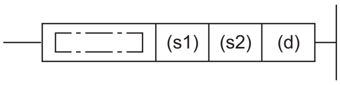
There is no operation error.

Scaling 16-bit binary data (point coordinates)

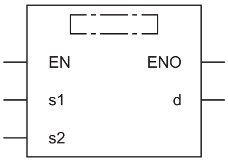
SCL(P)(_U)



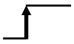
These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (point coordinates).

Ladder	ST	
	ENO:=SCL(EN,s1,s2,d); ENO:=SCLP(EN,s1,s2,d);	ENO:=SCL_U(EN,s1,s2,d); ENO:=SCLP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCL SCL_U	
SCLP SCLP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCL(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCL(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	SCL(P)	Start device containing scaling conversion data	—	16-bit signed binary[*1]	ANY16_S[*2]
	SCL(P)_U			16-bit unsigned binary[*1]	ANY16_U[*2]
(d)	SCL(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCL(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[*1] The number of coordinate points in (s2) is represented in 16-bit unsigned binary.

[*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.

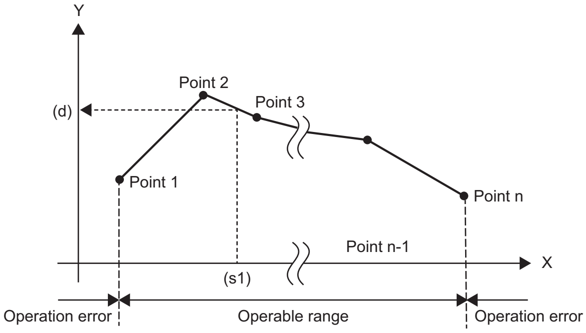
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)
Point 1	X coordinate	(s2)+1
	Y coordinate	(s2)+2
Point 2	X coordinate	(s2)+3
	Y coordinate	(s2)+4
⋮		
Point n	X coordinate	(s2)+2n-1
	Y coordinate	(s2)+2n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2)).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data specified by (s2).

Precautions

- The search method and the number of searches vary depending on whether SM755 is on or off.

SM755	Search method	Number of searches
Off	Sequential search	1≤number of searches≤65535
On	Binary search	1≤number of searches≤16

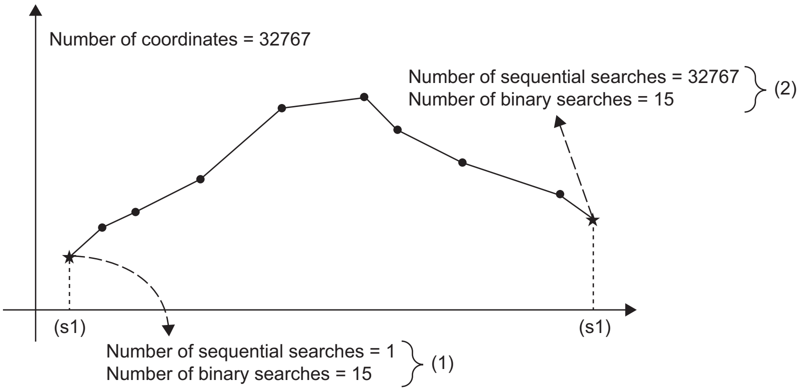
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

- Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s1) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

- Case in which the processing speed of binary search is faster

The maximum number of searches is 16 and therefore when coordinate point 17 or later is specified by (s1), the number of binary searches is equal to or greater than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the binary search is faster because the number of sequential searches is less than the number of binary searches.
(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

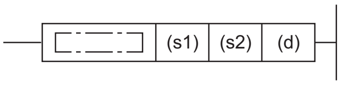
Error code (SD0)	Description
3405H	The X-coordinate data of the scaling conversion data before the point specified in (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.

Scaling 32-bit binary data (point coordinates)

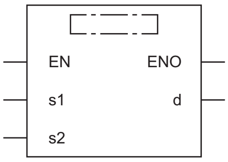
DSCL(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

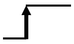
These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (point coordinates).

Ladder	ST	
	ENO:=DSCL(EN,s1,s2,d); ENO:=DSCLP(EN,s1,s2,d);	ENO:=DSCL_U(EN,s1,s2,d); ENO:=DSCLP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCL DSCL_U	
DSCLP DSCLP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCL(P)	Input value to be scaled or the start device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCL(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DSCL(P)	Start device containing scaling conversion data	—	32-bit signed binary[1]	ANY32_S[2]
	DSCL(P)_U			32-bit unsigned binary[1]	ANY32_U[2]
(d)	DSCL(P)	Start device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCL(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[1] The number of coordinate points in (s2)+0 and (s2)+1 is represented in 32-bit unsigned binary.

[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.

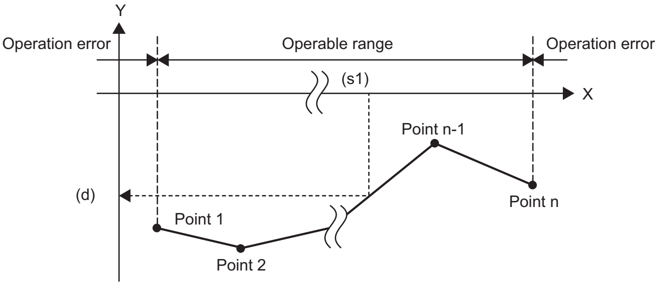
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)+1, (s2)
Point 1	X coordinate	(s2)+3, (s2)+2
	Y coordinate	(s2)+5, (s2)+4
Point 2	X coordinate	(s2)+7, (s2)+6
	Y coordinate	(s2)+9, (s2)+8
⋮		
Point n	X coordinate	(s2)+4n-1, (s2)+4n-2
	Y coordinate	(s2)+4n+1, (s2)+4n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2), (s2)+1).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data specified by (s2).

Precautions

- The search method and the number of searches vary depending on whether SM755 is on or off.

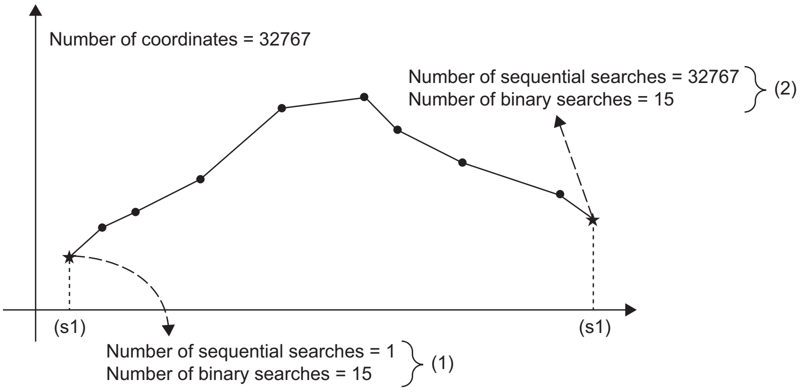
SM755	Search method	Number of searches
Off	Sequential search	1≤number of searches≤4294967295
On	Binary search	1≤number of searches≤32

- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.
  - Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s1) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

- Case in which the processing speed of binary search is faster

The maximum number of searches is 32 and therefore when coordinate point 33 or later is specified by (s1), the number of binary searches is equal to or greater than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the binary search is faster because the number of sequential searches is less than the number of binary searches.
(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

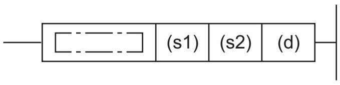
Error code (SD0)	Description
3405H	The X-coordinate data of the scaling conversion data before the point specified in (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 4294967295.

Scaling 16-bit binary data (XY coordinates)

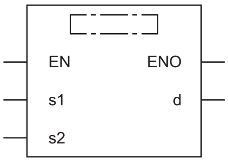
SCL2(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

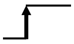
These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (XY coordinates).

Ladder	ST	
	ENO:=SCL2(EN,s1,s2,d); ENO:=SCL2P(EN,s1,s2,d);	ENO:=SCL2_U(EN,s1,s2,d); ENO:=SCL2P_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCL2 SCL2_U	
SCL2P SCL2P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCL2(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCL2(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	SCL2(P)	Start device containing scaling conversion data	—	16-bit signed binary[1]	ANY16_S[2]
	SCL2(P)_U			16-bit unsigned binary[1]	ANY16_U[2]
(d)	SCL2(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCL2(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[1] The number of coordinate points in (s2) is represented in 16-bit unsigned binary.

[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.

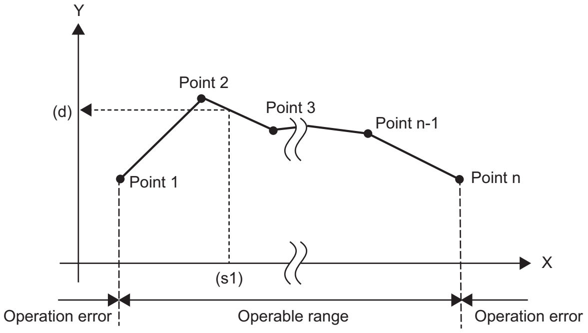
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		**Device assignment**
Number of coordinate points		(s2)
X coordinate	Point 1	(s2)+1
	Point 2	(s2)+2
	⋮	⋮
	Point n	(s2)+n
Y coordinate	Point 1	(s2)+n+1
	Point 2	(s2)+n+2
	⋮	⋮
	Point n	(s2)+2n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2)).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. For details, refer to the SCL(P)(_U) instruction.

☞ Page 869 SCL(P)(_U)

Operation error

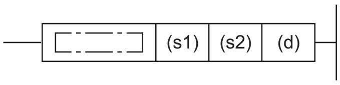
Error code (SD0)	**Description**
3405H	The X-coordinate data is not sorted in ascending order.
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.

Scaling 32-bit binary data (XY coordinates)

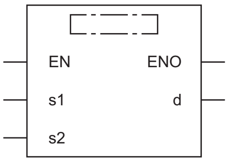
DSCL2(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

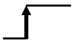
These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (XY coordinates).

Ladder	ST	
	ENO:=DSCL2(EN,s1,s2,d); ENO:=DSCL2P(EN,s1,s2,d);	ENO:=DSCL2_U(EN,s1,s2,d); ENO:=DSCL2P_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCL2 DSCL2_U	
DSCL2P DSCL2P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCL2(P)	Input value to be scaled or the start device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCL2(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DSCL2(P)	Start device containing scaling conversion data	—	32-bit signed binary[*1]	ANY32_S[*2]
	DSCL2(P)_U			32-bit unsigned binary[*1]	ANY32_U[*2]
(d)	DSCL2(P)	Start device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCL2(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[*1] The number of coordinate points in (s2) to (s2)+1 is represented in 32-bit unsigned binary.

[*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.

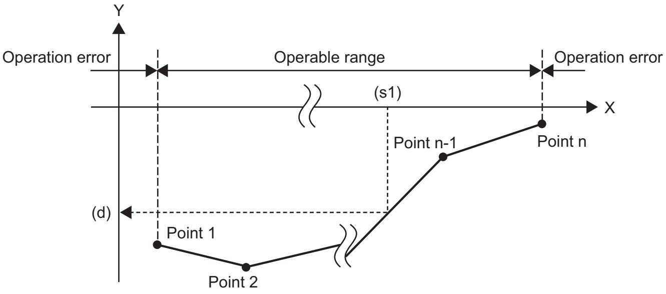
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		**Device assignment**
Number of coordinate points		(s2)+1, (s2)
X coordinate	Point 1	(s2)+3, (s2)+2
	Point 2	(s2)+5, (s2)+4
	⋮	⋮
	Point n	(s2)+2n+1, (s2)+2n
Y coordinate	Point 1	(s2)+2n+3, (s2)+2n+2
	Point 2	(s2)+2n+5, (s2)+2n+4
	⋮	⋮
	Point n	(s2)+4n+1, (s2)+4n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2) to (s2)+1).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. For details, refer to the DSCL(P)(_U) instruction.

☞ Page 872 DSCL(P)(_U)

Operation error

Error code (SD0)	**Description**
3405H	The X-coordinate data is not sorted in ascending order.
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 4294967295.

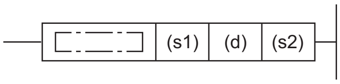
7.13 Special Counter Instructions

Counting up or down the current value (1-phase input)

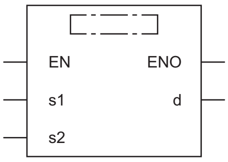
UDCNT1

RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction updates the current value of the specified counter.

Ladder	ST
	ENO:=UDCNT1(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDCNT1	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	(s1)+0: Count input number	—	Bit	ANYBIT_ARRAY[1] (Number of elements: 2)
	(s1)+1: Count up/down flag Off indicates count-up (counting up the current value). On indicates count-down (counting down the current value).			
(d)	Number of the counter (device name) to be counted by the UDCNT1 instruction	—	Device name	ANY16[2]
(s2)	Set value	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (X) can be used.

*2 Only labels assigned to device (C) can be used.

■Applicable devices

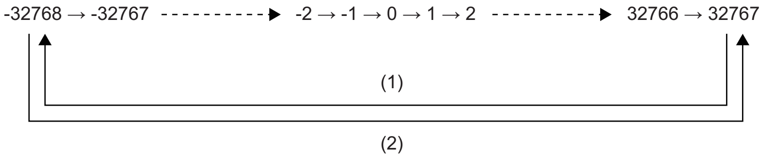
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○[1]	—	—	—	—	—	—	—	—	—	—	—
(d)	—	—	○[2]	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only X can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).

*2 Only C can be used.

Processing details

- When the input specified by (s1) is turned on, this instruction updates the current value of the counter specified by (d).
- Counting up or down is determined by whether the input specified by (s1)+1 is on or off.
  - Off: Count-up (counting up the current value)
  - On: Count-down (counting down the current value)
- Count processing is performed as follows.
  - When the current value equals the value specified by (s2) during count-up, the contact of the counter specified by (d) is turned on. The current value is kept counting even when the contact of the counter is turned on.
  - When the current value equals "the set value - 1" during countdown, the contact of the counter specified by (d) is turned off.
  - The counter specified by (d) is a ring counter. Counting up the counter when the current value is 32767 proceeds to -32768. Similarly, counting down the counter when the current value is -32768 proceeds to 32767. The following figure shows the processing for counting the current value.



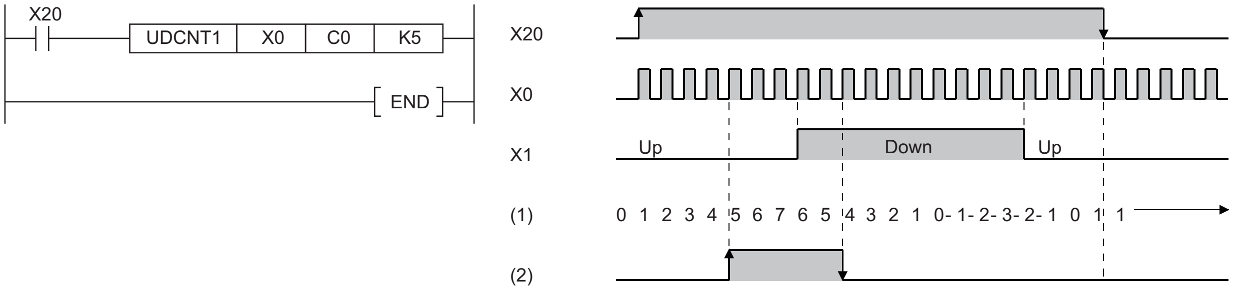
(1) When counting up

(2) When counting down

- The following figure shows the operation of count processing using the UDCNT1 instruction.

Ex.

Program which uses C0 (up/down counter) to count the number of times X0 turns off and on after X20 turns on



(1) Current value of C0

(2) Contact of C0

- When executed, the UDCNT1 instruction starts counting when the execution command turns on and stops counting when the command turns off. If the execution command is turned on again, the instruction resumes counting from the current value with which it stopped counting previously.
- The RST instruction is used to clear the current value of the counter specified by (d) and turn off the contact.

Point

- The UDCNT1 instruction stores the device data of the argument in the work area of the CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The set value cannot be changed during counting by the UDCNT1 instruction (while the execution command is on). To change the set value, turn off the execution command in advance.
- The counter specified by the UDCNT1 instruction cannot be used by any other instruction. If another instruction uses it, normal counting is disabled.
- The UDCNT1 instruction can be used a maximum of six times in all running programs. The seventh or subsequent UDCNT1 instruction, if issued, causes no processing.

Operation error

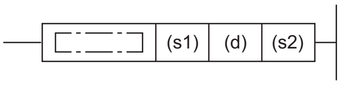
There is no operation error.

Counting up or down the current value (2-phase input)

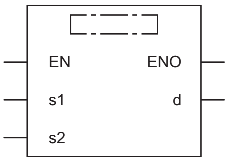
UDCNT2

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction updates the current value of the counter depending on the status of phases A and B pulses.

Ladder	ST
	ENO:=UDCNT2(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDCNT2	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	(s1)+0: Count input number (phase A pulse)	—	Bit	ANYBIT_ARRAY*1 (Number of elements: 2)
	(s1)+1: Count input number (phase B pulse)			
(d)	Number of the counter (device name) to be counted by the UDCNT2 instruction	—	Device name	ANY16*2
(s2)	Set value	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (X) can be used.

*2 Only labels assigned to device (C) can be used.

■Applicable devices

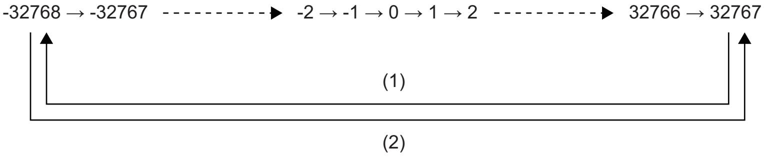
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	—	—	—	—	—	—	—	—	—	—	—
(d)	—	—	○*2	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only X can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).

*2 Only C can be used.

Processing details

- This instruction updates the current value of the counter specified by (d) depending on the status of the input (phase A pulse) specified by (s1) and the status of the input (phase B pulse) specified by (s1)+1.
- Counting up or down is determined as follows.
  - (s1)+1 is turned on while (s1) is on: Count-up (counting up the current value)
  - (s1)+1 is turned off while (s1) is on: Countdown (counting down the current value)
  - The instruction does not count while (s1) is off.
- Count processing is performed as follows.
  - When the current value equals the value specified by (s2) during count-up, the contact of the counter specified by (d) is turned on. The current value is kept counting even when the contact of the counter is turned on.
  - When the current value equals "the set value - 1" during countdown, the contact of the counter specified by (d) is turned off.
  - The counter specified by (d) is a ring counter. Counting up the counter when the current value is 32767 proceeds to -32768. Similarly, counting down the counter when the current value is -32768 proceeds to 32767. The following figure shows the processing for counting the current value.



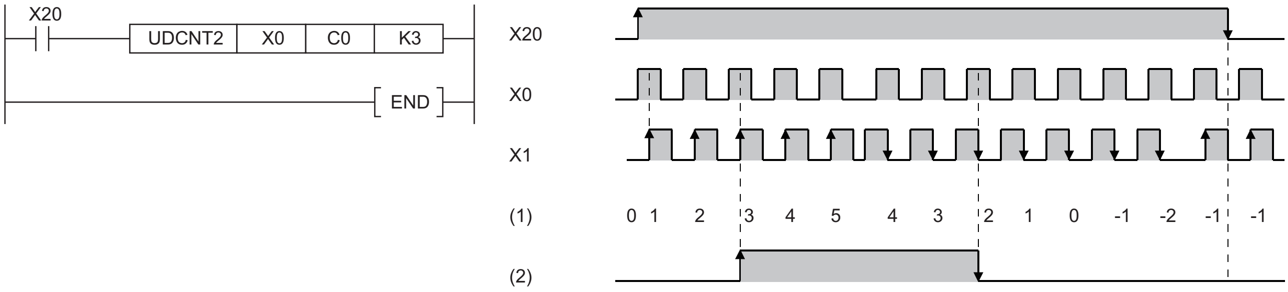
(1) When counting up

(2) When counting down

- The following figure shows the operation of count processing using the UDCNT2 instruction.

Ex.

Program which uses C0 (up/down counter) to count the states of X0 and X1 after X20 turns on



(1) Current value of C0

(2) Contact of C0

- When executed, the UDCNT2 instruction starts counting when the execution command turns on and stops counting when the command turns off. If the execution command is turned on again, the instruction resumes counting from the current value with which it stopped counting previously.
- The RST instruction is used to clear the current value of the counter specified by (d) and turn off the contact.

Point

- The UDCNT2 instruction stores the device data of the argument in the work area of the CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The set value cannot be changed during counting by the UDCNT2 instruction (while the execution command is on). To change the set value, turn off the execution command in advance.
- The counter specified by the UDCNT2 instruction cannot be used by any other instruction. If another instruction uses it, normal counting is disabled.
- The UDCNT2 instruction can be used a maximum of five times in all running programs. The sixth or subsequent UDCNT1 instruction, if issued, causes no processing.

Operation error

There is no operation error.

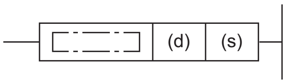
7.14 Special Timer Instructions

Teaching timer

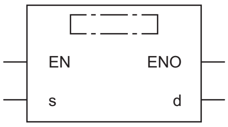
TTMR

RnCPU RnENCPU RnPCPU (Process) ✕RnPCPU (Redundant) RnSFCPU (Standard) ✕RnSFCPU (Safety)

This instruction measures the on time of the measurement command in seconds, multiplies it by a multiplier, and stores the operation result.

Ladder	ST
	ENO:=TTMR(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TTMR	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(d)	(d)+0: Device for storing the measurement value (d)+1: Device for the system of CPU module	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(s)	Multiplier of measurement value	0 to 2	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s)	—	○	○	○	○	—	—	○	○	—	—	—

■Control data

Operand: (d)

Device	Description	Setting range	Set by
+0	Device for storing the measurement value	—	System
+1	Device for the system of CPU module	—	System

Processing details

- This instruction measures the on time of the execution command in seconds, multiplies it by the multiplier specified by (s), and stores the resultant value in the device specified by (d).
- When the execution command is turned on, the instruction clears the device specified by (d)+0, (d)+1.
- The table below lists the multipliers that can be specified by (s).

(s)	Multiplier
0	1
1	10
2	100

Point

- When executed, the TTMR instruction implements time measurement. Do not use the JMP instruction to skip the TTMR instruction. Otherwise, accurate measurement is disabled.
- Do not change the multiplier specified by (s) during execution of the TTMR instruction. Otherwise, accurate values cannot be determined.
- The device specified by (d)+1 is used by the system of the CPU module. Do not change the value. If the value is changed, an accurate resultant value is not stored in the device specified by (d).

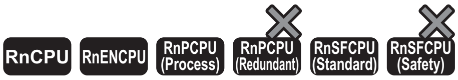
- When the value in the device specified by (s) is not in the range from 0 to 2, no processing is performed.

Operation error

There is no operation error.

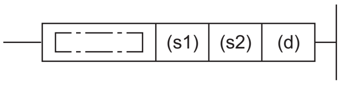
Special function timer

STMR

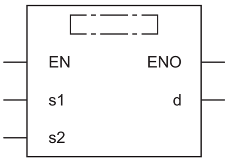


This instruction implements the following four types of timer output.

- Off delay timer output
- After-off one-shot timer output
- After-on one-shot timer output
- On delay + off delay timer output

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

FBD/LD



■Execution condition

Instruction	Execution condition
STMR	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Timer device or timer type label	—	Device name	ANY16
(s2)	Set value	0 to 32767	16-bit signed binary	ANY16
(d)	(d)+0: Off delay timer output	—	Bit	ANYBIT_ARRAY (Number of elements: 4)
	(d)+1: After-off one-shot timer output			
	(d)+2: After-on one-shot timer output			
	(d)+3: On delay + off delay timer output			
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○[*1]	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	—	—	—	—	—	—	—	—	—	—

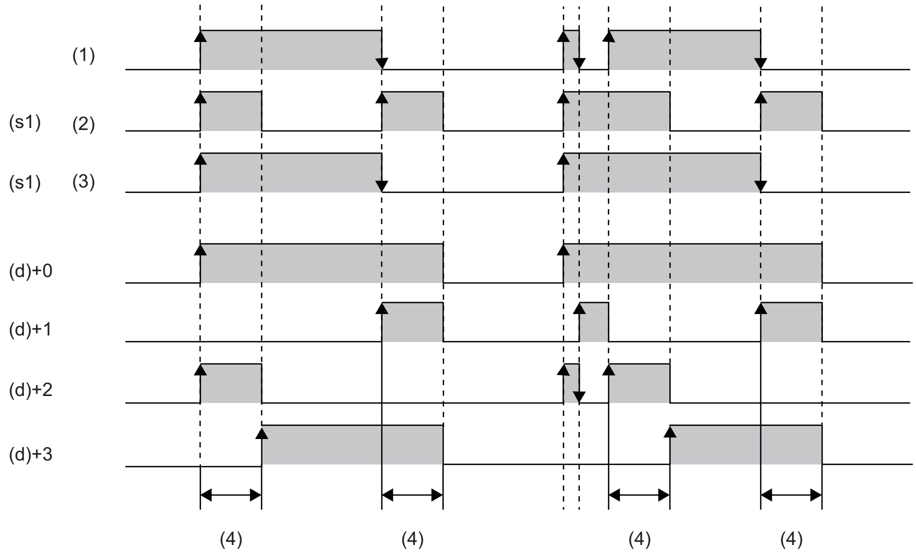
[*1] Only T can be used.

Processing details

- This instruction uses four points from the device specified by (d) to implement four types of timer output.

Setting data		Description	
(d)	+0	Off delay timer output:	Turns on on the rising edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2) after the falling edge of the command.
	+1	After-off one-shot timer output	Turns on on the falling edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2).
	+2	After-on one-shot timer output	Turns on on the falling edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2) or when the command of the instruction turns off.
	+3	On delay + off delay timer output	Turns on on the falling edge of the timer coil, and turns off after a lapse of the time specified by (s2) after the falling edge of the command of the STMR instruction.

- The coil of the timer specified by (s) turns on, on the rising and falling edges of the command of the STMR instruction, to start measurement of the current value.
- The coil of the timer keeps measurement during the time specified by (s2) and turns off when the time is up.
- The coil of the timer is kept on even if the STMR instruction is turned off before time-up. Timer measurement is continued. When the STMR instruction is turned on again, the coil resets the current value to 0 and restarts measurement.
- The contact of the timer turns on, on the rising edge of the command of the STMR instruction, and turns off on the falling edge of the command after the coil of the timer falls. Users cannot use the contact of the timer because it is reserved for the system.



(1) Command for the STMR instruction

(2) Coil

(3) Contact

(4) Value specified by (s2)

- Measurement of the current value of the timer specified by the STMR instruction is executed regardless of whether the command of the STMR instruction is on or off. If the STMR instruction is skipped such as by the JMP instruction, normal measurement is not performed.
- The measurement unit of the timer specified by (d) is the same as that of the low-speed timer.
- A value from 0 to 32767 can be specified in (s2). If a value out of the range is specified, no processing is performed.
- Do not use the OUT instruction for the timer specified by (s1). If the same timer device or timer type label is used for the STMR and OUT instructions, normal operation is not performed.

Precautions

If there is an STMR instruction within the range for changing the ladder block online or writing data to the running programmable controller, the STMR instruction is executed.

For details, refer to the following.

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

Operation error

There is no operation error.

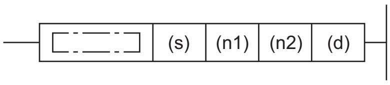
7.15 Shortcut Control Instruction

Rotary table shortest direction control

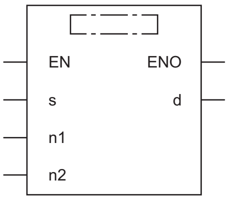
ROTC

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✗

This instruction controls shortcut rotation on the rotary table divided equally by the specified value.

Ladder	ST
	ENO:=ROTC(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ROTC	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	(s)+0: Device for measuring the number of table rotations (reserved for the system)	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
	(s)+1: Call counter number		16-bit unsigned binary	
	(s)+2: Call item number		16-bit unsigned binary	
(n1)	Number of table divisions	2 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of low-speed sections	0 to less than (n1)	16-bit unsigned binary	ANY16
(d)	(d)+0: Phase A input signal	—	Bit	ANYBIT_ARRAY (Number of elements: 8)
	(d)+1: Phase B input signal			
	(d)+2: 0-point detection input signal			
	(d)+3: High-speed forward rotation output signal (reserved for the system)			
	(d)+4: Low-speed forward rotation output signal (reserved for the system)			
	(d)+5: Stop output signal (reserved for the system)			
	(d)+6: Low-speed reverse rotation output signal (reserved for the system)			
	(d)+7: High-speed reverse rotation output signal (reserved for the system)			
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	—	—	—	—	—	—	—	—	—	—

Processing details

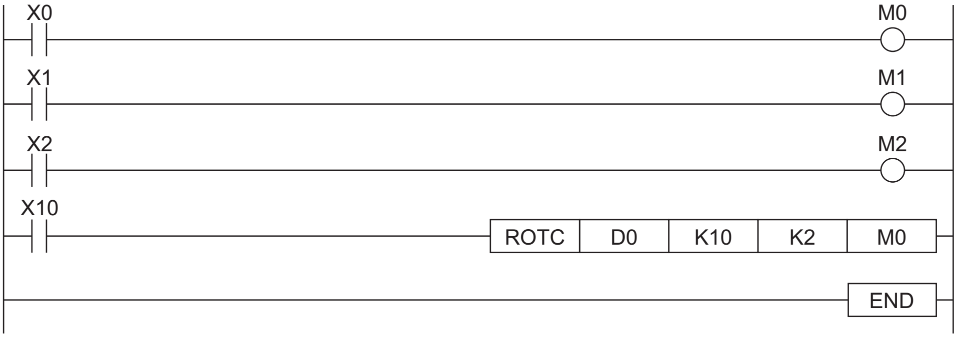
- This instruction controls the rotation of the rotary table divided equally by the value specified by (n1) so that it rotates at short cut to the position of the counter number specified by (s)+1 to get in and out the item of the number specified by (s)+2.
- It performs control by assuming that the item numbers and counter numbers are assigned counterclockwise.
- (s)+0 is the counter used for the system to count items to determine which item is in the 0th counter. Do not rewrite the data with the program. Otherwise, accurate control cannot be performed.
- The value specified in (n2) must be less than the number of table divisions specified by (n1).
- (d)+0 and (d)+1 are the phase A input signal and phase B input signal used to detect the forward and reverse rotations of the rotary table. The direction of rotation is determined by whether phase B is on the rising or falling edge when phase A is on.
  - Phase B is on the rising edge: Forward rotation (clockwise)
  - Phase B is on the falling edge: Reverse rotation (counterclockwise)
- (d)+2 is the 0-point detection signal that turns on when the 0th item reaches the 0th counter. When the device specified by (d)+2 turns on during execution of the ROTC instruction, the device specified by (s)+0 is cleared. Start shortcut control with the ROTC instruction after performing this clearing operation.
- (d)+3 to (d)+7 are output signals for controlling table operations. One of the output signals in (d)+3 to (d)+7 is turned on according to the execution result of the ROTC instruction.
- When the command of the ROTC instruction is off, shortcut control is not performed and (d)+3 to (d)+7 are all turned off.
- The ROTC instruction can be used only once in all running programs. If it is used more than once, normal operation cannot be performed.
- If the value in (s)+0 to (s)+2 or (n2) is greater than (n1), no processing is performed.

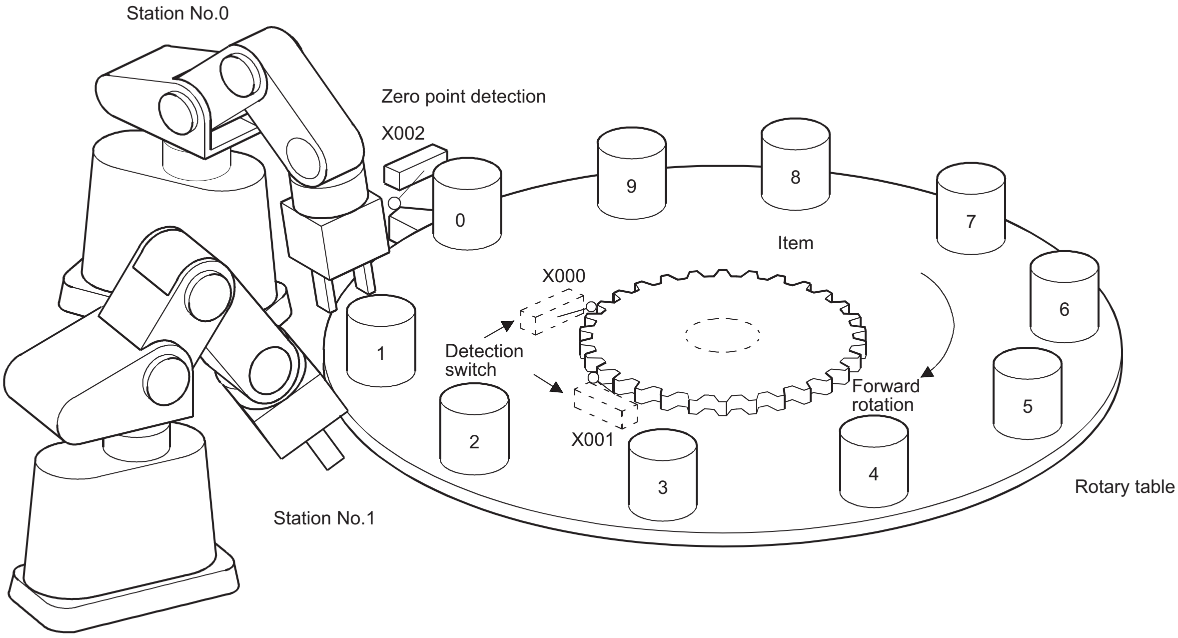
■Program example

When getting in and out at counters, the items placed on the rotary table divided into 10 sections, this program controls the rotary table so that the items rotate at short cut.

The item number is specified by D2, and the counter number is specified by D1.

The table is rotated at low speeds in two front and rear sections.





Operation error

There is no operation error.

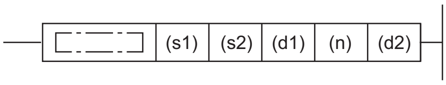
7.16 Ramp Signal Instruction

Ramp signal

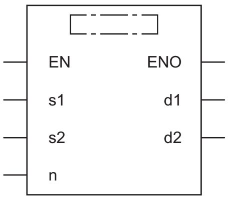
RAMPQ

RnCPU **RnENCPU** **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

This instruction shifts from a specified value to another specified value in (n) times.

Ladder	ST
	ENO:=RAMPQ(EN,s1,s2,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
RAMPQ	⎍

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Initial value	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Last value	-32768 to 32767	16-bit signed binary	ANY16
(d1)	(d1)+0: Current value	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
	(d1)+1: Number of executions			
(n)	Number of shifts	1 to 32767	16-bit signed binary	ANY16
(d2)	(d2)+0: Completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
	(d2)+1: Bit for selecting data retention at completion			
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(d2)	○	—	—	—	—	—	—	—	—	—	—	—

Processing details

- When the execution command is on, this instruction performs processing as follows.
  - Shifting from the value specified by (s1) to the value specified by (s2) in the number of times specified by (n).
  - For (n), specify the number of scans (number of shifts) to be performed to shift from (s1) to (s2). If the value specified in (n) is out of the range between 0 and 32768, no processing is performed.
  - (d)+1 is used for the system to store the number of times the RAMPQ instruction has been executed.
  - The change value per scan is calculated by the following equation.

$$Cv = \frac{(s2) - (s1)}{(n)}$$

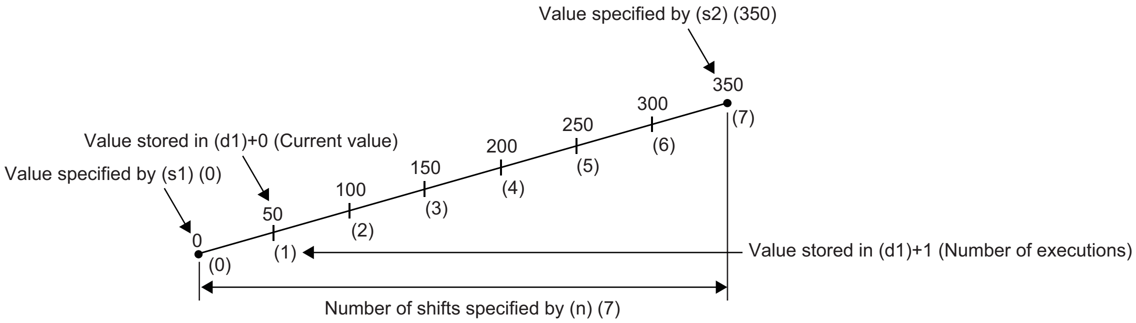
Cv: Amount of change in 1 scan

(s2): Value specified by (s2)

(s1): Value specified by (s1)

(n): Value specified by (n)

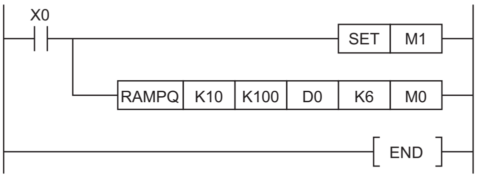
- The following figure shows how to change values from 0 to 350 in seven scans.

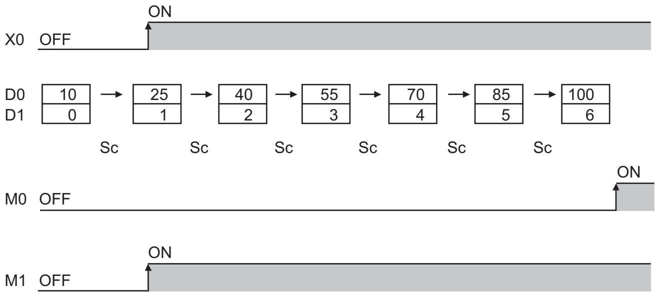


- If the change value in one scan is indivisible, correct it so that it becomes the value specified by (s2) in the number of shifts specified by (n). For this reason, a linear ramp may not be created.
- The following figure shows the operation of processing using the RAMPQ instruction.

Ex.

Program which, when X0 turns on, changes the content of D0 from 10 to 100 in six scans and holds the content of D0 when the change is completed





Sc: 1 scan

- After scanning is performed the number of shifts specified by (n), the completion device specified by (d2)+0 turns on. The on/off status of the completion device and the data in (d1)+0 are determined by on/off of the device specified by (d2)+1. When (d2)+1 is off, the RAMPQ instruction turns off (d2)+0 in the next scan and restarts shifting from the initial value. When (d2)+1 is on, (d2)+0 is kept on and the data in (d1)+0 remains unchanged.
- If the command turns off during execution of the RAMPQ instruction, the data in (d1)+0 will not change thereafter. When the command turns on again, the RAMPQ instruction restarts shifting from the initial value.
- Do not change the values in (s1) and (s2) before the completion device specified by (d2)+0 turns on. The value to be stored in (d1)+1 is calculated using the same calculation formula every scan, and therefore changing the values in (s1) and (s2) may result in a sudden change.
- When making the digit specification using a bit device in (d1), specify it in K8Dn format.

Precautions

When the digit specification is made using a bit device in (d1), it is acceptable only when the number of digits is specified in K8.

Operation error

There is no operation error.

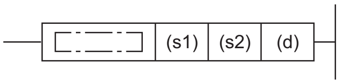
7.17 Pulse Related Instructions

Measuring the density of pulses

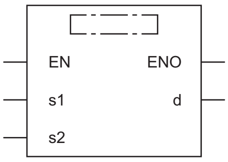
SPD

RnCPU **RnENCPU** **RnPCPU (Process)** ~~**RnPCPU (Redundant)**~~ **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

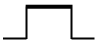
This instruction counts the device input only for the specified time.

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

FBD/LD



■Execution condition

Instruction	Execution condition
SPD	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Pulse input device number	—	Bit	ANY_BOOL[1]
(s2)	Measurement time, or the device number of the device containing the measurement time (unit: ms)	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the measurement result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Only labels assigned to device (X) can be used.

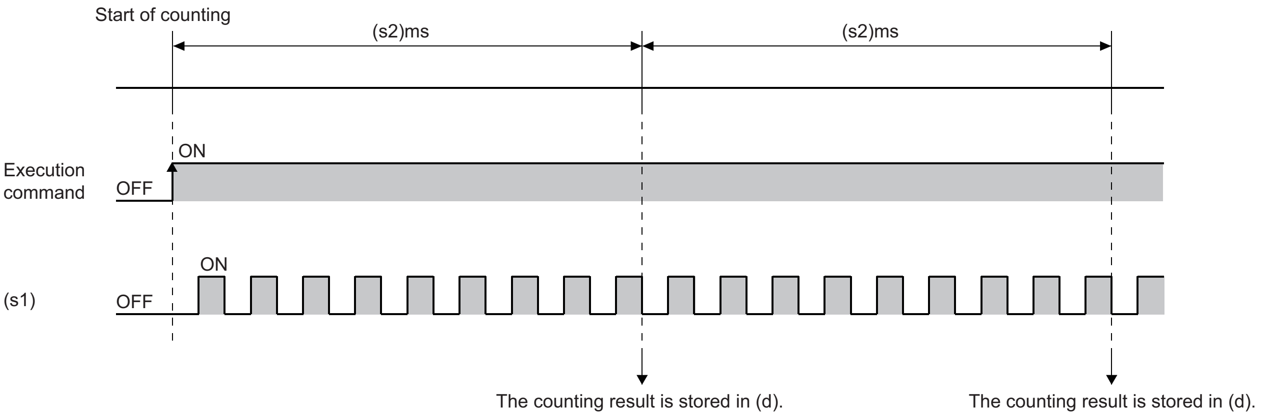
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○[1]	—	—	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

[1] Only X can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).

Processing details

- This instruction counts the number of times the input of the device specified by (s1) turns on for the duration specified by (s2), and stores the count result in the device specified by (d).



- Upon completion of measurement, the SPD instruction starts measurement from 0 again. To stop measurement by the SPD instruction, turn off the execution command.
- If the value specified in (s2) is 0, no processing is performed.

Operation error

There is no operation error.

Point

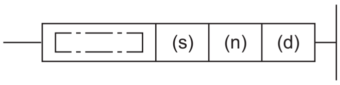
- The SPD instruction stores the data of the argument device in the work area of the CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The SPD instruction can be used a maximum of six times in all running programs. The seventh or subsequent UDCNT1 instruction, if issued, causes no processing.
- The set value cannot be changed during measurement by the SPD instruction (while the command input is on). To change the set value, turn off the command input in advance.

Outputting pulses at regular intervals

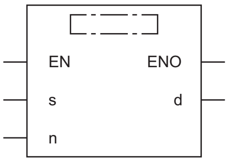
PLSY

RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction outputs the pulses of the specified frequency to the output module.

Ladder	ST
	ENO:=PLSY(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PLSY	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Frequency, or the start number of the device containing the frequency	1 to 100	16-bit signed binary	ANY16[*1]
(n)	Number of outputs, or the start number of the device containing the number of outputs	0 to 65535	16-bit unsigned binary	ANY16
(d)	Device used for pulse output	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (Y) can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○[*1]	—	—	—	—	—	—	—	—	—	—	—

*1 Only Y can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).

Processing details

- This instruction outputs the pulses at the frequency specified by (s), by the number of times specified by (n), to the output module with the output number (Y) in the device specified by (d).
- A frequency from 1Hz to 100Hz can be specified in (s). If the specified value in (s) is not in the range from 1 to 100, no processing is performed.
- A value from 0 to 65535 (0000H to FFFFH) can be specified for the number of outputs in (n). If 0 is specified in (n), pulses are output continuously.
- Only the output number (Y) corresponding to the output module can be specified for the pulse output in (d).
- The PLSY instruction starts pulse output on the rising edge of the command. When the command turns off, the PLSY instruction stops pulse output.

Operation error

There is no operation error.

Point

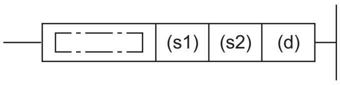
- The PLSY instruction stores the device data of the argument in the work area of the CPU module, and performs the actual output operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be out must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- Do not change the argument of the PLSY instruction during pulse output by the instruction (the execution command is on). To change the argument, turn off the execution command in advance.
- The PLSY instruction can be used only once in all programs running in the CPU module. The second or subsequent PLSY instruction, if issued, causes no processing.

Performing the pulse width modulation

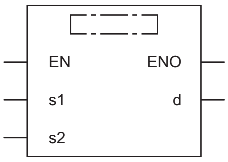
PWM

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) RnSFCPU (Safety) ✗

When on continues for the specified time, this instruction outputs the pulse of the period to the output module.

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

FBD/LD



■Execution condition

Instruction	Execution condition
PWM	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	On time, or the start number of the device containing the on time (Unit: ms)	1 to 65535	16-bit unsigned binary	ANY16[1]
(s2)	Period, or the start number of the device containing the period (Unit: ms)	1 to 65535	16-bit unsigned binary	ANY16
(d)	Pulse output device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Only labels assigned to device (Y) can be used.

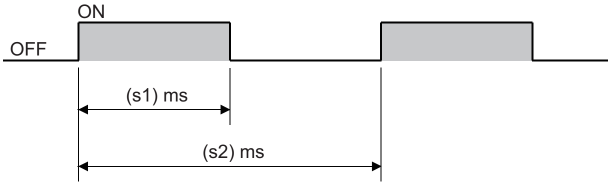
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○[1]	—	—	—	—	—	—	—	—	—	—	—

[1] Only Y can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).

Processing details

- This instruction outputs the on time specified by (s1) and the pulse of the period specified by (s2) to the output module specified by (d).



- A value in the range from 1 to 65535 (0001H to FFFFH) can be specified in (s1) and (s2). (The value specified in (s1) must be less than the value specified in (s2).)
- No processing is performed in the following cases.
  - (s1) and (s2) are 0.
  - (s1)≥(s2)
  - The PWM instruction is executed more than once.

Operation error

There is no operation error.

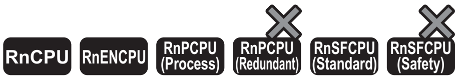
Point

- The PWM instruction stores the device data of the argument in the work area of the CPU module, and performs the actual output operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) The interrupt interval of the CPU module is 1ms. The PWM instruction can be used only once in all programs running in the CPU module.
- Do not change the argument of the PWM instruction during pulse output by the instruction (the execution command is on). To change the argument, turn off the execution command in advance.

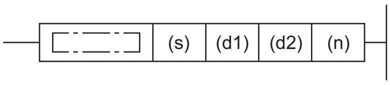
7.18 Matrix Input Instruction

Matrix input

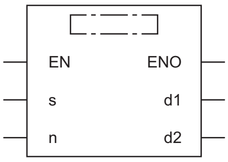
MTR



This instruction sequentially reads the input of 16 points×n columns connected to the specified input number and after.

Ladder	ST
	ENO:=MTR(EN,s,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
MTR	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device of input	—	Bit	ANY_BOOL[*1*3]
(d1)	Start device of output	—	Bit	ANY_BOOL[*2*3]
(d2)	Start device for storing the matrix input data	—	Bit	ANY_BOOL[*3]
(n)	Number of input columns	2 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (X) can be used.

*2 Only labels assigned to device (Y) can be used.

*3 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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[*1]	—	—	—	—	—	—	—	—	—	—	—
(d1)	○[*2]	—	—	—	—	—	—	—	—	—	—	—
(d2)	○	—	—	—	—	—	—	—	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only X can be used.

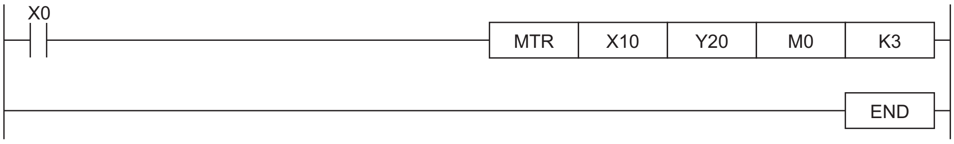
*2 Only Y can be used.

Processing details

- This instruction sequentially reads the input of 16 points×n columns connected to the (s) specified input number and after, and stores the input data that has been read in the device specified by (d2) and later.
- One scan reads one column (16 points) of data.
- The instruction sequentially repeats the reading of data from column 1 to column (n).
- In the device specified by (d2) and later, the data in column 1 is stored in the 16 points from the start and the data in column 2 is stored in the next 16 points. For this reason, the MTR instruction occupies 16×(n) points from the device specified by (d2).
- (d1) is the output for selecting the column to be read and is turned on and off automatically by the system. The (n) points from the device specified by (d1) is used.
- Only a device number which is a multiple of 16 can be specified in (s), (d1), and (d2).
- A value from 2 to 8 can be specified in (n).
- No processing is performed in the following cases.
  - The device number specified by (s), (d1), or (d2) is not a multiple of 16.
  - The device specified by (s) is outside the range of actual inputs.
  - The device specified by (d1) is outside the range of actual outputs.
  - In the device specified by (d2) and later, 16×(n) points of data is outside the range of the relevant device.
  - (n) is outside the range from 2 to 8.

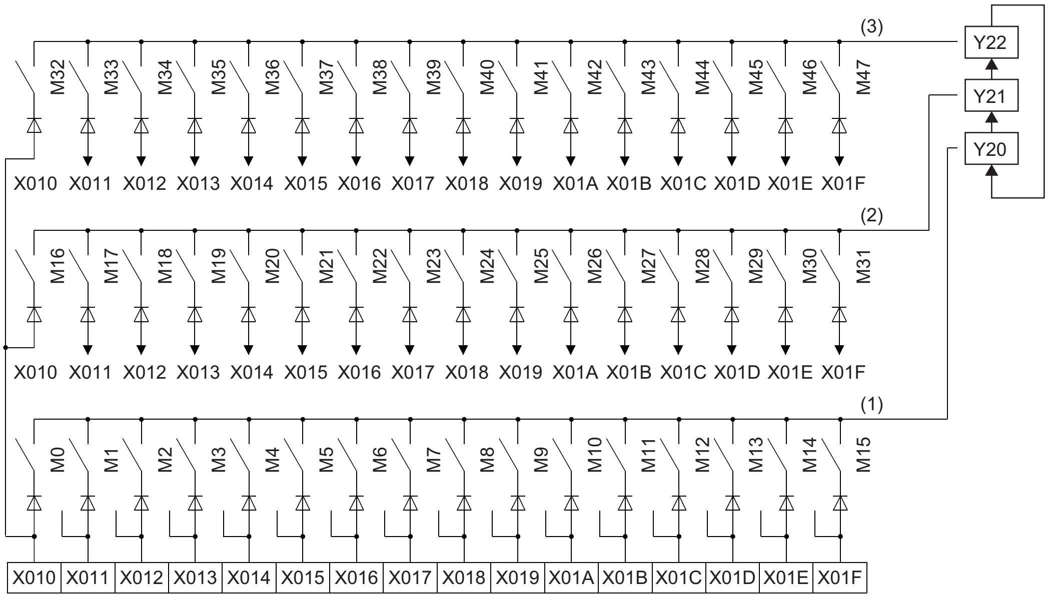
■Program example

A program that reads the matrix data(16 points × three columns) connected to X10 and later, and stores the read data to M0 and later when X0 turns on



[Operation]

- When Y20 turns on, the instruction reads the input signals of the 1st column, and stores the read data to M0 to M15.
- When Y21 turns on, the instruction reads the input signals of the 2nd column, and stores the read data to M16 to M31.
- When Y22 turns on, the instruction reads the input signals of the 3rd column, and stores the read data to M32 to M47.



(1) 1st column

(2) 2nd column

(3) 3rd column

Precautions

- Note that the MTR instruction directly operates the actual input/output. Even when the command of the MTR instruction turns off, the output that has been turned on by the MTR instruction is not turned off. Turn off the output specified by (d1) in the program.
- The MTR instruction execution interval should be longer than the total response time of the input and output modules. If the MTR instruction execution interval is shorter than the above time, inputs cannot be read normally. If the scan time in the program is short, select the constant scan and set longer scan time than the total of the response time.

Operation error

Error code (SD0)	Description
2820H	A device other than the input (X) is specified by (s).
	A device other than the output (Y) is specified by (d1).

7.19 Check Code Instructions

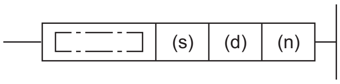
Check code

CCD(P)

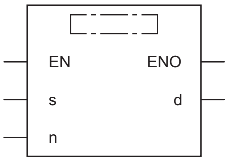
RnCPU RnENCPU ✕RnPCPU (Process) ✕RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

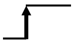
These instructions perform addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1.

Ladder	ST
	ENO:=CCD(EN,s,n,d); ENO:=CCDP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CCD	
CCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where target data is stored	—	16-bit signed binary	ANY16
(d)	Start device for storing the calculated data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of data	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[*1]	—	○	—	—	—	—	○	—	—	—	—
(d)	○[*1]	—	○	—	—	—	—	○	—	—	—	—
(n)	○[*1]	○	○	○	○	—	—	○	○	—	—	—

[*1] FX and FY cannot be used.

Processing details

- These instructions perform addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1. The 16-bit conversion mode and 8-bit conversion mode are available for these instructions. The conversion mode can be selected by turning on or off SM772.
- If the value specified in (n) is 0, no processing is performed.
- If an overflow occurs in the calculated result stored in (d), the following operations are performed. SM700 does not turn on.

32767+2=-32767 (7FFFH+0002H=8001H)	-32767+-2=32767 (8001H+FFFEH=7FFFH)

- These instructions calculate the horizontal parity value and sum check value as the error check methods used in communications. There is another check method called CRC (cyclic redundancy check) in addition to the CCD(P) instruction. (☞ Page 944 CRC(P))
- The operation in each conversion mode is described below.

■16-bit conversion mode (while SM772 is OFF)

With regard to (n) data points starting from (s), the addition data and horizontal parity data of upper 8 bits and lower 8 bits are stored to (d) and (d)+1 respectively.

Ex.

When (n)=6

<Calculation of addition data value>

In 16-bit conversion mode, addition data is determined by adding 6 bytes in the following shaded portion. The addition data is thus determined as "0315H", and therefore "0315H" is stored in the device specified by (d).

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

<Calculation of horizontal parity value>

In 16-bit conversion mode, the above shaded portion becomes the horizontal parity calculation target.

The number of ON (1) bits is calculated to determine the parity value which becomes ON (1) when the number of ON (1) bits is finally odd or OFF (0) when it is finally even. The horizontal parity value is stored in the device specified by (d)+1.

In the following table, 5FH is stored in the device specified by (d)+1.

Bit/horizontal parity value	b7	b6	b5	b4	b3	b2	b1	b0	Value
Upper 8 bits of D0	0	1	1	0	0	0	0	1	61H
Lower 8 bits of D0	0	1	1	0	0	1	0	0	64H
Upper 8 bits of D1	0	0	0	1	0	0	0	0	10H
Lower 8 bits of D1	0	1	1	1	1	0	1	1	7BH
Upper 8 bits of D2	1	1	1	1	1	0	1	0	FAH
Lower 8 bits of D2	1	1	0	0	1	0	1	1	CBH
Horizontal parity value	0	1	0	1	1	1	1	1	5FH

■8-bit conversion mode (while SM772 is ON)

With regard to (n) data points starting from (s), the addition data and horizontal parity data of only low-order 8 bits are stored to (d) and (d)+1 respectively.

Ex.

When (n)=6

<Calculation of addition data value>

In 8-bit conversion mode, addition data is determined by adding 6 bytes in the following shaded portion. The addition data is thus determined as "03B2H", and therefore "03B2H" is stored in the device specified by (d).

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

<Calculation of horizontal parity value>

In 8-bit conversion mode, the above shaded portion becomes the horizontal parity calculation target. The number of ON (1) bits is calculated to determine the parity value which becomes ON (1) when the number of ON (1) bits is finally odd or OFF (0) when it is finally even. The horizontal parity value is stored in the device specified by (d)+1.

In the following table, C2H is stored in the device specified by (d)+1.

Bit/horizontal parity value	b7	b6	b5	b4	b3	b2	b1	b0	Value
Lower 8 bits of D0	0	1	1	0	0	1	0	0	64H
Lower 8 bits of D1	0	1	1	1	1	0	1	1	7BH
Lower 8 bits of D2	1	1	0	0	1	0	1	1	CBH
Lower 8 bits of D3	1	1	1	1	1	1	1	1	FFH
Lower 8 bits of D4	1	1	1	1	1	0	0	1	F9H
Lower 8 bits of D5	0	0	0	1	0	0	0	0	10H
Horizontal parity value	1	1	0	0	0	0	1	0	C2H

Operation error

There is no operation error.

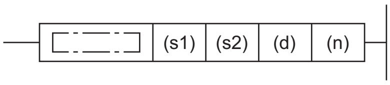
7.20 Data Processing Instructions

Searching 16-bit binary data

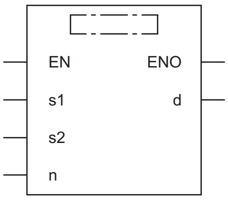
SERDATA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

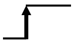
These instructions search the (n) points in the 16-bit binary data specified by (s2) for the 16-bit binary data specified by (s1).

Ladder	ST
	ENO:=SERDATA(EN,s1,s2,n,d); ENO:=SERDATAP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SERDATA	
SERDATAP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Search data or the device containing the search data	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Data to be searched or the start device containing the data to be searched	—	16-bit signed binary	ANY16[*1]
(d)	Start device for storing the search result	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of search target data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

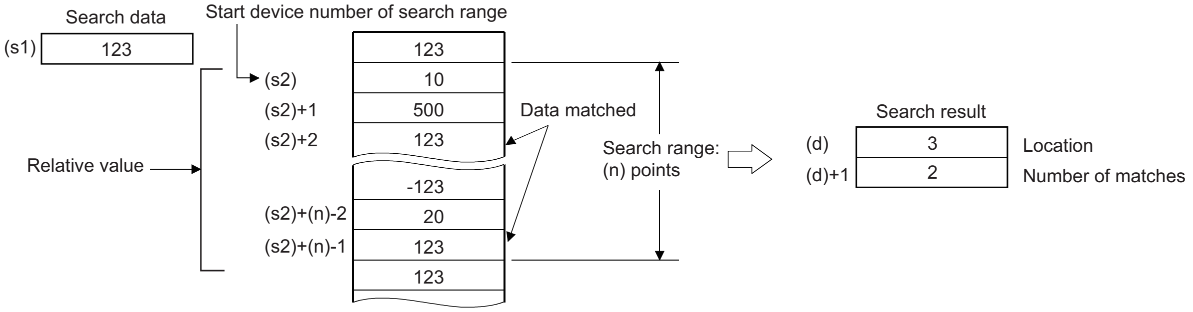
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

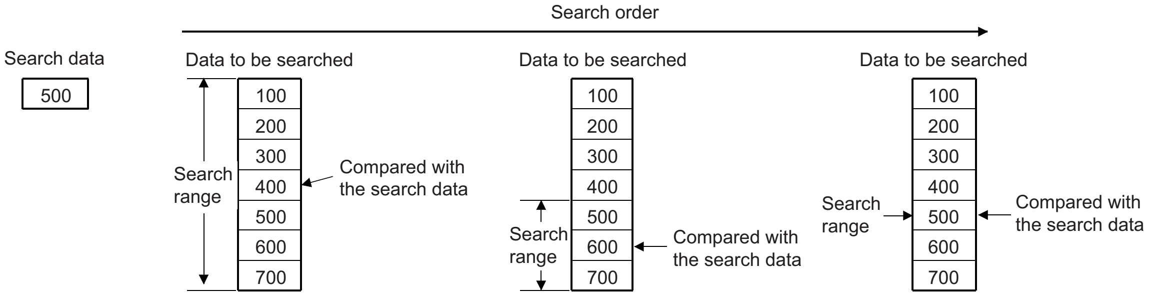
- These instructions search the (n) points in units of 16-bit binary data from the device specified by (s2) using the 16-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword in the device specified by (d)+1 and also stores the relative value of the first-matched device number from (s2) in the device specified by (d).



- If the value specified in (n) is 0, no processing is performed.
- If no matching data is found as the result of search, 0 is stored in the devices specified by (d) and (d)+1.

Point

- If the data to be searched by the SERDATA(P) instruction has been sorted in ascending order, turning on SM702[1] enables a binary search which can process the search faster. If SM702 is turned on even though the data to be search has not been sorted in ascending order, normal search results cannot be obtained. The following figure shows an example of binary search.



---

[1] SM702 is a special relay for setting the search method.
SM702 is off: Sequential search (linear search)
This method compares the search data with the data to be searched for starting from the start of data
SM702 is on: Binary search
For the data that has been sorted in ascending order, this method checks the center value of the search range, determining whether the center value is larger or smaller than the search value, and thereby narrows the search range to either side. Thus, target data is searched for by repeating this processing.

Operation error

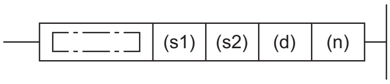
There is no operation error.

Searching 32-bit binary data

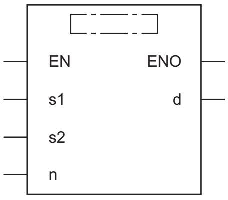
DSERDATA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

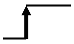
These instructions search the (n) points in the 32-bit binary data specified by (s2) for the 32-bit binary data specified by (s1).

Ladder	ST
	ENO:=DSERDATA(EN,s1,s2,n,d); ENO:=DSERDATAP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSERDATA	
DSERDATAP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Search data or the start device containing the search data	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Data to be searched or the start device containing the data to be searched	—	32-bit signed binary	ANY32[*1]
(d)	Start device for storing the search result	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of search target data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

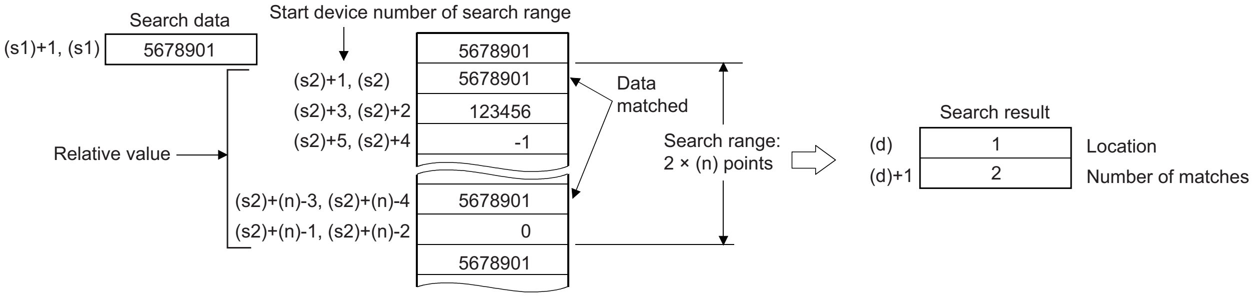
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

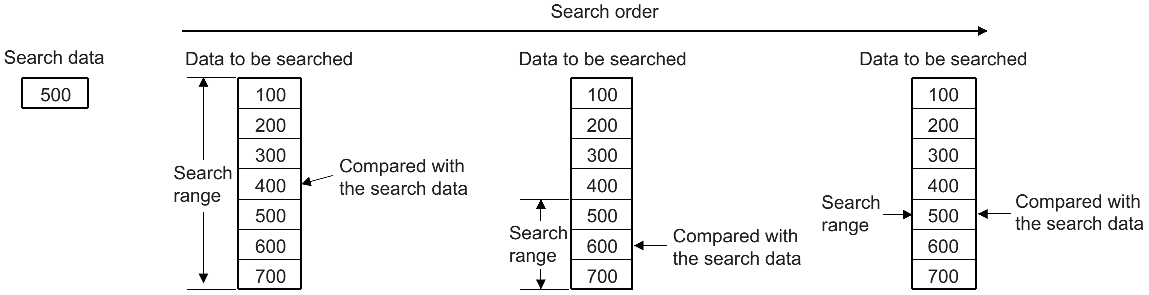
- These instructions search the (n) points of data in units of 32-bit binary data (2 × (n) points of data in units of 16 bits) from the device specified by (s2), using the 16-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword in the device specified by (d)+1 and also stores the relative value of the first-matched device number from (s2) in the device specified by (d).



- If the value specified in (n) is 0, no processing is performed.
- If no matching data is found as the result of search, 0 is stored in the devices specified by (d) and (d)+1.

Point

- If the data to be searched by the DSERDATA(P) instruction has been sorted in ascending order, turning on SM702[*1] enables a binary search which can process the search faster. If SM702 is turned on even though the data to be search has not been sorted in ascending order, normal search results cannot be obtained. The following figure shows an example of binary search.



---

*1 SM702 is a special relay for setting the search method.
SM702 is off: Sequential search (linear search)
This method compares the search data with the data to be searched for starting from the start of data
SM702 is on: Binary search
For the data that has been sorted in ascending order, this method checks the center value of the search range, determining whether the center value is larger or smaller than the search value, and thereby narrows the search range to either side. Thus, target data is searched for by repeating this processing.

Operation error

There is no operation error.

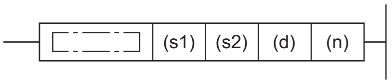
Searching 16-bit binary data (minimum, match, maximum)

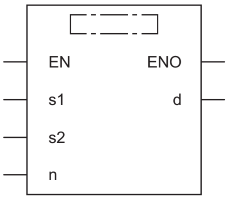
SERMM(P)

RnCPU **RnENCPU** ✗ **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

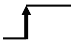
- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

These instructions search the (n) points in the 16-bit binary data specified by (s1) for the same data as the 16-bit binary data specified by (s2), the minimum value, and the maximum value.

Ladder	ST
	ENO:=SERMM(EN,s1,s2,n,d); ENO:=SERMMP(EN,s1,s2,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
SERMM	
SERMMP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device in which the same data, maximum value, and minimum value are searched	—	16-bit signed binary	ANY16
(s2)	Data to be searched for or device storing data	—	16-bit signed binary	ANY16
(d)	Start device storing number of the same data, maximum value, and minimum value detected by search	—	16-bit unsigned binary	ANY16_ARRAY (Number of elements: 5)
(n)	Number of data in which the same data, maximum value, and minimum value are searched	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions search the same data as the 16-bit binary data of (s2) in (n) data starting from (s1), and store the search result in (d) to (d)+4.
- When the same data exists, five devices starting from (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position.
- When the same data does not exist, five devices starting from (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position. In this case, however, 0s are stored in three devices starting from (d) (which store the number of the same data, first and last positions of the same data).
- When there are two or more maximum or minimum values in the searched data, the last position of the maximum/minimum values is stored.
- If the value specified in (n) is 0, no processing is performed.
- The following table shows example of configuration of search result table and data. (n=10)

Searched device (s1)	Searched data (s1) value (example)	Comparison data (s2) value (example)	Data position	Search result		
				Maximum value (d)+4	Match (d)	Minimum value (d)+3
(s1)	100	100	0	—	○ (First position)	—
(s1)+1	111		1	—	—	—
(s1)+2	100		2	—	○	—
(s1)+3	98		3	—	—	—
(s1)+4	123		4	—	—	—
(s1)+5	66		5	—	—	○
(s1)+6	100		6	—	○ (Last position)	—
(s1)+7	95		7	—	—	—
(s1)+8	210		8	○	—	—
(s1)+9	88		9	—	—	—

- The following table shows the search result table obtained by the above example.

Device number	Description	Search result item
(d)	3	Number of the same data
(d)+1	0	Position of the same data (first position)
(d)+2	6	Position of the same data (last position)
(d)+3	5	Minimum value position (last position)
(d)+4	8	Maximum value position (last position)

Operation error

There is no operation error.

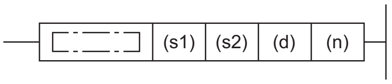
Searching 32-bit binary data (minimum, match, maximum)

DSERMM(P)

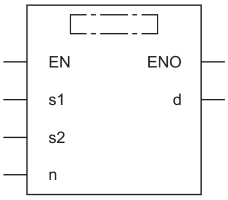
RnCPU RnENCPU ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

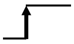
These instructions search the (n) points in the 32-bit binary data specified by (s1) for the same data as the 32-bit binary data specified by (s2), the minimum value, and the maximum value.

Ladder	ST
	ENO:=DSERMM(EN,s1,s2,n,d); ENO:=DSERMMP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSERMM	
DSERMMP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device in which the same data, maximum value, and minimum value are searched	—	32-bit signed binary	ANY32
(s2)	Data to be searched for or device storing data	—	32-bit signed binary	ANY32
(d)	Start device storing number of the same data, maximum value, and minimum value detected by search	—	32-bit unsigned binary	ANY32_ARRAY (Number of elements: 5)
(n)	Number of data in which the same data, maximum value, and minimum value are searched	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions search the same data as the 32-bit binary data of (s2)+1 and (s2) in (n) data starting from (s1)+1 and (s1), and store the search result in [(d)+1, (d)] to [(d)+9, (d)+8].
- When the same data exists, five devices starting from (d)+1 and (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position.
- When the same data does not exist, five devices starting from (d)+1 and (d) store the number of the same data, first and last positions of the same data, maximum value position and minimum value position. In this case, however, 0s are stored in three devices starting from (d)+1 and (d) (which store the number of the same data, first and last positions of the same data).
- When there are two or more maximum or minimum values in the searched data, the last position of the maximum/minimum values is stored.
- If the value specified in (n) is 0, no processing is performed.
- The following table shows example of configuration of search result table and data. (n=10)

Searched device (s1)	**Searched data (s1) value (example)**	**Comparison data (s2) value (example)**	**Data position**	**Search result**		
				Maximum value (d)+9, (d)+8	**Match (d)**	**Minimum value (d)+7, (d)+6**
(s1)+1, (s1)	100000	100000	0	—	○ (First position)	—
(s1)+3, (s1)+2	110100		1	—	—	—
(s1)+5, (s1)+4	100000		2	—	○	—
(s1)+7, (s1)+6	98000		3	—	—	—
(s1)+9, (s1)+8	123000		4	—	—	—
(s1)+11, (s1)+10	66000		5	—	—	○
(s1)+13, (s1)+12	100000		6	—	○ (Last position)	—
(s1)+15, (s1)+14	95000		7	—	—	—
(s1)+17, (s1)+16	910000		8	○	—	—
(s1)+19, (s1)+18	910000		9	○	—	—

- The following table shows the search result table obtained by the above example.

Device number	**Description**	**Search result item**
(d)+1, (d)	3	Number of the same data
(d)+3, (d)+2	0	Position of the same data (first position)
(d)+5, (d)+4	6	Position of the same data (last position)
(d)+7, (d)+6	5	Minimum value position (last position)
(d)+9, (d)+8	9	Maximum value position (last position)

Operation error

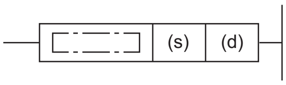
There is no operation error.

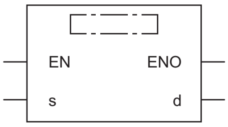
Checking 16-bit binary data

SUM(P)

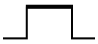
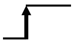
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions store the total number of "1" bits in the 16-bit binary data stored in the specified device.

Ladder	ST
	ENO:=SUM(EN,s,d); ENO:=SUMP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
SUM	
SUMP	

Setting data

■Descriptions, ranges, and data types

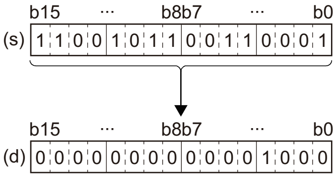
Operand	Description	Range	Data type	Data type (label)
(s)	Device containing data in which the total number of "1" bits is to be counted	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions store the total number of "1" bits in the 16-bit binary data, which is stored in the device specified by (s), in the device specified by (d).



The total number of 1s is stored in binary. (Eight 1s in the left example)

Operation error

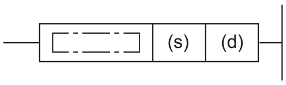
There is no operation error.

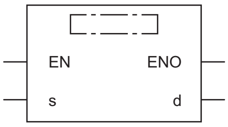
Checking 32-bit binary data

DSUM(P)

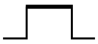
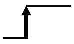
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These instructions store the total number of "1" bits in the 32-bit binary data stored in the specified device.

Ladder	ST
	ENO:=DSUM(EN,s,d); ENO:=DSUMP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
DSUM	
DSUMP	

Setting data

■Descriptions, ranges, and data types

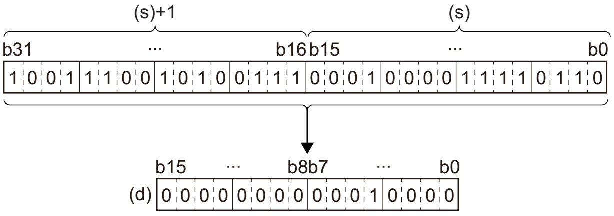
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing data in which the total number of "1" bits is to be counted	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions store the total number of "1" bits in the 32-bit binary data, which is stored in the device specified by (s), in the device specified by (d).



The total number of 1s is stored in binary. (Sixteen 1s in the left example)

Operation error

There is no operation error.

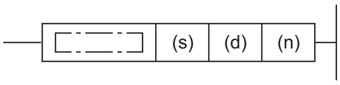
Checking the bit status in 16-bit binary data

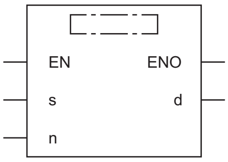
BON(P)

RnCPU **RnENCPU** ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

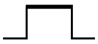
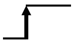
- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

These instructions check whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).

Ladder	ST
	ENO:=BON(EN,s,n,d); ENO:=BONP(EN,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
BON	
BONP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device storing the data	—	16-bit signed binary	ANY16
(d)	Bit device for storing the result	—	Bit	ANY_BOOL
(n)	Bit position to be checked	0 to 15	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

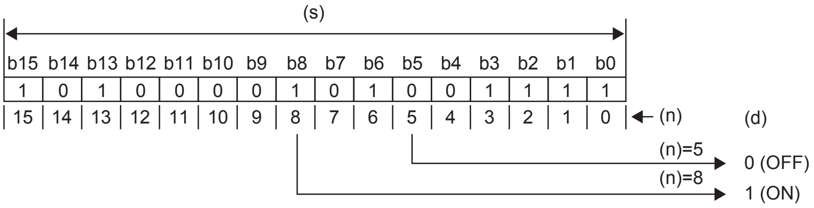
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○*2	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions check whether (n) bit(s) of the device specified by (s) are on or off, and stores the result in the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.

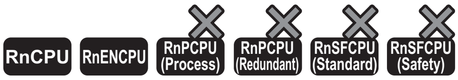


Operation error

Error code (SD0)	Description
3405H	The value specified for (n) is outside the following range. 0 to 15

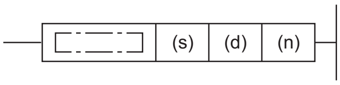
Checking the bit status in 32-bit binary data

DBON(P)

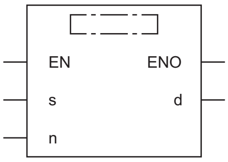


- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

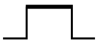
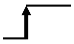
These instructions check whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).

Ladder	ST
	ENO:=DBON(EN,s,n,d); ENO:=DBONP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBON	
DBONP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device storing the data	—	32-bit signed binary	ANY32
(d)	Bit device for storing the result	—	Bit	ANY_BOOL
(n)	Bit position to be checked	0 to 31	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

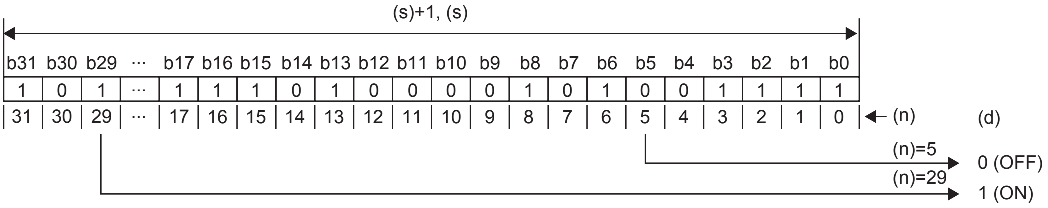
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○*2	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions check whether (n) bit(s) of the device specified by (s) are on or off, and stores the result in the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.



Operation error

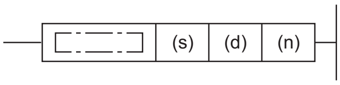
Error code (SD0)	Description
3405H	The value specified for (n) is outside the following range. 0 to 31

Searching the maximum value of 16-bit binary data

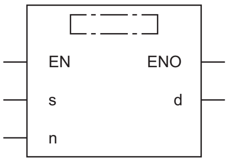
MAX(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions search the (n) points of 16-bit binary data in the specified device for the maximum value.

Ladder	ST[1]	
	ENO:=MAXP(EN,s,n,d);	ENO:=MAXP_U(EN,s,n,d);

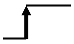
FBD/LD[1]



---

[1] The MAX and MAX_U instructions do not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
MAX MAX_U	
MAXP MAXP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	MAX(P)	Start device where the search target data is stored	—	16-bit signed binary	ANY16_S[1]
	MAX(P)_U			16-bit unsigned binary	ANY16_U[1]
(d)	MAX(P)	Start device for storing the search result of the maximum value	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 3)
	MAX(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 3)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

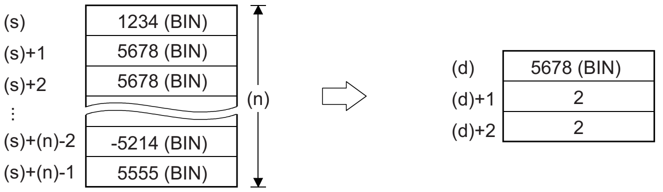
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 16-bit binary data in the device specified by (s) for the maximum value, and store the maximum value in the device specified by (d). Each instruction searches data starting from the device specified by (s) and detects first the maximum value in the xth point from (s), and stores x in (d)+1 and the number of maximum values in (d)+2.



(d): Maximum value
(d)+1: Location
(d)+2: Number of maximum values

Operation error

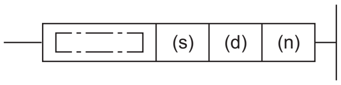
There is no operation error.

Searching the maximum value of 32-bit binary data

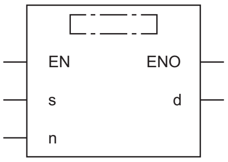
DMAX(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions search the (n) points of 32-bit binary data in the specified device for the maximum value.

Ladder	ST[1]	
	ENO:=DMAXP(EN,s,n,d);	ENO:=DMAXP_U(EN,s,n,d);

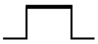
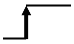
FBD/LD[1]



---

[1] The DMAX and DMAX_U instructions do not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
DMAX DMAX_U	
DMAXP DMAXP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DMAX(P)	Start device where the search target data is stored	—	32-bit signed binary	ANY32_S[1]
	DMAX(P)_U			32-bit unsigned binary	ANY32_U[1]
(d)	DMAX(P)	Start device for storing the search result of the maximum value	—	32-bit signed binary	ANY32_S_ARRAY (Number of elements: 4)
	DMAX(P)_U			32-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 4)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

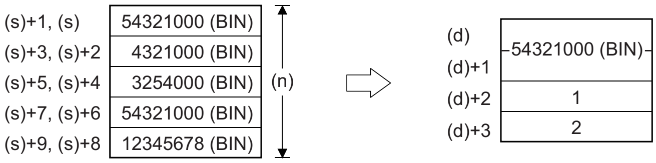
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 32-bit binary data in the device specified by (s) for the maximum value, and store the maximum value in the devices specified by (d) and (d)+1. Each instruction searches data starting from the device specified by (s) and detects first the maximum value in the xth point from (s), and stores x in (d)+2 and the number of maximum values in (d)+3.



(d), (d)+1: Maximum value

(d)+2: Location

(d)+3: Number of maximum values

Operation error

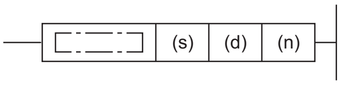
There is no operation error.

Searching the minimum value of 16-bit binary data

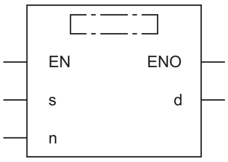
MIN(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions search the (n) points of 16-bit binary data in the specified device for the minimum value.

Ladder	ST[1]	
	ENO:=MINP(EN,s,n,d);	ENO:=MINP_U(EN,s,n,d);

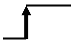
FBD/LD[1]



---

[1] The MIN and MIN_U instructions do not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
MIN MIN_U	
MINP MINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	MIN(P)	Start device where the search target data is stored	—	16-bit signed binary	ANY16_S[1]
	MIN(P)_U			16-bit unsigned binary	ANY16_U[1]
(d)	MIN(P)	Start device for storing the search result of the minimum value	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 3)
	MIN(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 3)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

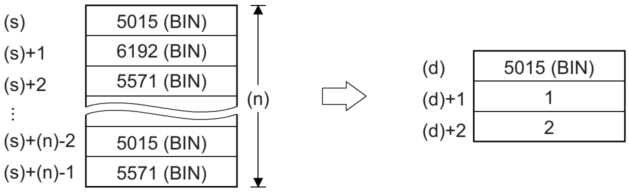
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 16-bit binary data in the device specified by (s) for the minimum value, and store the minimum value in the device specified by (d). Each instruction searches data starting from the device specified by (s) and detects first the minimum value in the xth point from (s), and stores x in (d)+1 and the number of minimum values in (d)+2.



(d) Minimum value

(d)+1: Location

(d)+2: Number of minimum values

Operation error

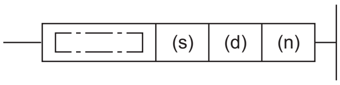
There is no operation error.

Searching the minimum value of 32-bit binary data

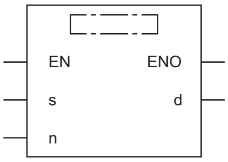
DMIN(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions search the (n) points of 32-bit binary data in the specified device for the minimum value.

Ladder	ST[1]	
	ENO:=DMINP(EN,s,n,d);	ENO:=DMINP_U(EN,s,n,d);

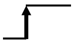
FBD/LD[1]



---

[1] The DMIN and DMIN_U instructions do not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1933 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
DMIN DMIN_U	
DMINP DMINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DMIN(P)	Start device where the search target data is stored	—	32-bit signed binary	ANY32_S[1]
	DMIN(P)_U			32-bit unsigned binary	ANY32_U[1]
(d)	DMIN(P)	Start device for storing the search result of the minimum value	—	32-bit signed binary	ANY32_S_ARRAY (Number of elements: 4)
	DMIN(P)_U			32-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 4)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

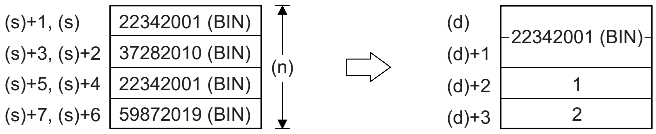
[1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 32-bit binary data in the device specified by (s) for the minimum value, and store the minimum value in the devices specified by (d) and (d)+1. Each instruction searches data starting from the device specified by (s) and detects first the minimum value in the xth point from (s), and stores x in (d)+2 and the number of minimum values in (d)+3.



(d), (d)+1: Minimum value

(d)+2: Location

(d)+3: Number of minimum values

Operation error

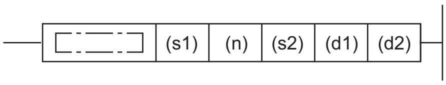
There is no operation error.

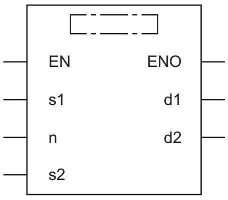
Sorting 16-bit binary data

SORTD(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions sort (n) points of 16-bit binary data in ascending or descending order.

Ladder	ST	
	ENO:=SORTD(EN,s1,n,s2,d1,d2);	ENO:=SORTD_U(EN,s1,n,s2,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
SORTD SORTD_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SORTD	Start device of the table data to be sorted	—	16-bit signed binary	ANY16_S[*1]
	SORTD_U			16-bit unsigned binary	ANY16_U[*1]
(n)		Number of sort data	0 to 65535	16-bit unsigned binary	ANY16
(s2)		Number of data to be compared once	0 to 65535	16-bit unsigned binary	ANY16
(d1)		Number of the bit device to be turned on upon completion of sort	—	Bit	ANY_BOOL
(d2)		Device used by the system	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

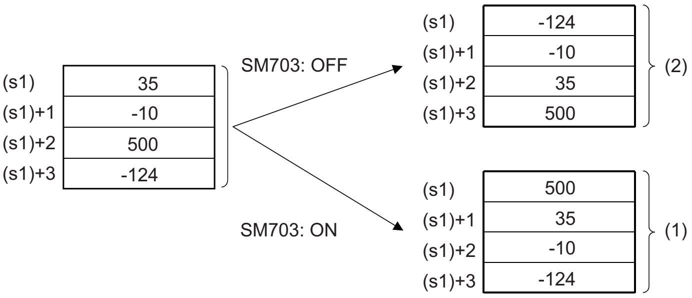
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	—	○[*1]	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

*1 T, C, and ST cannot be used.

Processing details

- These instructions sort (n) points of 16-bit binary data from (s1) in ascending or descending order. Data is sorted in ascending order when SM703 is off and in descending order when SM703 is on.



(1) Data are sorted in descending order.
(2) Data are sorted in ascending order.

- Sorting by the SORTD(_U) instruction requires several scans. The number of scans required till completion of sorting is determined by dividing the maximum number of executions performed before completion of sorting by the number of data compared once specified by (s2). (The decimal fractions are rounded up.) When the value in (s2) is increased, the number of scans before completion of sorting is decreased but the scan time is increased.
- The maximum number of executions before completion of sorting is calculated by (n)×(n-1)÷2 (times). When (n)=10, for example, 10×(10-1)÷2 = 45 times. At this time, setting (s2)=2, for example, makes 45÷ 2=22.5 meaning that 23 scans are required before completion of sorting.
- The completion device specified by (d1) turns off at start of execution of the SORT(_U) instruction and turns on upon completion of sorting. After completion of sorting, the device specified by (d1) is kept on. Turn it off as needed.
- The two points from the device specified by (d2) are used by the system at execution of the SORT(_U) instruction. Do not change the two points from the device specified by (d2). If they are changed, an error may occur. (Error code: 3405H)
- If the value in (n) is changed during sorting, the new number of sort data is used for sorting.
- If the execution command is turned off during sorting, sorting is interrupted. If the execution command is turned on again, sorting is performed from the beginning.
- If the next sorting is performed continuously after completion of the previous sorting, the execution command needs to be turned off and turned on again.

Operation error

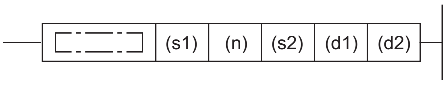
Error code (SD0)	Description
2821H	The device range of (n) points from the device specified by (s1) and the device range of two points from the device specified by (d2) are overlapping.
3405H	The value in (s2) is 0.
	In the second scan or after, the value in (d2) used by the system is equal to or greater than the value in (n).
	In the second scan or after, the value in (d2) used by the system is (d2)<(d2)+1.

Sorting 32-bit binary data

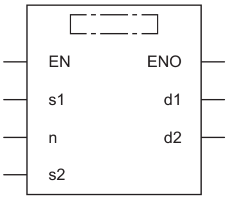
DSORTD(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions sort (n) points of 32-bit binary data in ascending or descending order.

Ladder	ST	
	ENO:=DSORTD(EN,s1,n,s2,d1,d2);	ENO:=DSORTD_U(EN,s1,n,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DSORTD DSORTD_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSORTD	Start device of the table data to be sorted	—	32-bit signed binary	ANY32_S[*1]
	DSORTD_U			32-bit unsigned binary	ANY32_U[*1]
(n)		Number of sort data	0 to 65535	16-bit unsigned binary	ANY16
(s2)		Number of data to be compared once	0 to 65535	16-bit unsigned binary	ANY16
(d1)		Number of the bit device to be turned on upon completion of sort	—	Bit	ANY_BOOL
(d2)		Device used by the system	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

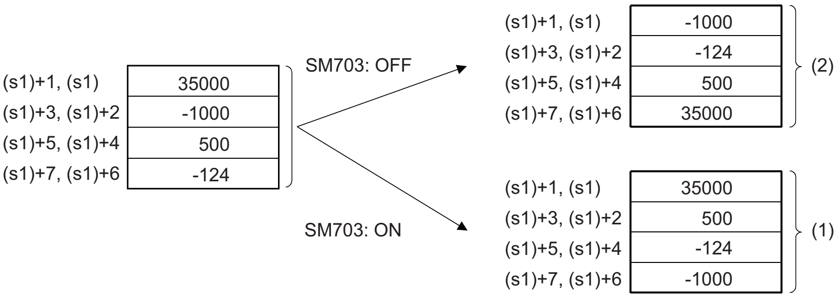
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	—	○[*1]	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

[*1] T, C, and ST cannot be used.

Processing details

- These instructions sort (n) points of 32-bit binary data from (s1) in ascending or descending order. Data is sorted in ascending order when SM703 is off and in descending order when SM703 is on.



(1) Data are sorted in descending order.
(2) Data are sorted in ascending order.

- Sorting by the DSORTD(_U) instruction requires several scans. The number of scans required till completion of sorting is determined by dividing the maximum number of executions performed before completion of sorting by the number of data compared once specified by (s2). (The decimal fractions are rounded up.) When the value in (s2) is increased, the number of scans before completion of sorting is decreased but the scan time is increased.
- The maximum number of executions before completion of sorting is calculated by (n)×(n-1)÷2 (times). When (n)=10, for example, 10×(10-1)÷2 = 45 times. At this time, setting (s2)=2, for example, makes 45÷ 2=22.5 meaning that 23 scans are required before completion of sorting.
- The completion device specified by (d1) turns off at start of execution of the DSORTD(_U) instruction and turns on upon completion of sorting. After completion of sorting, the device specified by (d1) is kept on. Turn it off as needed.
- The two points from the device specified by (d2) are used by the system at execution of the DSORTD(_U) instruction. Do not change the two points from the device specified by (d2). If they are changed, an error may occur. (Error code: 3405H)
- If the value in (n) is changed during sorting, the new number of sort data is used for sorting.
- If the execution command is turned off during sorting, sorting is interrupted. If the execution command is turned on again, sorting is performed from the beginning.
- If the next sorting is performed continuously after completion of the previous sorting, the execution command needs to be turned off and turned on again.

Operation error

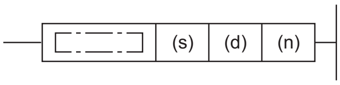
Error code (SD0)	Description
2821H	The device range of 2×(n) points from the device specified by (s1) and the device range of two points from the device specified by (d2) are overlapping.
3405H	The value in (s2) is 0.
	In the second scan or after, the value in (d2) used by the system is equal to or greater than the value in (n).
	In the second scan or after, the value in (d2) used by the system is (d2)<(d2)+1.

Adding 16-bit binary data

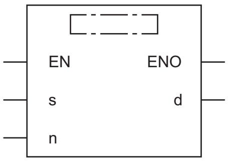
WSUM(P)(_U)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

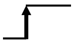
These instructions add the (n) points of 16-bit binary data from the specified device.

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

FBD/LD



■Execution condition

Instruction	Execution condition
WSUM WSUM_U	
WSUMP WSUMP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	WSUM(P)	Start device where the data for calculating the total value are stored	—	16-bit signed binary	ANY16_S[*1]
	WSUM(P)_U			16-bit unsigned binary	ANY16_U[*1]
(d)	WSUM(P)	Start device for storing the total value	—	32-bit signed binary	ANY32_S
	WSUM(P)_U			32-bit unsigned binary	ANY32_U
(n)		Number of data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

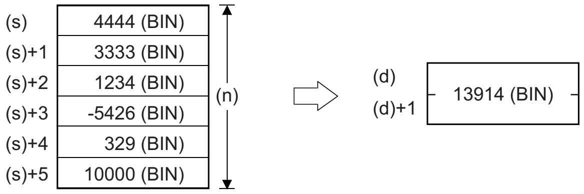
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(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).



Operation error

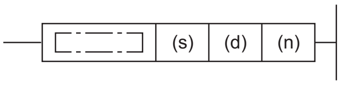
There is no operation error.

Adding 32-bit binary data

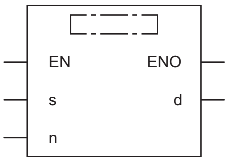
DWSUM(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

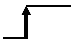
These instructions add the (n) points of 32-bit binary data in the devices starting from the specified one.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DWSUM DWSUM_U	
DWSUMP DWSUMP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	DWSUM(P)	Start device where the data for calculating the total value are stored	—	32-bit signed binary	ANY32_S[*1]
	DWSUM(P)_U			32-bit unsigned binary	ANY32_U[*1]
(d)	DWSUM(P)	Start device for storing the total value	—	64-bit signed binary	ANY32_ARRAY (Number of elements: 2)
	DWSUM(P)_U			64-bit unsigned binary	
(n)		Number of data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

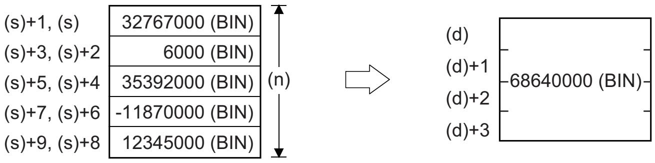
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(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

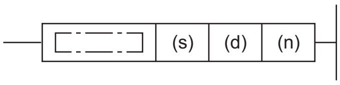
There is no operation error.

Calculating the mean value of 16-bit binary data

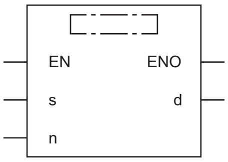
MEAN(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

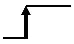
These instructions calculate the average value of the (n) points of 16-bit data in the devices starting from the specified one.

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

FBD/LD



■Execution condition

Instruction	Execution condition
MEAN MEAN_U	
MEANP MEANP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	MEAN(P)	Start device where the data for calculating the average value are stored	—	16-bit signed binary	ANY16_S[*1]
	MEAN(P)_U			16-bit unsigned binary	ANY16_U[*1]
(d)	MEAN(P)	Device for storing the mean value	—	16-bit signed binary	ANY16_S
	MEAN(P)_U			16-bit unsigned binary	ANY16_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

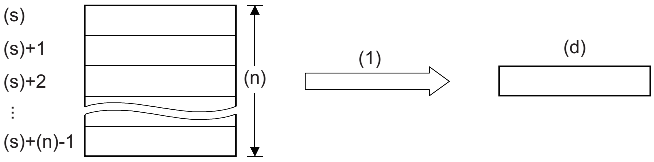
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(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

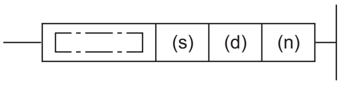
There is no operation error.

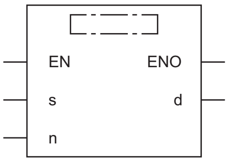
Calculating the mean value of 32-bit binary data

DMEAN(P)(_U)

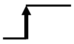
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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

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

FBD/LD


■Execution condition

Instruction	Execution condition
DMEAN DMEAN_U	
DMEANP DMEANP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	DMEAN(P)	Start device where the data for calculating the average value are stored	—	32-bit signed binary	ANY32_S[*1]
	DMEAN(P)_U			32-bit unsigned binary	ANY32_U[*1]
(d)	DMEAN(P)	Start device for storing the average value	—	32-bit signed binary	ANY32_S
	DMEAN(P)_U			32-bit unsigned binary	ANY32_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

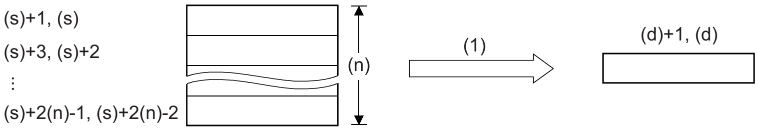
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	○	—	○	—	—	—	—
(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.

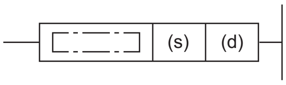
Calculating the square root of 16-bit binary data

SQRT(P)

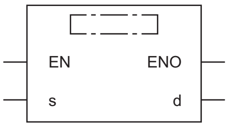
RnCPU **RnENCPU** ✕ **RnPCPU (Process)** ✕ **RnPCPU (Redundant)** ✕ **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

These instructions perform a square root operation of the specified 16-bit binary data.

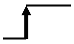
Ladder	ST*1
	ENO:=SQRTP(EN,s,d);

FBD/LD*1



*1 The SQRT instruction is not supported by the structured text language and the FBD/LD language. Use the standard function, SQRT. (☞ Page 1895 SQRT(_E))

■Execution condition

Instruction	Execution condition
SQRT	
SQRTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data whose square root is to be calculated is stored	0 to 65535	16-bit unsigned binary	ANY16
(d)	Device where the obtained square root is stored	—	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions perform a square root operation of the 16-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.

$\sqrt{(s)} \rightarrow (d)$

Operation error

There is no operation error.

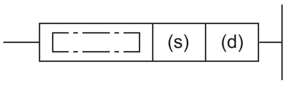
Calculating the square root of 32-bit binary data

DSQRT(P)

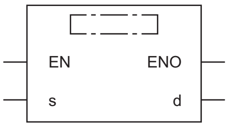
RnCPU **RnENCPU** ✕**RnPCPU (Process)** ✕**RnPCPU (Redundant)** ✕**RnSFCPU (Standard)** ✕**RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

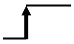
These instructions perform a square root operation of the specified 32-bit binary data.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DSQRT	
DSQRTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data whose square root is to be calculated is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	Device where the obtained square root is stored	—	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[*1]	○	○	○	○	○	○	○	○	—	—	—
(d)	○[*1]	○	○	○	○	○	○	○	—	—	—	—

[*1] FX and FY cannot be used.

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.

$\surd(s)+1,\ (s)\rightarrow(d)$

Operation error

There is no operation error.

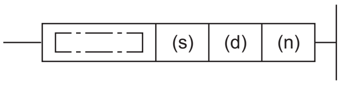
CRC operation

CRC(P)

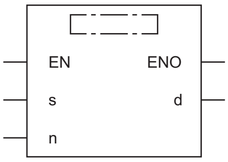
RnCPU **RnENCPU** ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

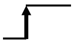
These instructions generate the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d).

Ladder	ST
	ENO:=CRC(EN,s,n,d); ENO:=CRCP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CRC	
CRCP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the target data of CRC value generation is stored	—	16-bit signed binary	ANY16
(d)	Device where the generated CRC value is stored	—	16-bit signed binary	ANY16
(n)	Number of 8-bit data (unit: byte) for which the CRC value is generated or the device storing the number of 8-bit data (unit: byte)	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, 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)	○*1	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions generate the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d). "X^{16} + X^{15} + X^2 +1" is used in the generator polynomial of the CRC value (CRC-16). The 16-bit conversion mode and 8-bit conversion mode are available for these instructions. The conversion mode can be selected by turning on or off SM772.
- If the value specified in (n) is 0, no processing is performed.
- These instructions calculate the CRC (cyclic redundancy check) value which is an error check method used in communication. In addition to CRC, there are other methods to check an error, such as parity check and sum check (checksum). For obtaining the horizontal parity value and sum check value, the CCD(P) instruction is available. (☞ Page 903 CCD(P))
- The operation in each conversion mode is described below.

■16-bit conversion mode (while SM772 is OFF)

In this mode, the CRC operation is executed for upper 8 bits (in units of byte) and lower 8 bits (in units of byte) of the device specified by (s). The operation result is stored to 16 bits in one device specified in (d).

Ex.

When (n)=6

In 16-bit conversion mode, the six bytes in the following shaded portion become an operation target. The CRC value is determined as "A57BH", and therefore "A57BH" is stored in the device specified by (d).

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

■8-bit conversion mode (while SM772 is ON)

CRC operation is executed only for lower 8 bits (lower byte) of the device specified by (s). With regard to the operation result, lower 8 bits (in units of byte) are stored to the device specified by (d), and upper 8 bits (in units of byte) are stored to a device specified by (d)+1.

Ex.

When (n)=6

In 8-bit conversion mode, the six bytes in the following shaded portion become an operation target. The CRC value is "BDA1H" and therefore "A1H" is stored in the device specified by (d) and "BDH" is stored in the device specified by (d)+1.

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

Operation error

Error code (SD0)	Description
3405H	A digit other than 4 is specified in the digit-specified bit device in (s) and (d).

7.21 Database Access Instructions

The database access instructions add, update, obtain, or delete data with respect to the tabular data such as product and production information managed as databases in the programmable controller.

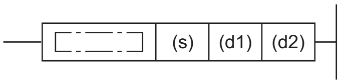
These instructions construct a database from the Unicode text file that defines information such as a table configuration, and operates the database thus constructed. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Importing data to the data base

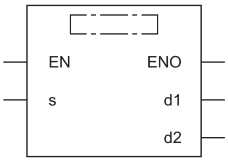
DBIMPORT(P)

RnCPU RnENCPU ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

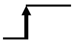
These instructions import the data stored in the Unicode text file at the path specified by (s) and construct a database.

Ladder	ST
	ENO:=DBIMPORT(EN,s,d1,d2); ENO:=DBIMPORTP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBIMPORT	
DBIMPORTP	

Setting data

■Descriptions, ranges, and data types

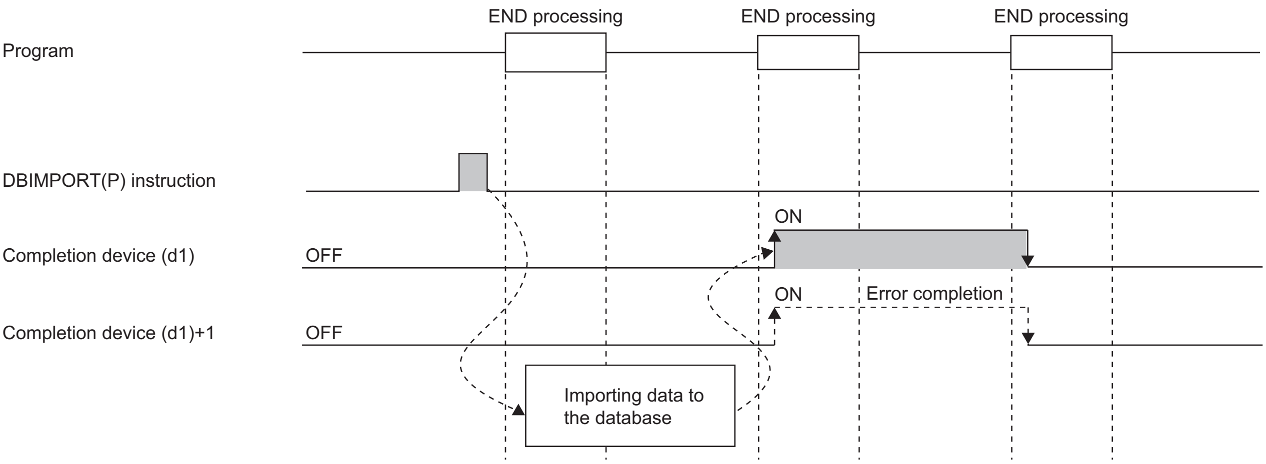
Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the Unicode text file name Example: "2:\database1\recipe\recipe_db.txt" Within 255 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions import the data stored in the Unicode text file specified by (s) and construct a database. Information such as a table configuration needs to be defined in advance in the Unicode text file used by the DBIMPORT(P) instruction. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- When the database that is already open exists and the DBIMPORT(P) instruction is executed, it is completed with an error.
- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBIMPORT(P) instruction.



- SM753 (File being accessed) turns on while the DBIMPORT(P) instruction is being executed.[*1] While SM753 is on, the DBIMPORT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBIMPORT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[*1] If executed, no processing is performed.
- When DBIMPORT(P) instruction is executed, if a database with the name specified in the Unicode text file specified by (s) exists on the same path as the Unicode text file, the database with the same name on the SD memory card (in the database folder) is deleted and a new database is created with the name specified in the Unicode text file.

[*1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBIMPORT(P) instruction is executed during execution of the database access instruction.
- The Unicode text file specified by (s) does not exist.
- The number of fields specified in the field name row of the Unicode text file does not match the number of fields in the record row.
- A table definition start tag or end tag is missing in the Unicode text file.
- An out-of-range value is set for the key constraint in the Unicode text file.
- The database name, table name, or field name in the Unicode text file exceeds 32 characters.
- The number of tables or fields in the Unicode text file exceeds the maximum number.
- An out-of-spec data type is specified in the Unicode text file.
- The number of records in the Unicode text file exceeds the maximum number (for a programmable controller CPU with firmware version earlier than "28").
- An access to the database has failed.
- The database name contains an invalid character.
- The total number of characters used in the database name specified in the Unicode text file and those used in the folder path (including the drive path character) specified by (s) exceeds 128.
- The database that is already open exists and the DBIMPORT(P) instruction is executed.
- The number of characters on the comment line exceeds the maximum number.
- A character other than those that can be represented as ASCII codes (0020H to 007EH) is used in the folder name of the Unicode text file specified by (s).
- The database name in the Unicode text file exists on the SD memory card and the database is used by another function when the DBIMPORT(P) instruction is executed.

If an error is detected because of the Unicode text format, the DBIMPORT(P) instruction turns on the error termination signal in (d1)+1 and stores the Unicode text line where an error was detected in SD760 and SD761.

Operation error

Error code (SD0)	Description
2840H	A numerical value other than 2 is specified for the drive number in (s).
3405H	The character string (path) in the device specified by (s) exceeds 255 characters.

For the error code stored in the completion status of the operand, refer to the following.

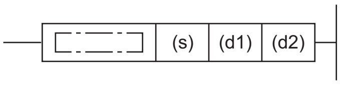
☞ Page 986 Error codes related to database access instructions

Exporting data from the data base

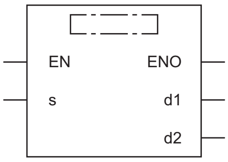
DBEXPORT(P)

RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

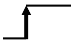
These instructions export the data stored in the specified database to the Unicode text file.

Ladder	ST
	ENO:=DBEXPORT(EN,s,d1,d2); ENO:=DBEXPORTP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBEXPORT	
DBEXPORTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the database folder path Example: "2:\database1\recipe" Within 128 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

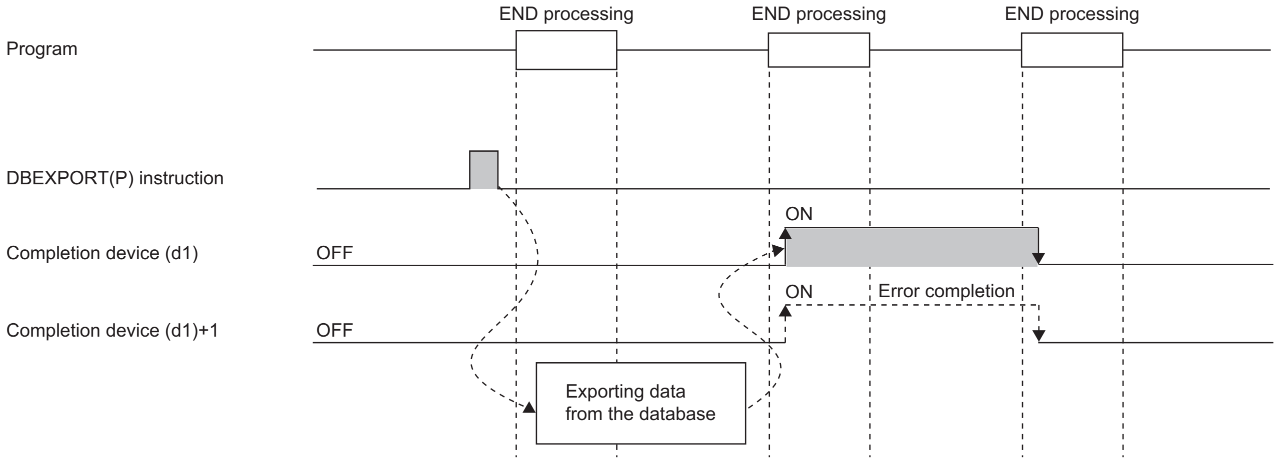
Processing details

- These instructions export the data in the database stored in the database folder at the path specified by (s) to the Unicode text file.
- The Unicode text file is created in the folder where the database folder is stored. The file name is "database_name.txt". If the same Unicode text file already exists, the file is overwritten with the exported data.

Ex.

When the path of the database folder is "2:\database\recipe1", executing the instruction creates Unicode text file "2:\database\recipe1.txt" and exports data to the file.

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBEXPORT(P) instruction.



- The internal configuration of the Unicode text file to which data is exported is the same as the file to which data is imported by the DBIMPORT(P) instruction. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- SM753 (File being accessed) turns on while the DBEXPORT(P) instruction is being executed.[1] While SM753 is on, the DBEXPORT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBEXPORT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[1] If executed, no processing is performed.
- During transaction, the DBEXPORT(P) instruction cannot be executed.

[1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBEXPORT(P) instruction is executed during execution of the database access instruction.
- The path specified by (s) is not a database.
- Writing data to the Unicode text failed due to the failure to access the database.
- The number of characters of the path (including the drive path character) specified by (s) exceeds 128.
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified in (s) for a programmable controller CPU with firmware version earlier than "28".
- The DBEXPORT(P) instruction is executed during transaction.

Operation error

Error code (SD0)	Description
2840H	A numerical value other than 2 is specified for the drive number in (s).

For the error code stored in the completion status of the operand, refer to the following.

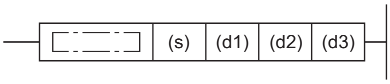
☞ Page 986 Error codes related to database access instructions

Opening the data base

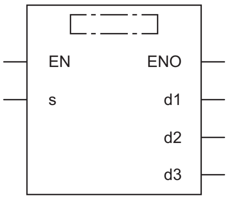
DBOPEN(P)

RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

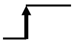
These instructions connect to the database specified by (s) and make it available.

Ladder	ST
	ENO:=DBOPEN(EN,s,d1,d2,d3); ENO:=DBOPENP(EN,s,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
DBOPEN	
DBOPENP	

Setting data

■Descriptions, ranges, and data types

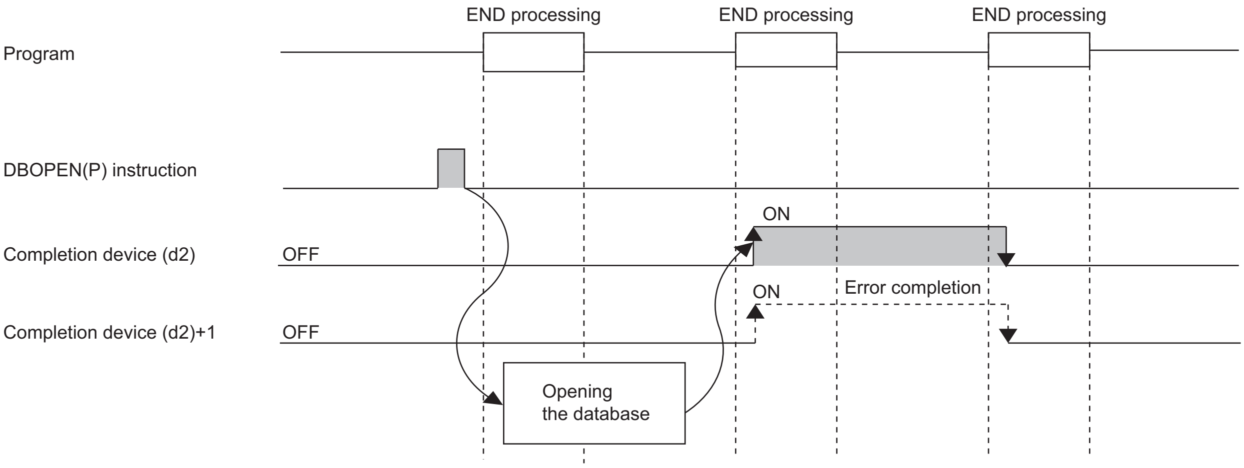
Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the folder path of the database to be opened. Example: "2:\database1\recipe" Within 128 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d2)	Completion device (start device that turns on one scan upon completion of instruction) • (d2)+0: Completion signal • (d2)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d3)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—
(d3)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions connect to the database stored in the folder path specified by (s) and makes it available.
- Specify "(drive number):(database folder path)" for the storage location. The drive number is fixed to 2 (SD memory card).
- Upon normal completion, the instruction stores the identification number of the connected database in the device (d1). The completion signal in the completion device (d2)+0 is turned on and 0 is stored as the completion status in the device (d3).
- The following figure shows the operation of the completion device at completion of the DBOPEN(P) instruction.



- The DBOPEN(P) instruction enables connections to a maximum of four different databases at the same time.
- During transaction, the database cannot be newly opened.
- SM753 (File being accessed) turns on while the DBOPEN(P) instruction is executed.*1 While SM753 is on, the DBOPEN(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBOPEN(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error termination signal in (d2)+1 is turned on and an error code is stored in (d3).

- The DBOPEN(P) instruction is executed during execution of the database access instruction.
- The storage location specified by (s) does not exist.
- An attempt is made to connect to the database that has already been connected.
- The DBOPEN(P) instruction is executed for a database exceeding the maximum number of databases that can be connected concurrently.
- The number of characters of the path (including the drive path character) specified by (s) exceeds 128.
- The database is opened during transaction.

Operation error

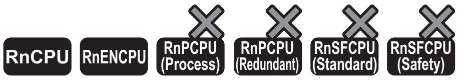
Error code (SD0)	Description
2820H	The area specified by (d2) exceeds the applicable range of the device/label used.
2840H	A numerical value other than 2 is specified for the drive number in (s).

For the error code stored in the completion status of the operand, refer to the following.

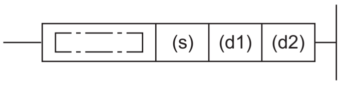
☞ Page 986 Error codes related to database access instructions

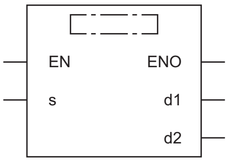
Closing the data base

DBCLOSE(P)

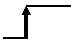


These instructions clear the connection from the specified database.

Ladder	ST
	ENO:=DBCLOSE(EN,s,d1,d2); ENO:=DBCLOSEP(EN,s,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
DBCLOSE	
DBCLOSEP	

Setting data

■Description, range, data type

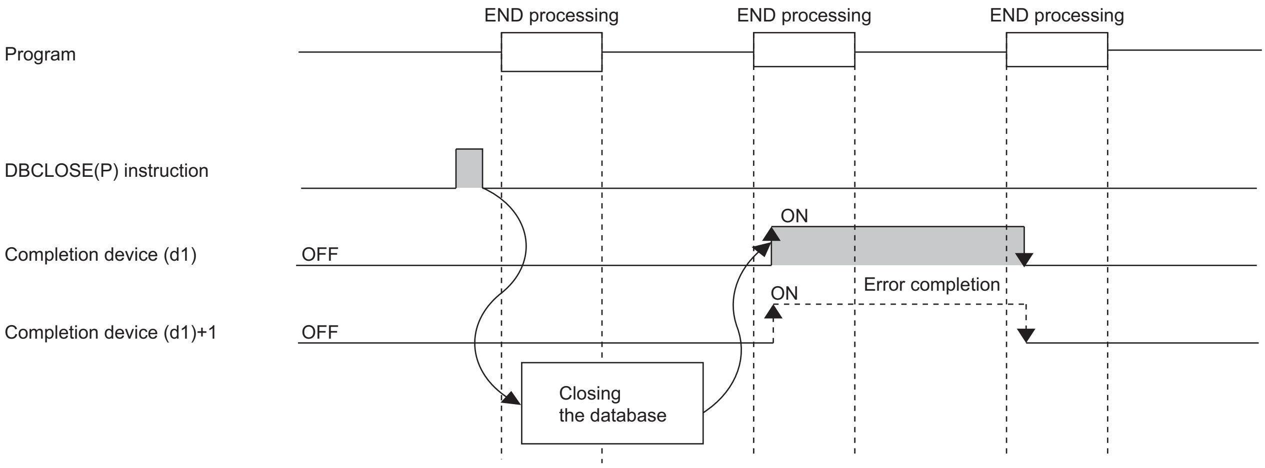
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions release the connection between the database identification number specified by (s) and the corresponding database.
- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBCLOSE(P) instruction.



- If the DBCLOSE(P) instruction is executed before DBCOMMIT(P) or DBROLBAK(P) while the transaction is run by the DBTRANS(P) instruction, the transaction is determined in the status at the execution of the DBCLOSE(P) instruction.
- SM753 (File being accessed) turns on while the DBCLOSE(P) instruction is executed.[1] While SM753 is on, the DBCLOSE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBCLOSE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[1] If executed, no processing is performed.

[1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBCLOSE(P) instruction is executed during execution of the database access instruction.
- The identification number specified by (s) is an already disconnected database.
- An identification number outside the setting range is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

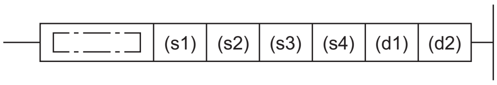
☞ Page 986 Error codes related to database access instructions

Adding a record to the data base

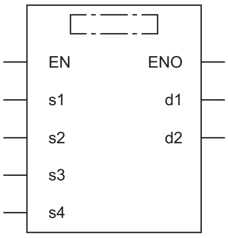
DBINSERT(P)

RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

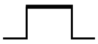
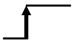
These instructions add a record to the table of the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBINSERT(EN,s1,s2,s3,s4,d1,d2); ENO:=DBINSERTP(EN,s1,s2,s3,s4,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBINSERT	
DBINSERTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the database table names.[*1]	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the database field names.[*1]	—	Word	ANY16[*2]
(s4)	Start device for storing insertion data	—	Word	ANY16[*2]
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] The table and field names are case-sensitive.

[*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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

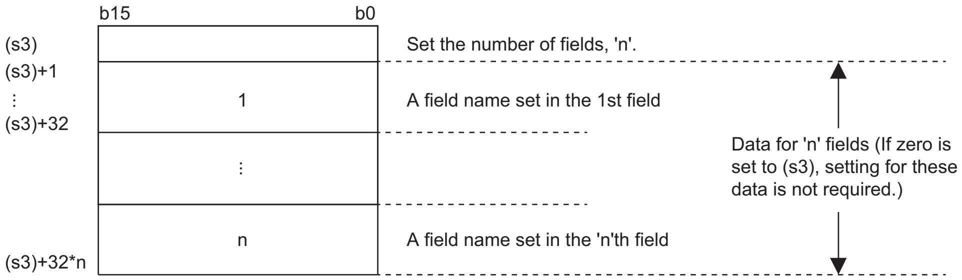
■Database field name

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of fields	Specify the number of fields to which a value is to be added. Specify a value equal to or less than the number of fields of the table specified in (s2). If 0 is specified, all fields of the table are subject to output.[1]	0 to 128[2]	User
+1 to +□	Field name	Specify the name of each field. Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. The last address □ in (s3) varies according to the number of fields. □=32×n (n: Number of fields) This setting of this item is not necessary when "0" is specified in (s3)+0. [1](Ignored if specified.)	—	User

[1] Programmable controller CPU with firmware version "28" or later supports this processing.

[2] "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s3).

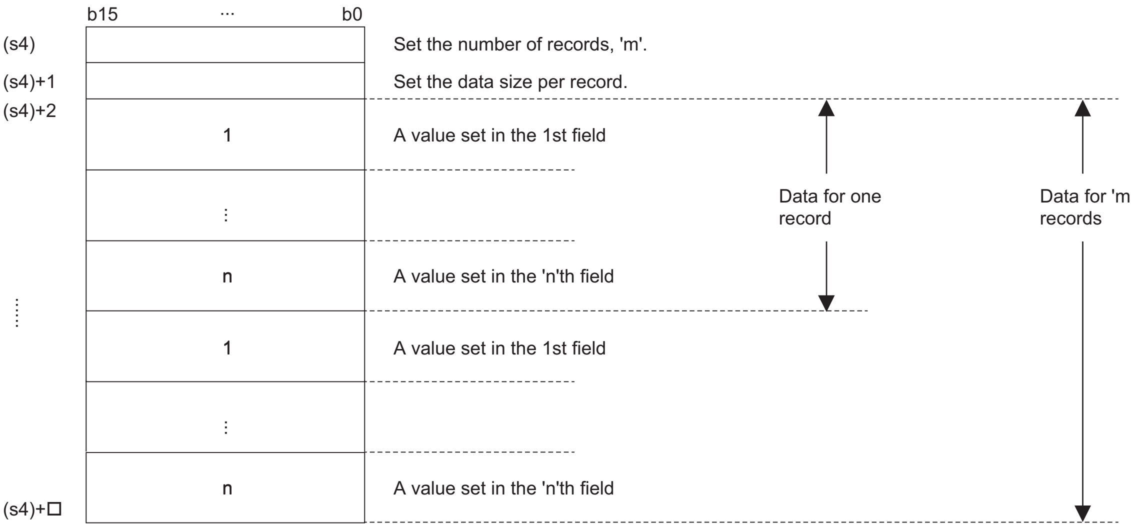


■Insertion data

Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of records	Specify the number of records to be added. The number of records to be added is a value which satisfies the following formula:[*1] Number of records (m) ≤ (32768/Data size per record)	1 to m[*1]	User
+1	Size	Specify the size of one record.	Depends on the data type	User
+2 to +□	Value to be added	Specify the data for the number of fields specified by (s3) for the number of records (m) specified by (s4)+0.	Depends on the data type	User

[*1] "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s4). Set the value corresponding to each field set in (s3).



- The data size of each value follows the size of the data type of each field.

Data type of field	Data size (unit: word)
BOOL	1
WORD	1
DWORD	2
INT	1
DINT	2
REAL	2
LREAL	4
STRING	• Even number of characters: (Number of characters÷2)+1 • Odd number of characters: Rounding up the number of characters÷2 [Example] STRING: 32: (32÷2)+1=17 STRING: 15: (15÷2)=7.5 rounding up→8
WSTRING	Number of characters+1 [Example] WSTRING: 32: (32)+1=33

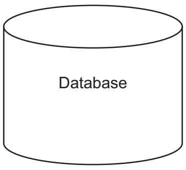
Ex.

When registering a record with ID = 0003H to the product information table (proInfo) in the database using the DBINSERT(P) instruction

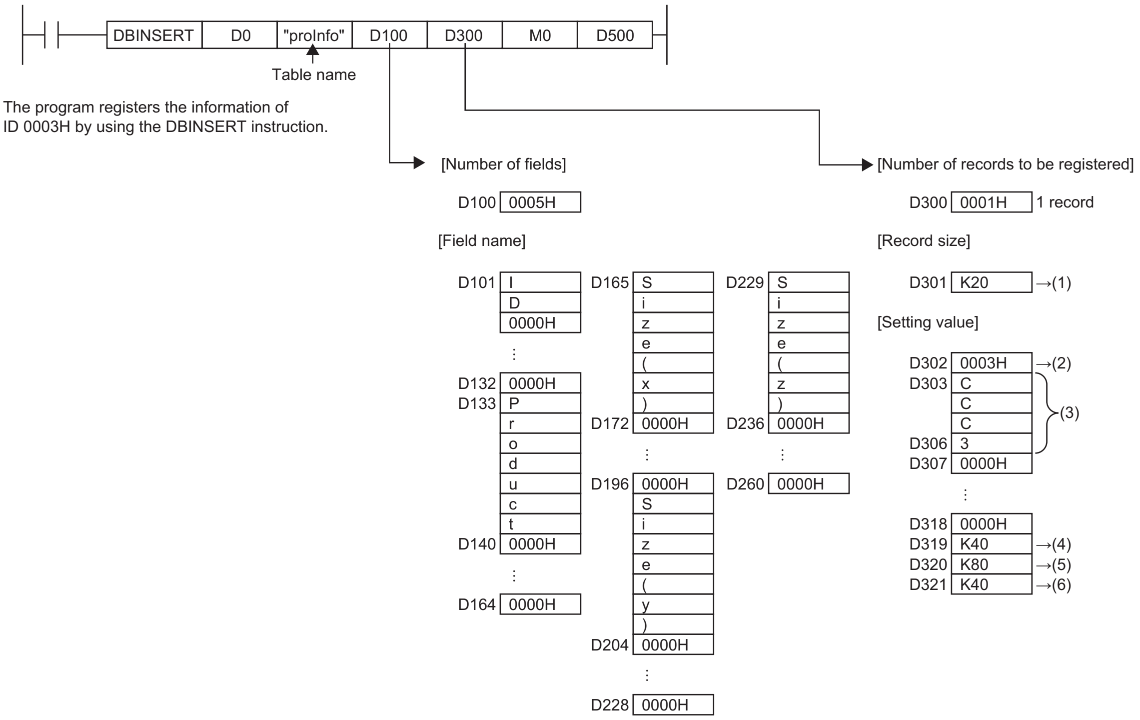
[Product information table (proInfo)]

ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
0003H	"CCC3"	40	80	40
⋮				

ID	WORD type
Product	WSTRING type (15 characters)
Size(x)	WORD type
Size(y)	WORD type
Size(z)	WORD type



[Program]



The program registers the information of ID 0003H by using the DBINSERT instruction.

(1) WSTRING type (16 words) + WORD type (1 word) × 4 = 20 words
(2) Setting value in field "ID"
(3) Setting value in field "Product"
(4) Setting value in field "Size(x)"
(5) Setting value in field "Size(y)"
(6) Setting value in field "Size(z)"

Processing details

- These instructions add a record to the table specified by (s2) in the database corresponding to the identification number specified by (s1).
- Specify the number of fields of the record to be added and field names in (s3). For the field names to be added, not all fields making up the table need to be specified. NULL is stored in a field which is not specified.
- Specify the number of records to be added and the size and value per record in (s4).
- The following figure shows an example when a record is added to the table recipeA by using the DBINSERT(P) instruction.

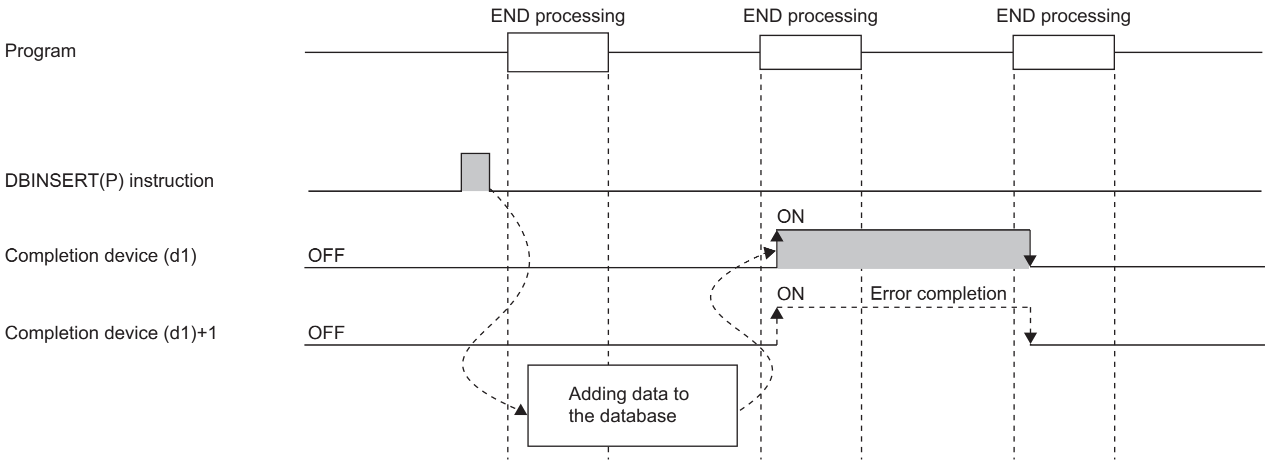
(1)	(2)	(3)
1	ProductA	100
2	ProductB	200

(1) Field 1: WORD

(2) Field 2: WSTRING (16 characters maximum)

(3) Field 3: INT

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- Upon completion with an error, the error completion signal in the completion device (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBINSERT(P) instruction.



- SM753 (File being accessed) turns on while the DBINSERT(P) instruction is executed.*2 While SM753 is on, the DBINSERT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBINSERT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*2 If executed, no processing is performed.

*2 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBINSERT(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the number of fields to be added.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- An out-of-range value is specified in (s4) for the number of records to be added.
- Database insertion processing failed.
- The range of the data for one record set in (s4)+2 does not match the size specified by (s4)+1.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

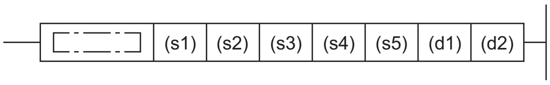
☞ Page 986 Error codes related to database access instructions

Updating the record in the data base

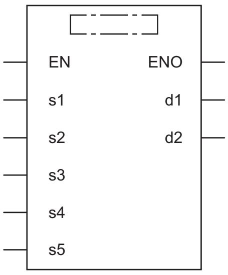
DBUPDATE(P)

RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

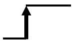
These instructions update all records that meet the specified condition in the specified table corresponding to the specified identification number.

Ladder	ST
	ENO:=DBUPDATE(EN,s1,s2,s3,s4,s5,d1,d2); ENO:=DBUPDATEP(EN,s1,s2,s3,s4,s5,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBUPDATE	
DBUPDATEP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database table to be updated.[*1]	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the field name of the database to be updated.[*1]	—	Word	ANY16[*2]
(s4)	Start device for storing the updated data	—	Word	ANY16[*2]
(s5)	Start device for storing the update conditions (a maximum of two conditions).	—	Word	ANY16[*2]
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[*1] The table and field names are case-sensitive.

[*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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(s5)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Update target field name

Operand: (s3)

Device	Item	Description	Setting range	Set by
+0	Number of fields	Specify the number of fields to be updated.	1 to 128[*1]	User
+1 to +□	Field name	Specify the name of each field. Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. The last address □ in (s3) varies according to the number of fields. □=32×n (n: number of fields)	—	User

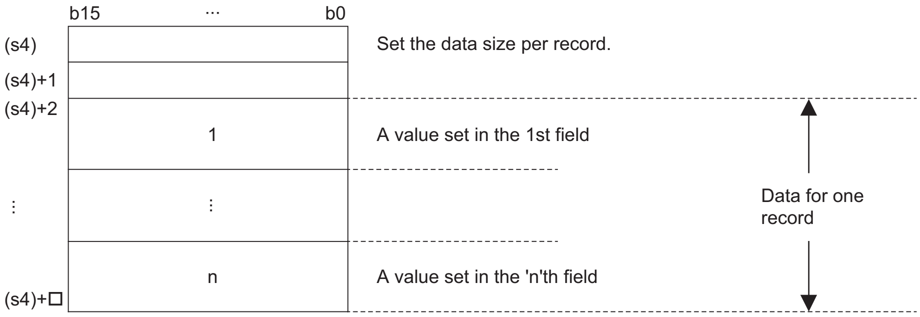
*1 "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

■Updated data

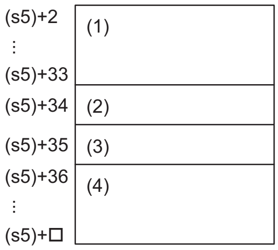
Operand: (s4)

Device	Item	Description	Setting range	Set by
+0	Data size	Set the data size of the field to be updated.	—	User
+1	Not used	—	—	—
+2 to +□	Set value	Set the updated data.	Depends on the data type	User

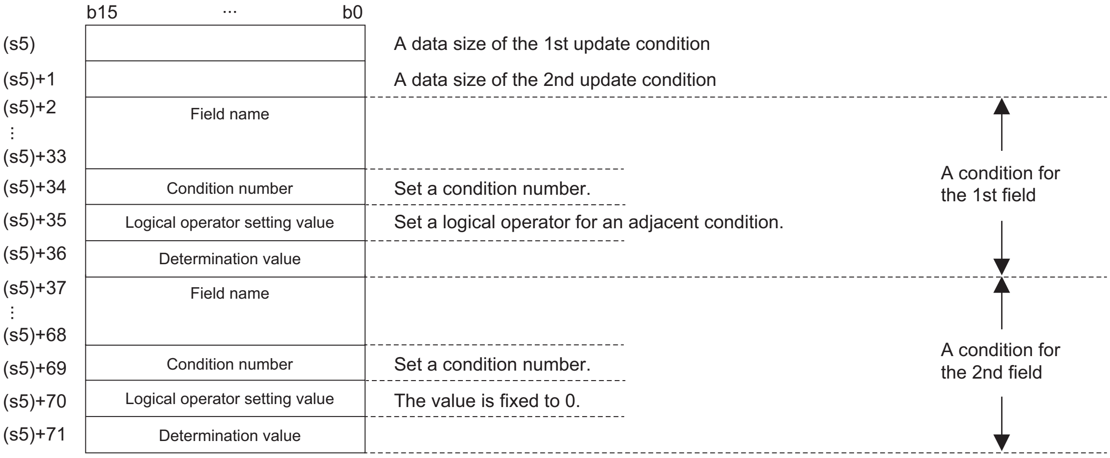
The following figure shows the setting format of (s4). Set the value corresponding to each field in (s3). The data size of each set value follows the size of the data type of each field. (☞ Page 956 DBINSERT(P))



■Update condition

Operand: (s5)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Data size of the first condition	Set the data size of the first update condition.	1 to 125	User
+1	Data size of the second condition	Set the data size of the second update condition. • 0: No condition • Other than 0: Data size	0 to 125	User
+2 to +□	Update condition	Set the update conditions. The last address in (s5) varies depending on the data type of the determination value.  ■Field name (1) Specify the field name with a Unicode character string in 32 characters (fixed). For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating an update condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one update condition is specified, set the logical operator setting value of the first update condition to 0. In this case, the second update condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User

The following figure shows the setting format of (s5). When the data type is WORD, set as many field names, condition numbers, logical operator setting value, and determination values as there are update conditions. The data size of the determination value follows the size of the data type of each field. (☞ Page 956 DBINSERT(P))

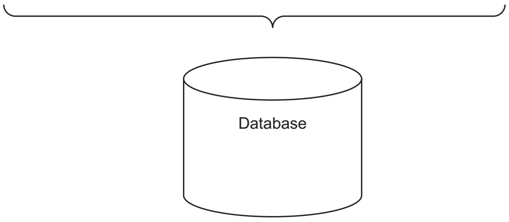


Ex.

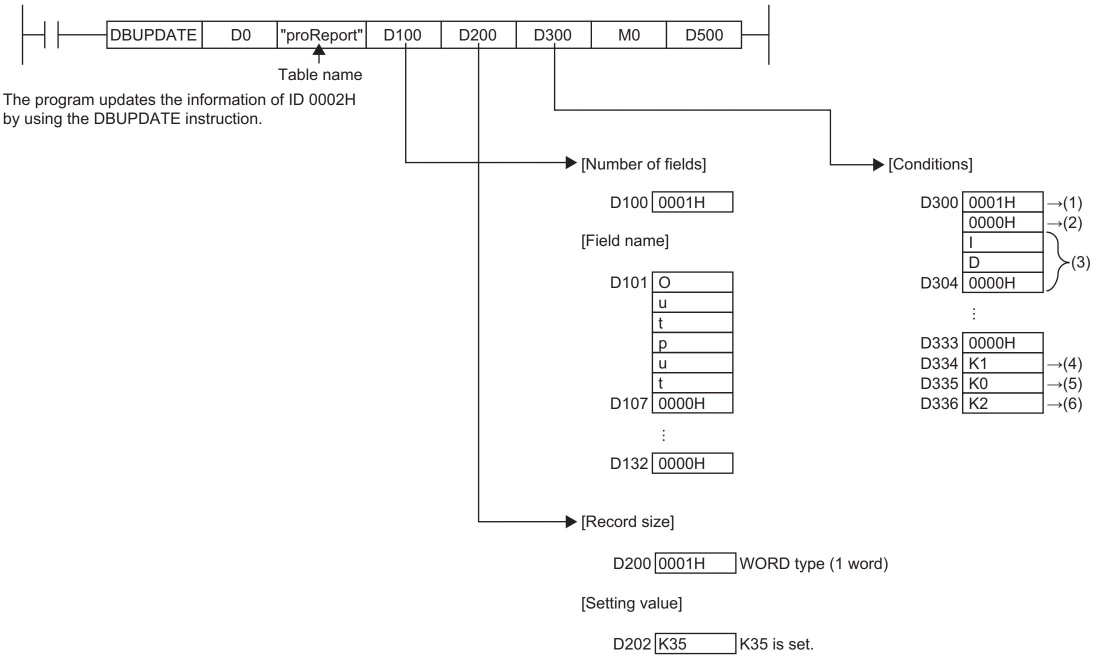
When updating Output of a record with ID = 0002H to 35 using the DBUPDATE(P) instruction

[Production result table (proReport)]

ID	Product	Output
0001H	"AAA1"	32
0002H	"BBB2"	35
0003H	"CCC3"	38
⋮		



[Program]



The program updates the information of ID 0002H by using the DBUPDATE instruction.

(1) Second condition data size→WORD type (1 word)
(2) 0 for the second condition which is not used
(3) Field name of first update condition
(4) The condition number is "=". Set K1.
(5) No logical operator setting value is used. Set K0.
(6) Set K2 for the determination value.

Processing details

- Updates all record that meets the condition specified by (s5) in the table specified by (s2) in the database specified by the identification number specified by (s1).
- Specify the field name of the record to be updated in (s3). Not all fields in the table need to be specified but at least one field needs to be specified.
- Specify the value of the record to be updated in (s4). The set value in any field not specified in (s3) is not updated.
- Specify the condition to be updated in (s5). At least one condition needs to be specified and a maximum of two conditions can be specified.
- When the DBUPDATE(P) instruction updates the table recipeA record that matches "field1="2" of the update condition in which the update target field name is 2 and the updated data is New-Product1, the following occurs.

Table recipeA (before update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300

Table recipeA (after update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	New-Product1	200
3	ProductC	300

- When the DBUPDATE(P) instruction updates the table recipeA record that matches "field1>="2" and field1<=3 of the update condition in which the update target field name is 2 and the updated data is New-Product1, the following occurs.

Table recipeA (before update)

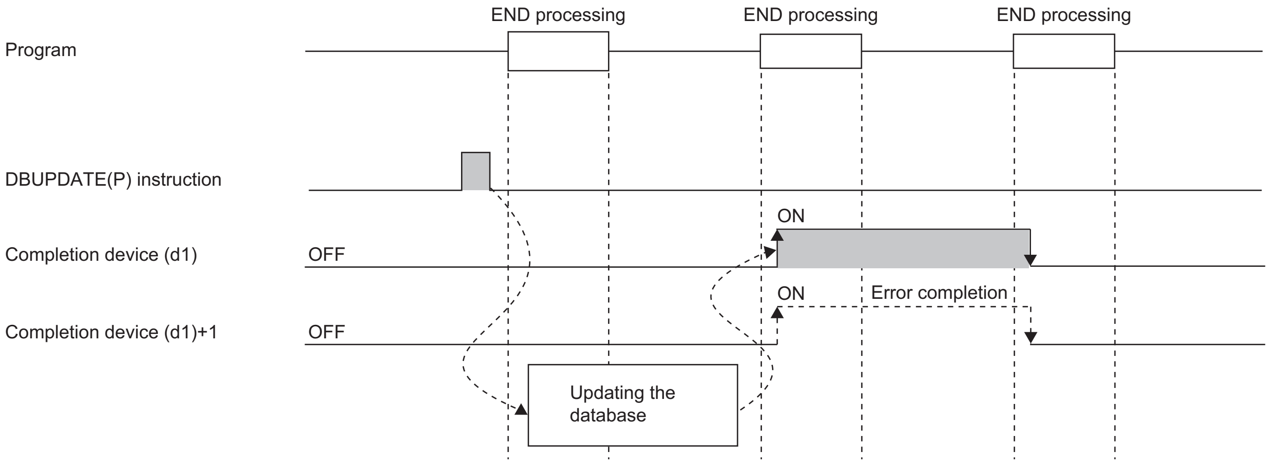
Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300
4	ProductD	200

Table recipeA (after update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	New-Product1	200
3	New-Product1	300
4	ProductD	400

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).

- The following figure shows the operation of the completion device at completion of the DBUPDATE(P) instruction.



- SM753 (File being accessed) turns on while the DBUPDATE(P) instruction is executed.[*1] While SM753 is on, the DBUPDATE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBUPDATE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[*1] If executed, no processing is performed.

[*1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBUPDATE(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the number of fields to be updated.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- An out-of-range value is specified in (s5) for the condition size.
- The size of the first condition in (s5) is set to 0.
- An out-of-range value is specified in (s5) for the condition symbol.
- An out-of-range value is specified in (s5) for the logical operator setting value.
- Database update processing failed.
- The logical operator setting value specified by (s5)+35 is 1 or 2, and that in (s5)+1 is set to 0.
- The range of the data for one record set in (s4)+2 does not match the size specified by (s4).
- The field name to be set in (s5)+2 is left unset.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), (s5), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

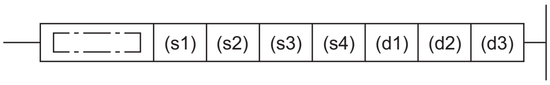
☞ Page 986 Error codes related to database access instructions

Searching the record in the data base

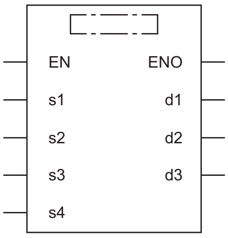
DBSELECT(P)

RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

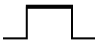
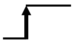
These instructions search the records in the table in the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBSELECT(EN,s1,s2,s3,s4,d1,d2,d3); ENO:=DBSELECTP(EN,s1,s2,s3,s4,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
DBSELECT	
DBSELECTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database to be searched.[*1]	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the field name of the database to be searched.[*1]	—	Word	ANY16[*2]
(s4)	Start device for storing the search conditions (a maximum of two conditions).	—	Word	ANY16[*2]
(d1)	Search result	—	Word	ANY16[*2]
(d2)	Completion device (start device that turns on one scan upon completion of instruction) • (d2)+0: Completion signal • (d2)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d3)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 The table and field names are case-sensitive.

*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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—
(d3)	—	—	○	—	—	—	—	○	—	—	—	—

■Search target table name

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0 to +□	Table name	Specify the search target table name with up to 32 characters.	—	User

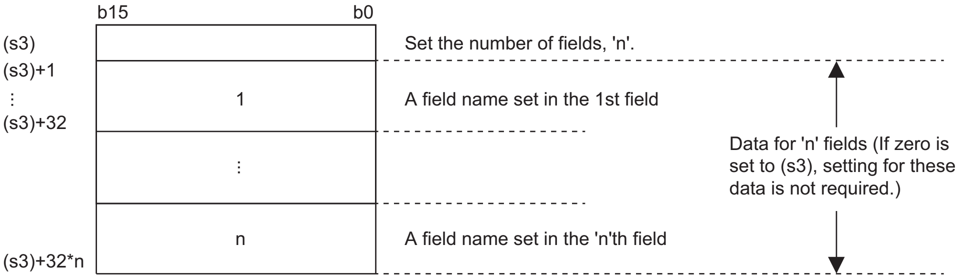
■Search target field name

Operand: (s3)

Device	Item	Description	Setting range	Set by
+0	Number of fields	Specify the number of fields to be searched. If 0 is specified, all fields of the table are subject to output.	0 to 128*1	User
+1 to +□	Field name	Specify each search target field name with 32 characters. The last address in (s3) varies according to the number of fields. □=33×n Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. This setting of this item is not necessary when "0" is specified in (s3)+0. (Ignored if specified.)	—	User

*1 "0" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s3).



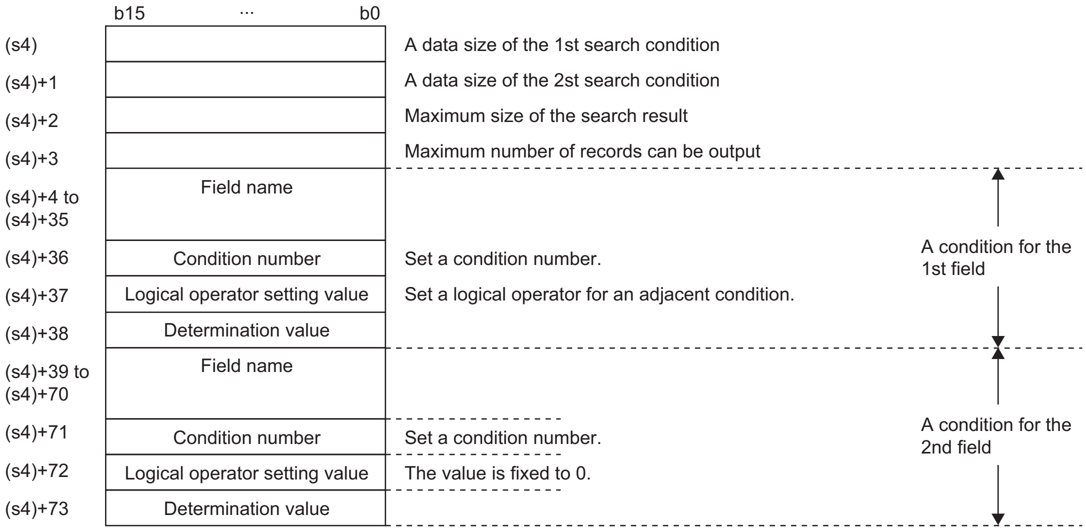
■Search condition

Operand: (s4)					
Device	**Item**	**Description**	**Setting range**	**Set by**	**Data type**
+0	Data size of the first condition	Set the data size of the first search condition in units of words in (s4)+4. If 0 is set in (s4), all records in the table are to be output. • 0: No condition • Other than 0: Data size of field (When (s4) is set to 0, set (s4)+1 also to 0.)	0 to 125	User	WORD
+1	Data size of the second condition	Set the data size of the second search condition in (s4)+4. • 0: No condition • Other than 0: Data size of field	0 to 125	User	
+2	Maximum output size	Set the maximum output size in the search result (d1). • 0: Default value (1024 words)	0 to 32768[*1]	User	
+3	Maximum number of outputs	Set the maximum number of outputs in the search result (d1). • 0: Default number of outputs (16 outputs)	0 to 64	User	
+4 to +□	Search condition	Set the search conditions. The last address in (s4) varies depending on the data type of the determination value. (s4)+4 ⋮ (s4)+35: (1) (s4)+36: (2) (s4)+37: (3) (s4)+38 ⋮ (s4)+□: (4) ■Field name (1) Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating a search condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one search condition is specified, set the logical operator setting value of the first search condition to 0. In this case, the second search condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User	

[*1] "0" to "3072" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the setting format of (s4). (In the case of data type WORD)

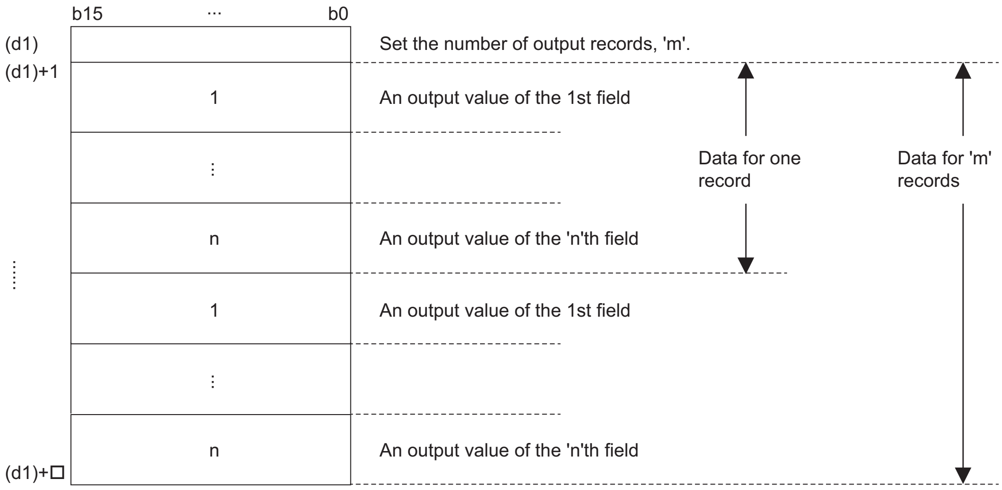
Set as many field names, condition numbers, logical operator setting values, and determination values as there are update conditions. The data size of the determination value follows the size of the data type of each field. (☞ Page 956 DBINSERT(P))



■Search result

Operand: (d1)					
Device	**Item**	**Description**	**Setting range**	**Set by**	**Data type**
+0	Number of outputs	The number of records that meet the condition set in (s4) is output. The number of outputs should be within the maximum number of outputs set in (s4)+3, and any record exceeding the maximum number of outputs is not output.	0 to 64	System	WORD
+1 to +□	Output value	The value of the records that meet the condition set in (s4) is output. Records are output right-justified in the range from (d1) to (d1)+(maximum output size), and any record that does not fit in the range is not output.	Depends on the data type	System	

The following figure shows the setting format of (d1). Data is output according to the data type corresponding to the field specified by (s3).

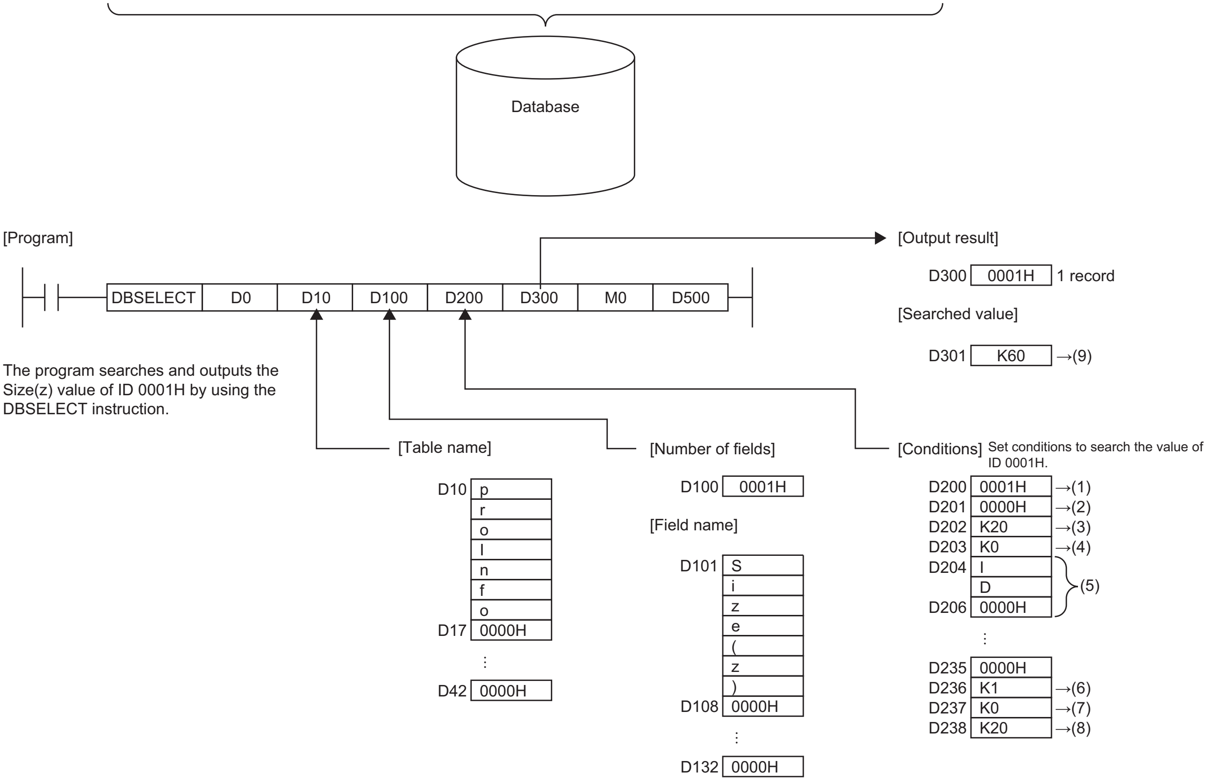


Ex.

When retrieving "Size(z)" with ID = 0001H using the DBSELECT(P) instruction

[Product information table (proInfo)]

ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
⋮				



The program searches and outputs the Size(z) value of ID 0001H by using the DBSELECT instruction.

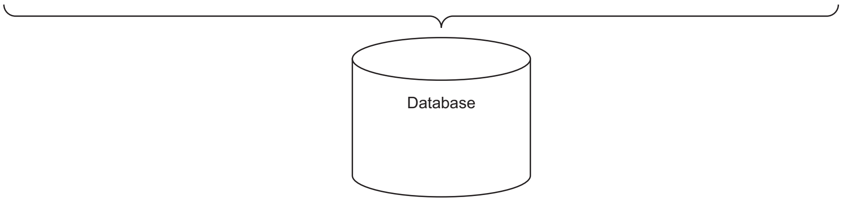
(1) Second condition data size→WORD type (1 word)
(2) 0 for the second condition which is not used
(3) Maximum output size → 20 words
(4) Maximum number of outputs → Default (0)
(5) Field name of first search condition
(6) The condition number is "=". Set K1.
(7) No logical operator setting value is used. Set K0.
(8) Set K1 for the determination value.
(9) K60 is output in WORD type (1 word).

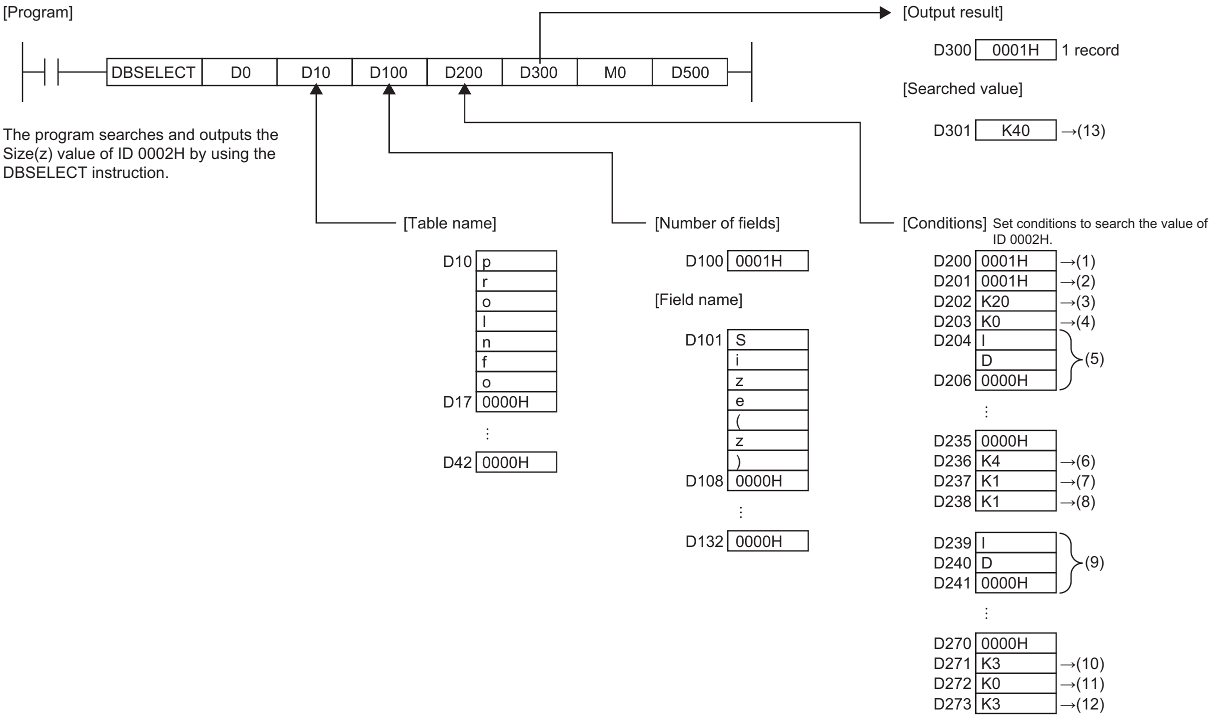
Ex.

When retrieving the "Size(z)" value greater than ID = 0001H and less than ID = 00003H using the DBSELECT(P) instruction

[Product information table (proInfo)]

ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
0003H	"CCC3"	60	110	50
⋮				





The program searches and outputs the Size(z) value of ID 0002H by using the DBSELECT instruction.

(1) Second condition data size→WORD type (1 word)
(2) Second condition data size→WORD type (2 word)
(3) Maximum output size → 20 words
(4) Maximum number of outputs → Default (0)
(5) Field name of first search condition
(6) The condition number is ">". Set K4.
(7) The logical operator setting value is "AND". Set K1.
(8) Set K1 for the determination value.
(9) Field name of second search condition
(10)The condition number is "<". Set K3.
(11)No logical operator setting value is used. Set K0.
(12)Set K3 for the determination value.
(13)K40 is output in WORD type (1 word).

Processing details

- These instructions search the records in the table specified by (s2) in the database corresponding to the identification number specified by (s1). The maximum number of outputs is 64.
- Specify the field name of the record to be searched for in (s3).
- Specify the search conditions in (s4). A maximum of two conditions can be specified.
- The search result is stored in (d1).
- Even when executed from the interrupt program, this instruction performs a record search.
- When the DBSELECT(P) instruction retrieves and outputs a table recipeA record which matches the conditions in which the output fields are 2 and 3, and the condition is field 1=2, the following occurs.

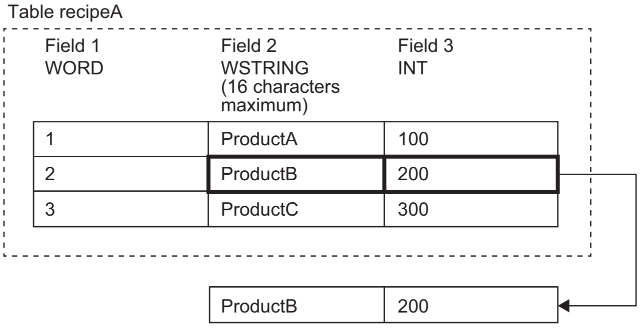
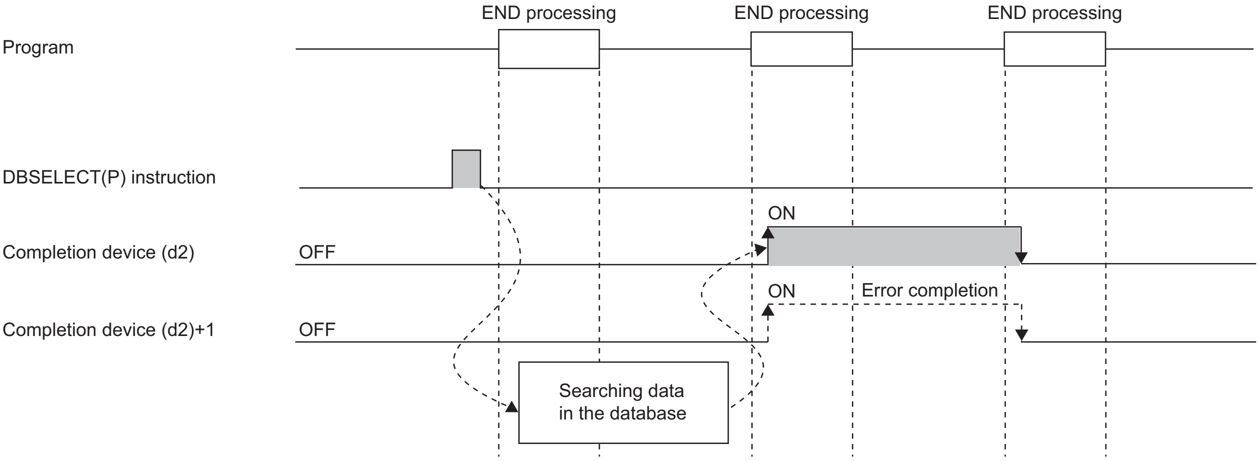


Table recipeA

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300

ProductB	200

- Upon normal completion, the completion signal in the completion device (d2) is turned on and 0 is stored as the completion status in the device (d3).
- The following figure shows the operation of the completion device at completion of the DBSELECT(P) instruction.



- SM753 (File being accessed) turns on while the DBSELECT(P) instruction is executed.[1] While SM753 is on, the DBSELECT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBSELECT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[1] If executed, no processing is performed.

[1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d2)+1 is turned on and an error code is stored as the completion status in the device (d3).

- The DBSELECT(P) instruction is executed during execution of the database access instruction.
- An invalid identification number is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of fields to be searched according to (s3) exceeds the maximum value.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- The number of records to be searched according to (s4) exceeds the maximum value.
- The size of the field to be searched according to (s4) is outside the range.
- Database selection processing failed.
- The number of records output to (d1) exceeds the number specified by (s4).
- The size of the records output to (d1) exceeds the size specified by (s4).
- The logical operator setting value specified by (s4) is out of the range.
- The logical operator setting value specified by (s4)+37 is 1 or 2, and that in (s4)+1 is set to 0.
- 0 is set in (s4) and a value other than 0 is set in (s4)+1.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), (d1), or (d2) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

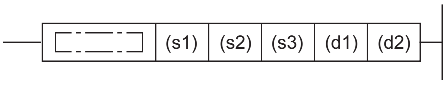
☞ Page 986 Error codes related to database access instructions

Deleting the record in the data base

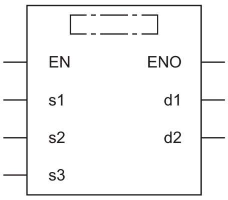
DBDELETE(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

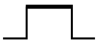
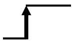
These instructions delete the record that meets the specified condition in the specified table in the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBDELETE(EN,s1,s2,s3,d1,d2); ENO:=DBDELETEP(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBDELETE	
DBDELETEP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database to be deleted.[1]	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the deletion conditions (a maximum of two conditions).	—	Word	ANY16[2]
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] The table names are case-sensitive.

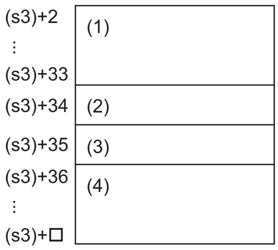
[2] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Deletion condition

Operand: (s3)

Device	Item	Description	Setting range	Set by	Data type
+0	Data size of the first condition	Set the data size of the first deletion condition.	1 to 125	User	WORD
+1	Data size of the second condition	Set the data size of the second deletion condition. • 0: No condition • Other than 0: Data size	0 to 125	User	
+2 to +□	Deletion condition[*1]	Set the deletion conditions. Set as many deletion conditions as there are records to be deleted. The last address in (s3) varies depending on the data type of the determination value.  ■Field name (1) Specify the field name with a Unicode character string in 32 characters (fixed). For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating a deletion condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one deletion condition is specified, set the logical operator setting value of the first deletion condition to 0. In this case, the second deletion condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. The data size of the determination value follows the size of the data type of each field. (☞ Page 956 DBINSERT(P)) Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User	

[*1] Set as many field names, condition numbers, logical operator setting values, and determination values as there are update conditions. (☞ Page 962 DBUPDATE(P))

Processing details

- These instructions delete the record that meets the condition specified by (s3) in the table specified by (s2) in the database corresponding to the identification number specified by (s1).
- Specify the deletion conditions in (s3). A maximum of two deletion conditions can be specified.
- If this instruction is executed by an interrupt program during execution of another instruction, no processing is performed.
- When the DBSELECT(P) instruction deletes a table recipeA record which matches the conditions in which the condition is field 1=2, the following occurs.

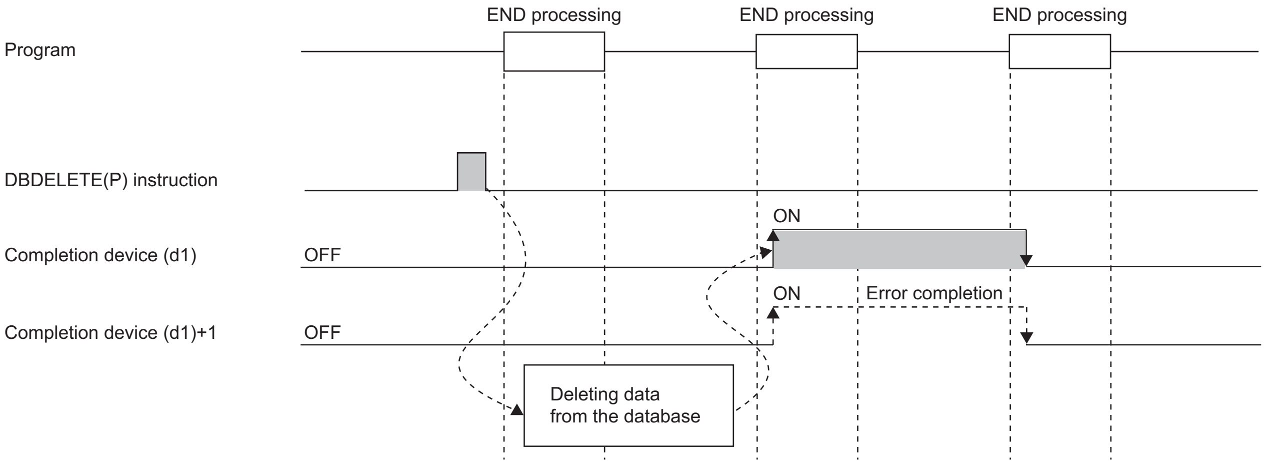
Table recipeA (before deletion)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300

Table recipeA (after deletion)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
3	ProductC	300

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBDELETE(P) instruction.



- SM753 (File being accessed) turns on while the DBDELETE(P) instruction is executed.[2] While SM753 is on, the DBDELETE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBDELETE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[2] If executed, no processing is performed.

[2] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- This instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name to be deleted according to (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the deletion condition.
- Database deletion processing failed.
- An out-of-range value is specified in (s3) for the logical operator setting value.
- The logical operator setting value specified by (s3)+35 is 1 or 2, and that in (s3)+1 is set to 0.
- The field name to be set in (s3)+2 is left unset.
- An out-of-range field name is set in the field name specified by (s3)+2 to (s3)+□.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

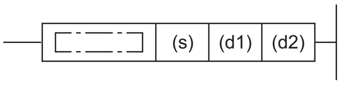
☞ Page 986 Error codes related to database access instructions

Starting a transaction

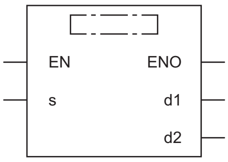
DBTRANS(P)

RnCPU **RnENCPU** ✗ **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

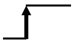
These instructions declare the start of a transaction in relation to the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBTRANS(EN,s,d1,d2); ENO:=DBTRANSP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBTRANS	
DBTRANSP	

Setting data

■Description, range, data type

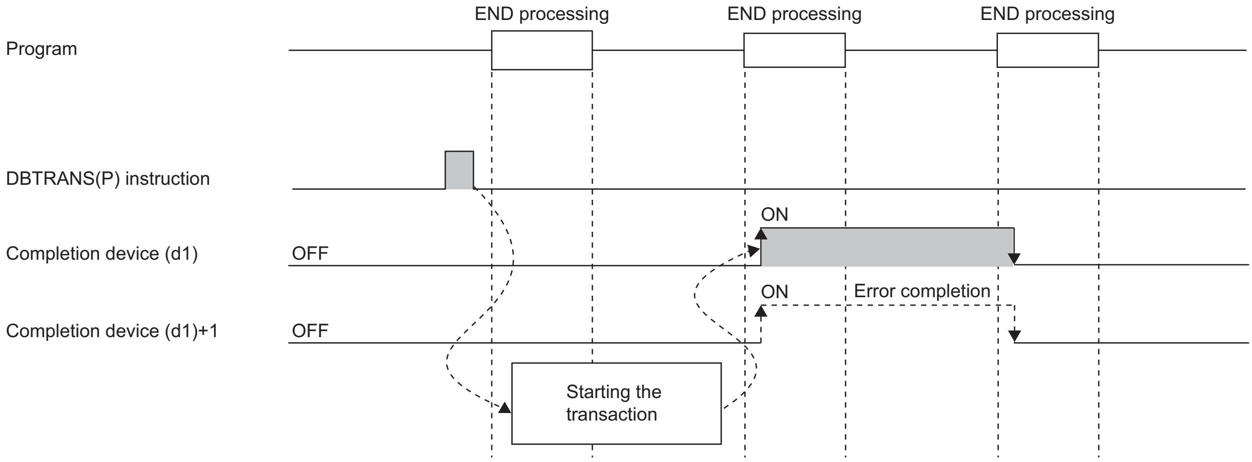
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions declare the start of a transaction in relation to the database corresponding to the identification number specified by (s). However, if a data base other than the one specified by (s) is open, the DBTRANS instruction cannot start a transaction and is completed with an error.
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBTRANS(P) instruction.



- After a transaction is started by the DBTRANS(P) instruction, the DBCOMMIT(P) instruction needs to be executed to determine the transaction or the DBROLBAK(P) instruction needs to be executed to restore the state before the start of the transaction. (If the DBCLOSE(P) instruction is executed before DBCOMMIT(P) or DBROLBAK(P), the transaction is determined in the status at the execution of the DBCLOSE(P) instruction.)
- SM753 (File being accessed) turns on while the DBTRANS(P) instruction is executed.[*1] While SM753 is on, the DBTRANS(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBTRANS(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[*1] If executed, no processing is performed.

[*1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBTRANS(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified.
- The DBTRANS(P) instruction is executed while the transaction has already been started.
- A data base other than the one specified by (s) is open.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	**Description**
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

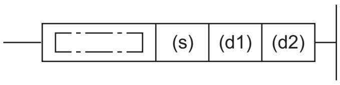
☞ Page 986 Error codes related to database access instructions

Committing a transaction

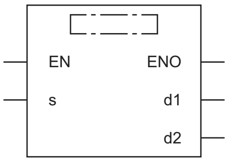
DBCOMMIT(P)

RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

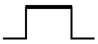
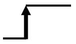
These instructions commit the transaction in relation to the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBCOMMIT(EN,s,d1,d2); ENO:=DBCOMMITP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCOMMIT	
DBCOMMITP	

Setting data

■Description, range, data type

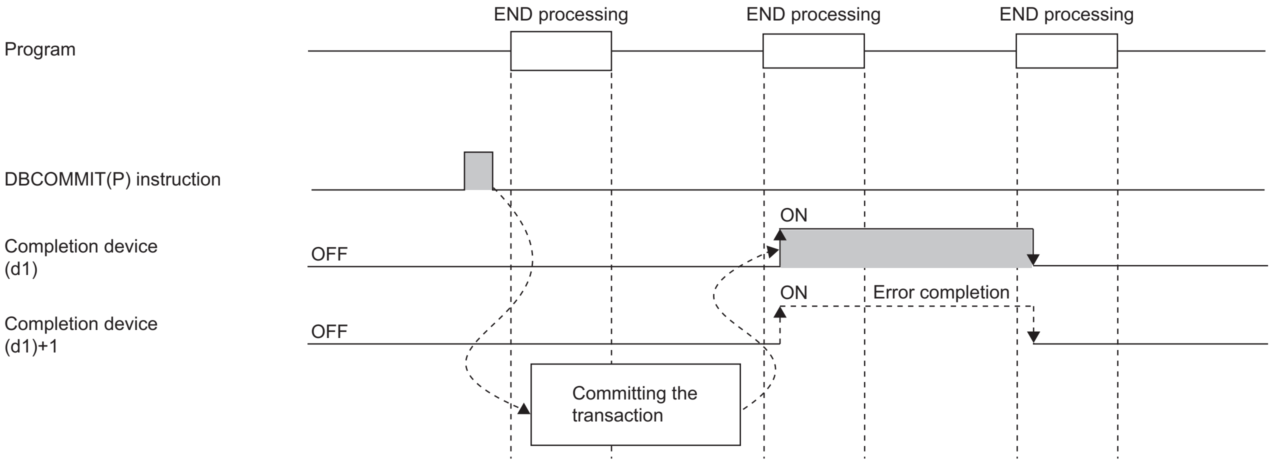
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions commit the transaction in relation to the database corresponding to the identification number specified by (s).
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBCOMMIT(P) instruction.



- SM753 (File being accessed) turns on while the DBCOMMIT(P) instruction is executed.[*1] While SM753 is on, the DBCOMMIT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBCOMMIT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[*1] If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBCOMMIT(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified.
- The DBCOMMIT(P) instruction is executed while no transaction is going on.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.[*2]

*2 For details, refer to the following.
☞ Page 55 Checking the ranges of instruction runtime devices and labels

For the error code stored in the completion status of the operand, refer to the following.

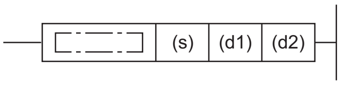
☞ Page 986 Error codes related to database access instructions

Performing a database rollback

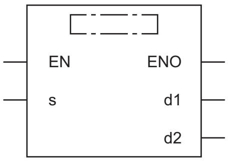
DBROLBAK(P)

RnCPU RnENCPU ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

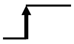
These instructions execute the rollback of the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBROLBAK(EN,s,d1,d2); ENO:=DBROLBAKP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBROLBAK	
DBROLBAKP	

Setting data

■Description, range, data type

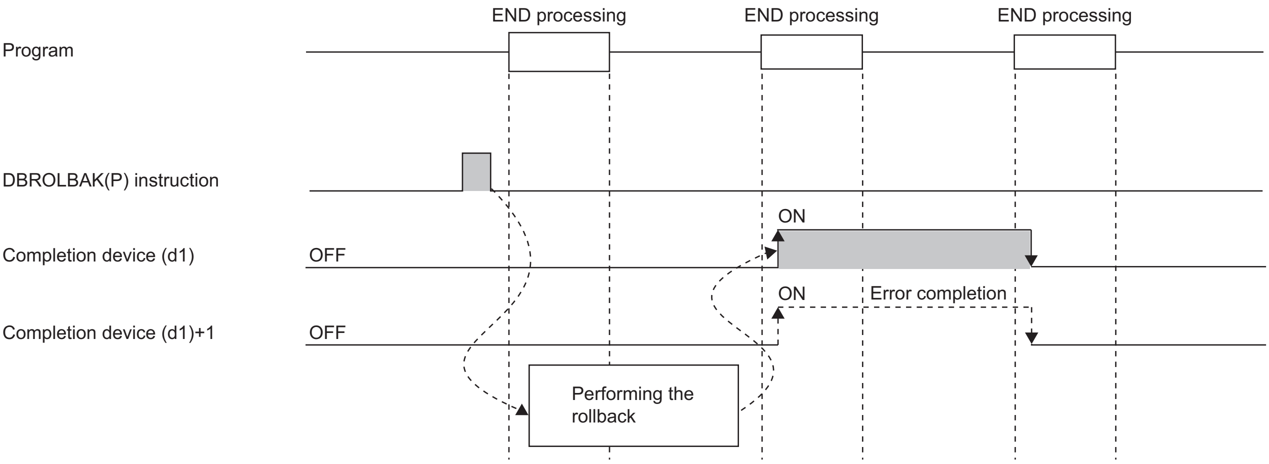
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions execute the rollback in relation to the database corresponding to the identification number specified by (s).
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBROLBAK(P) instruction.



- SM753 (File being accessed) turns on while the DBROLBAK(P) instruction is executed.[1] While SM753 is on, the DBROLBAK(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBROLBAK(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.[1] If executed, no processing is performed.

[1] For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBROLBAK(P) instruction is executed during execution of the database access instruction.
- An identification number outside the specified range is specified.
- The DBROLBAK(P) instruction is executed while no transaction is going on.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 986 Error codes related to database access instructions

Error codes related to database access instructions

The following table lists the error codes that could be stored in the completion status of database access instructions.

Error code	Error content	Action
0103H	CPU internal error	Turn power off and turn it on again or reset the system and check whether the instruction can be executed.
0104H	Another database access instruction is being executed.	Check that another database access instruction is not executed before one database access instruction is completed.
0105H	• The specified path is not a database folder. • The database is broken.	Check whether the folder specified by the path contains a database file. If not, correct the path properly.
0106H	• A file in the database cannot be opened. • A database built with a different firmware version is specified.	Create a new database using the DBIMPORT(P) instruction.
0107H	An invalid drive number is specified by the database access instruction.	Check whether the specified drive number is 2. If not, specify drive number 2 correctly.
0108H	Deletion of the database failed.	Check that another function is not accessing the database and create a new database using the DBIMPORT(P) instruction.
07D1H	The specified table is not found in the database.	Check that the database contains the specified table.
0940H	An invalid character is included in the table name.	Begin the table name with a non-numeric character and do not include an invalid character.
0941H	The table name is duplicated.	Database table names are not case-sensitive. Check whether same characters are used to set Unicode text file table names by differentiating them with uppercase and lowercase characters.
0942H	The specified table is not found in the database.	Check that the database contains the specified table.
0945H	The specified database is in use by another function.	After the database operation by another function has been completed, execute the instruction again.
0946H	The specified table is in use by another function.	After the table operation by another function has been completed, execute the instruction again.
094AH	The database name is not specified.	Specify a correct database name.
094BH	No database is stored in the specified folder.	Check that the folder name is correct and that the folder contains a database-related file.
094CH	The field name contains an invalid character.	Do not include any invalid character in the field name.
094DH	The specified field name is duplicated.	Check for duplicated field names in Unicode text files or instruction arguments. If any, correct it to prevent duplication.
094EH	A non-existent field is specified.	Check that the specified field exists in the table.
0950H	An invalid character is included in the index.	Do not include any invalid character in the index.
0951H	An attempt was made to set empty data (NULL value) in the main key or in the field that has a NOT NULL constraint.	Confirm the table structure. Check whether the field permits NULL. If not, set significant data.
0952H	An attempt was made to add the same value to the field that has a primary key constraint.	Check that the setting value of the record attempted to add to the field that has a primary key constraint is not the same as the record already registered.
0953H	If overlapped data are set to the field where an index is set repeatedly, the data for the index become large and the memory of the CPU module cannot secure enough free space.	Change the table definition where the error is detected and re-build the database.
0954H	The setting value cannot be automatically converted to the format appropriate for the data type.	Check that the format of the setting value meets the specifications.
0959H	The data type that is checked during importing differs from the data type defined in the table.	Check the Unicode text to verify that the setting value to be imported is appropriate for the data type of the field.
095CH	The data types of compared values are incorrect such as comparing the sizes of character strings.	Correct the values to be compared.
095DH	Converting the data type of the setting value specified in an argument of the database access instruction failed and therefore the setting value cannot be stored in the field.	Check that the setting value meets the data type specifications.
095EH	An attempt was made to assign more than one index to one field.	Check that an attempt has not been made to add an index to the field where the primary key or external key is set. Check also that an attempt has not been made to assign more than one index to one field.
095FH	The index name is duplicated.	Check that each index name is not duplicated.
096CH	The setting value to be added to the field where the external key is specified as a key constraint is not included in the setting values of the reference.	Check the setting values of the reference.

Error code	Error content	Action
096EH	The record cannot be deleted because it is referenced by another table.	Check whether the setting values of the record to be deleted include the value of a table referenced by an external key.
0970H	An attempt was made to add or delete a record but failed because no table was available for referencing a setting value included in the record.	Check that a referenced table is available.
0971H	An attempt was made to add a setting value not registered in the referenced table.	Check the setting values in the referenced table.
0972H	The data type in the field referenced by an external key is inconsistent.	Check that the data types in both fields match.
0974H	There is no table to be referenced by an external key.	Specify a correct table name.
0976H	There is no field of the table to be referenced by an external key.	Specify a correct field name.
0984H	A transaction is already running.	Check that multiple transactions are not executed concurrently.
09C2H	An operation is attempted for a database not supported by the CPU module used.	■For a programmable controller CPU with firmware version earlier than "28" After executing the DBEXPORT(P) instruction with a programmable controller CPU with firmware version "28" or later, use the acquired Unicode text file to execute the DBIMPORT(P) instruction with a programmable controller CPU with firmware version earlier than "28". ■For a programmable controller CPU with firmware version "28" or later After executing the DBEXPORT(P) instruction with a programmable controller CPU with firmware version earlier than "28", use the acquired Unicode text file to execute the DBIMPORT(P) instruction with a programmable controller CPU with firmware version "28" or later.
09C3H	A database created by executing the DBIMPORT(P) instruction on a programmable controller CPU with firmware version earlier than "28" was processed with a programmable controller CPU with firmware version "28" or later, and then processed on a programmable controller CPU with firmware version earlier than "28".	• Process the database where an error occurs with a programmable controller CPU with firmware version "28" or later. • To process the database on a programmable controller CPU with firmware version earlier than "28", execute the DBEXPORT(P) instruction on a programmable controller CPU with firmware version "28" or later, and then execute the DBIMPORT(P) instruction on a programmable controller CPU with firmware version earlier than "28" using the output Unicode text file.
09D7H	BOOL data is not specified.	Specify BOOL data.
09F9H	A non-existent field is specified to add a record.	Check the table structure.
0A0AH	A prohibited field name is specified.	Specify another field name.
0A16H	An attempt was made to add a record with a null value set in the field where the primary key was set.	Check the Unicode text to verify that a null value is not set in the field where the primary key is set.
0A17H	An attempt was made to add a record with the same value included in the field where the primary key was set.	Check the Unicode text to verify that the same value is not set in the field where the primary key is set.
0A2BH	An attempt was made to set multiple primary keys.	Check the key setting line in the Unicode text to verify that multiple primary keys are not set.
0A2FH	The primary key does not exist in the referenced table.	Check the Unicode text to verify that the setting value to be imported is appropriate for the data type of the field.
0A30H	The setting values in the field where an external key is set do not include a setting value in the field where the primary key is set.	Check that the setting value in the field where an external key is to be added is included in the setting values in the field where the primary key is set.
0A31H	An attempt was made to change or delete a setting value in the field referenced by an external key.	Check the reference relationships among the tables.
0A50H	An attempt was made to assign more than one index name to the same table.	Check the content of the Unicode text to avoid duplication of index names.
0A5DH	The data types of the fields to be connected do not match.	Check that the data types of the fields to be connected match.
0A5FH	A field name is duplicated in the same table.	Check whether a field name is specified more than once in the same table. If so, correct either field name.
0AAAH	A negative value is specified in the unsigned integral data type field.	Check that the field data type accepts negative values.
0AB1H	An attempt was made to specify a character string for the integral data type.	Check that no character string is specified for an integral data type field.
0AB9H	An attempt was made to assign an index to a field where no value has been set.	Check that values have been set in all field records where an index is to be specified.
0AECH	Table opening failed.	Turn power off and turn it on again or reset the system and check whether the table can be opened.

Error code	Error content	Action
1000H	The Unicode text file specified for importing does not exist.	Check whether the file indicated by the folder path is a Unicode text file. If not, specify a correct folder path.
1002H	There is no identification number that can be allocated.	A maximum of four databases can be opened. Adjust the number of databases that will be opened to 4.
1003H	The identification number of a database which is not open is specified.	Use the DBOPEN(P) instruction to open the database in advance and obtain an identification number.
1004H	The identification number specified by the instruction is out of range.	Specify the identification number obtained by the DBOPEN(P) instruction.
1007H	The number of fields to be specified by the instruction is not specified.	Set 1 or greater for the number of fields in the argument.
1008H	• The number of fields specified by the instruction is out of range. • The field name specified by the instruction is out of range.	• Set 128 or less for the number of fields in the argument.[*1] • Specify a field name within the available range (up to 128th field) in the argument.
1009H	The number of records to be specified by the instruction is not specified.	Set 1 or greater for the number of records in the argument.
100AH	The number of records specified by the instruction is out of range.	Set the number of records within the range in the argument.
100CH	The Unicode text table or set-value delimitation does not follow the format.	Check that the delimiters in the table or set-value delimitations follow the format.
100DH	An error occurred in the database.	Turn power off and turn it on again or reset the system and check whether the same symptom recurs.
100EH	The condition number specified by the instruction is out of range.	Check that the condition number range is correct.
100FH	When two conditions are specified by the instruction, the logical operator setting value is 0.	Check that the logical operator setting value in the argument is 1 or 2.
1010H	The maximum number of databases that can be transacted is exceeded.	Check that a transaction is not executed for two or more databases or that another transaction is not executed while one transaction is already running.
1011H	The commit or rollback instruction is executed without starting a transaction.	Check that the commit or rollback instruction is not executed while no transaction is running.
1012H	The value of the size per record used for adding or update is not appropriate.	Check that the data size of each field matches the size of the data to be added or updated.
1013H	A data type which is not supported is specified.	Check the data type of the Unicode text.
1014H	• Another database is opened during execution of the transaction. • The DBEXPORT(P) instruction is executed during transaction.	• Do not open a database newly during execution of the transaction. • Execute the DBEXPORT(P) instruction after the transaction is completed.
1015H	A transaction is executed for the multiple databases that were already open.	Close the databases that are not targeted for a transaction.
101BH	The first condition is not set in the relevant argument of each instruction.	Check that the field size set as the first condition in the argument is other than 0 or whether the field name is null.
101CH	The data size of a field is not included in multiple condition settings.	Check that the data size of the field specified by the condition is 0.
101DH	A field name is not included in multiple condition settings.	Check that the field name specified by the condition is not null.
101EH	The value of the inter-condition operator is out of range.	Check that the value of the inter-condition operator is correct.
101FH	The value of the inter-condition operator in the second condition setting is out of range.	Check that a value other than 0 is not set in the logical operator setting value of the second condition.
1020H	The setting value is not within the range specified for the data type.	Check that the setting value is within the range.
1021H	An attempt was made to open a database which was already open.	Check that the path corresponding to (s) of the DBOPEN(P) instruction is not specified more than once.
1022H	The DBIMPORT(P) instruction was executed while the database was already open.	Close the database that is already open, and execute the DBIMPORT(P) instruction.
1023H	A table number outside the range is specified.	Check that the table number is within the range.
102EH	The field size specified in the conditions is out of range.	Check that the field size in the conditions is within the range.
1030H	The output size exceeds the setting value.	Adjust the output size.
1031H	The number of outputs exceeds the setting value.	Adjust the number of outputs.
1032H	Since the SD memory card was removed while opening the database with the DBOPEN(P) instruction or executing a database access instruction, the database access instruction cannot be executed or the CPU module database access (from external device) function cannot be executed from an external device.	Power off and on the CPU module or reset it.
1040H	The total size of records to be added exceeds the limit.	Adjust the total size of records to be added.

Error code	Error content	Action
2000H	The format of the Unicode text file is incorrect.	• Check whether the format of the Unicode text file is correct. If not, correct it. • Check whether the number of characters in the comment line is within the range.
2001H	The key setting of the Unicode text file is incorrect.	Check that an out-of-range value is not set with regard to the key constraint of the Unicode text file.
2002H	The number of characters making up a database name, table name, or field name exceeds the limit.	• Check that the database name, table name, or field name in the Unicode text file does not exceed 32 characters. • Check whether 128 or more tabs are used in a single line.
2003H	The number of tables exceeds the limit.	Check that the number of tables in the Unicode text file does not exceed 32.*1
2004H	The format of the setting value of the record in the Unicode text file is incorrect.	Check that the format of the setting value in the Unicode text file is correct.
2005H	The maximum number of records that can be imported is exceeded.	Check that the number of records in the text file does not exceed the maximum number (100,000).
2006H	An invalid character is included in the database name.	Check that an invalid character is not included in the database name in the text file.
2007H	The format of the row index is incorrect.	Check that the format of the row index is correct.
2009H	Failed to read data from the Unicode text file.	• Check the status of the SD memory card. • Check the contents of the Unicode text file.
200AH	Failed to write data to the Unicode text file.	Check the status of the SD memory card.
200BH	The number of fields in the Unicode text file exceeds the maximum number.	Check that the number of fields in the Unicode text file does not exceed 128.*1
200CH	In the Unicode text file, the number of columns set in the field row does not match that set in another row.	Check the Unicode text file to verify that the number of columns set in the field row (number of fields) matches that set in another row.
200DH	The number of indexes in the Unicode text file exceeds the maximum number.	Check that the number of indexes in the Unicode text file does not exceed the number of table fields.
200EH	The number of characters of the main key name in the Unicode text file, external key name, or index name exceeds the maximum number.	Check whether the numbers of key name and index name characters are each 16 or less.
200FH	The database access instruction cannot be executed because the SD memory card is write-protected. Or, startup of the CPU module database access (from external device) function from an external device failed.	Clear the write protection of the SD memory card.
2010H	The database access instruction cannot be executed or the CPU module database access function cannot be executed from an external device because the SD memory card does not have enough free space (10M bytes or more).	Secure enough free space of the SD memory card.
2011H	The number of characters making up the path of the database exceeds 128.	Reduce the number of characters making up the path to 128 or less.
2020H	The CPUDB folder configuration is incorrect.	• Decrease the number of databases in the CPUDB folder to 32 or less. • Delete folders which are not databases in the CPUDB folder. • Refer to the error database check file (2:\CPUDB\ErrorDB.txt) and delete a folder which caused the error.
2021H	The database is broken.	• Create a new database. • Refer to the error database check file (2:\CPUDB\ErrorDB.txt) and delete a database which caused the error.
2023H	Startup of the database failed.	• Power off and on the CPU module or reset it. • Check the data memory capacity and secure a free space of 1K byte or more.
2024H	Suspension of the database failed. Or, the database was suspended because the SD memory card was removed.	• Power off and on the CPU module or reset it. • Insert the SD memory card again, and power off and on the CPU module or reset it.

Error code	Error content	Action
2025H	The database cannot be recognized. Or recovery of the database failed.	• Check the status of the SD memory card. • If the error database check file (2:\CPUDB\ErrorDB.txt) exists and it lists a database which resides under the CPUDB folder or its subfolder, delete the database with its folder and then create a new database. • If the problem persists after powering off and on the CPU module or reset it, or the error database check file (2:\CPUDB\ErrorDB.txt) exists and it lists a database which resides under other than the CPUDB folder or its subfolder, delete the dbmaintainpath.txt file on the $MELPRJ$\$DBASYS$ folder in the SD memory card, delete all database folders (excluding the CPUDB folder), and then create a new database using the DBIMPORT(P) instruction.
Others	• The character code of the Unicode text file is incorrect. • The database to be accessed is in an invalid status.	• Check the character code of the Unicode text file. • If the same error occurs even after powering off and on or reset the CPU module, delete the access-target database folder, and create a new database by using the DBIMPORT(P) instruction.

[1] Use 16 or less fields/tables for a programmable controller CPU with firmware version earlier than "28".

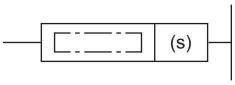
7.22 File Register Operation Instructions

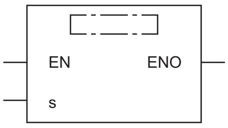
Switching the file register block number

RSET(P)

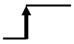
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions change the block number of the file register used in the program.

Ladder	ST
	ENO:=RSET(EN,s); ENO:=RSETP(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
RSET	
RSETP	

Setting data

■Description, range, data type

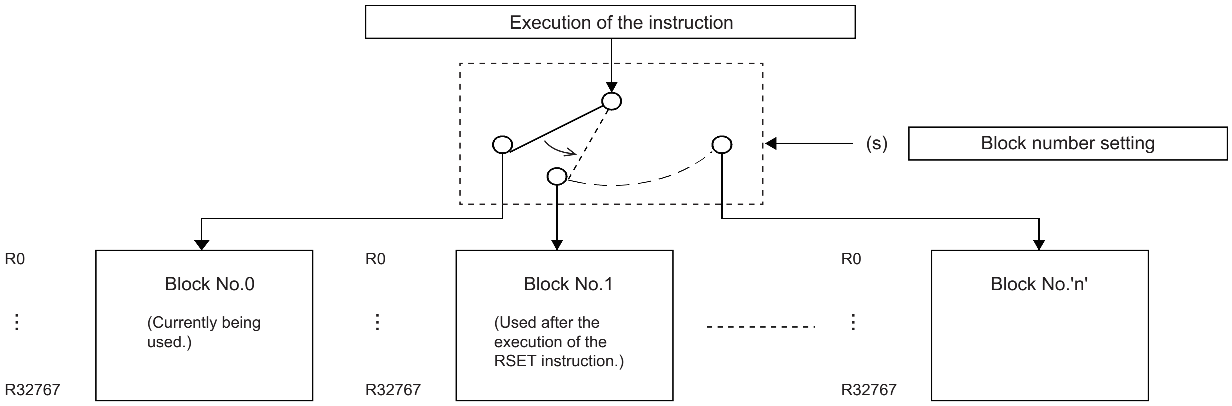
Operand	Description	Range	Data type	Data type (label)
(s)	Block number data to be changed or the device number where the block number data is stored	0 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions switch the block number of the file register used in the program to that stored in the device specified by (s). After the block number is changed, all file registers processed by the sequence program are those linked to the new block number.



Precautions

For the restrictions of the file register, refer to the following.

☞ Page 65 Restrictions on using file registers

Operation error

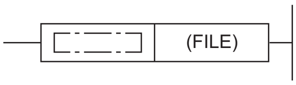
Error code (SD0)	Description
2820H	The specified file register does not exist.
3405H	The block number in the device specified by (s) does not exist.

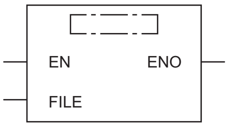
Changing the file register file name

QDRSET(P)

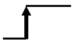
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✗

These instructions change the file name of the file register used in the program to that stored in the device specified by (file name).

Ladder	ST
 FILE: File name	ENO:=QDRSET(EN,filename); ENO:=QDRSETP(EN,filename);

FBD/LD
 FILE: File name

■Execution condition

Instruction	Execution condition
QDRSET	
QDRSETP	

Setting data

■Description, range, data type

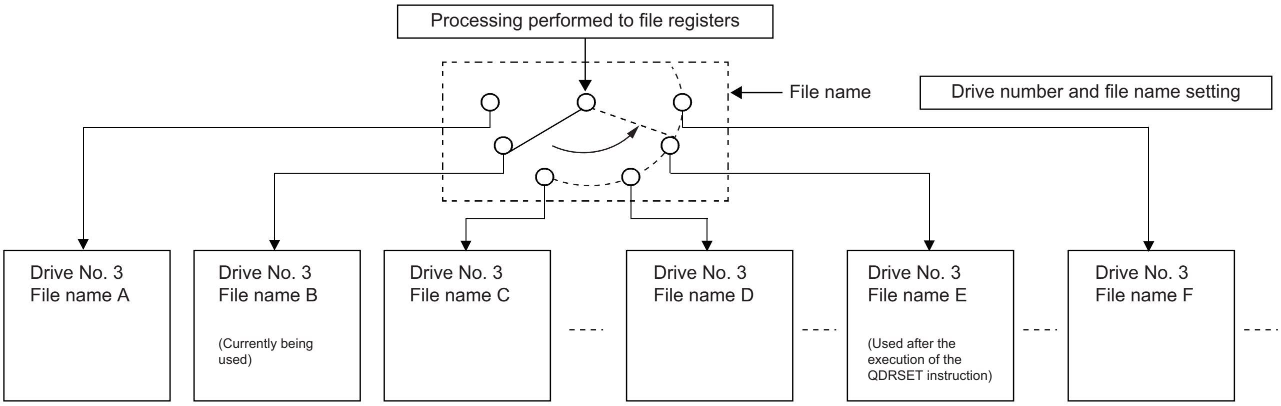
Operand	Description	Range	Data type	Data type (label)
(File name)	String data of drive number file name of the file register to be set, or the start device where the string data is stored Example: "1: ABC"	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the file name of the file register used in the program to that stored in the device specified by (file name). After the file name is changed, all file registers processed by the program are those linked to block number 0 of the new file name. The RSET(P) instruction is used to change the block number.



- For the drive number, 1 or 3 can be specified.
- When drive 1 is specified, the drive works as drive 3. The use status of the drive is reflected to SD614. It is not reflected to SD604.
- Extension ".QDR" need not be specified for the file name.
- Even if the drive number file name is specified by a parameter, the file name specified by the QDRSET(P) instruction takes precedence.

Point

- If the file name is changed by the QDRSET(P) instruction, operating the CPU module switch from STOP to RUN restores the file name specified by the parameter. To continue to use the file name specified by the QDRSET(P) instruction even when the CPU module switch is changed from STOP to RUN, execute the QDRSET(P) instruction using SM402 that triggers one scan when the switch is changed from STOP to RUN.
- When a file register is specified for the refresh device, do not use the QDRSET(P) instruction to change the file name of the file register.

Precautions

- Even when the NULL code (0000H) is specified for the file name, the file name setting is not cleared and no processing is performed.
- For the restrictions of the file register, refer to the following.

☞ Page 65 Restrictions on using file registers

Operation error

Error code (SD0)	Description
2840H	The drive number/file name specified by (file name) does not exist.
3405H	Out-of-range data is set to (file name). • A drive number other than 1 and 3 is specified. • Only the drive number is specified.

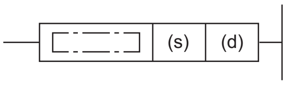
7.23 File Register Read/Write Instructions

Reading 1-byte data from the file register

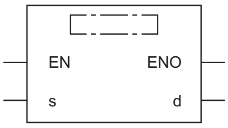
ZRRDB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

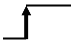
These instructions read the data from the file register with the specified serial byte number.

Ladder	ST
	ENO:=ZRRDB(EN,s,d); ENO:=ZRRDBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZRRDB	
ZRRDBP	

Setting data

■Descriptions, ranges, and data types

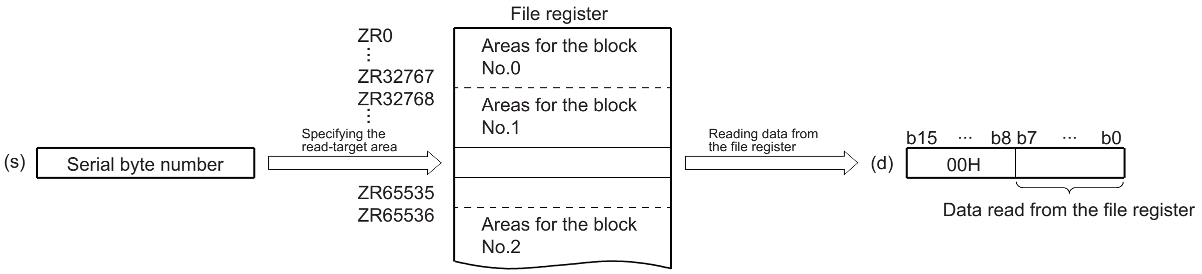
Operand	Description	Range	Data type	Data type (label)
(s)	Serial byte number of the file register to be read	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	Start number of the device for storing the data that has been read	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

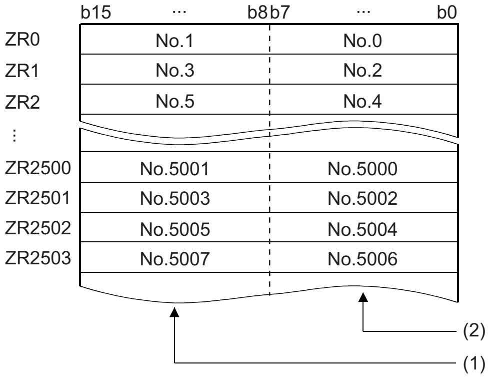
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- Without recognizing block numbers, these instructions read the data from the file register with the serial byte number specified by (s), and store the data in the lower 8 bits of the device specified by (d). 00H is stored in the upper 8 bits of the device specified by (d).



- The following figure shows the file register numbers corresponding to serial byte numbers.



(1) Data areas when an odd number is specified

(2) Data areas when an even number is specified

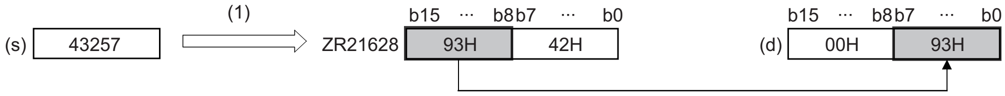
Ex.

When 23560 is specified in (s), the data in the lower 8 bits of ZR11780 is read.

(1) Specifying the read-target area

Ex.

When 43257 is specified in (s), the data in the upper 8 bits of ZR21628 is read.



(1) Specifying the read-target area

Operation error

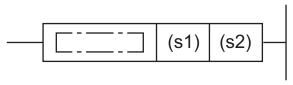
Error code (SD0)	Description
2820H	The specified device number (serial byte number) is out of range.

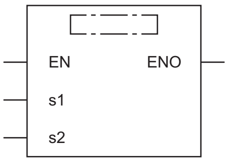
Writing 1-byte data to the file register

ZRWRB(P)

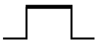
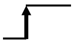
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions write the data in the lower bits of the specified device to the file register with the specified serial byte number.

Ladder	ST
	ENO:=ZRWRB(EN,s1,s2); ENO:=ZRWRBP(EN,s1,s2);

FBD/LD


■Execution condition

Instruction	Execution condition
ZRWRB	
ZRWRBP	

Setting data

■Descriptions, ranges, and data types

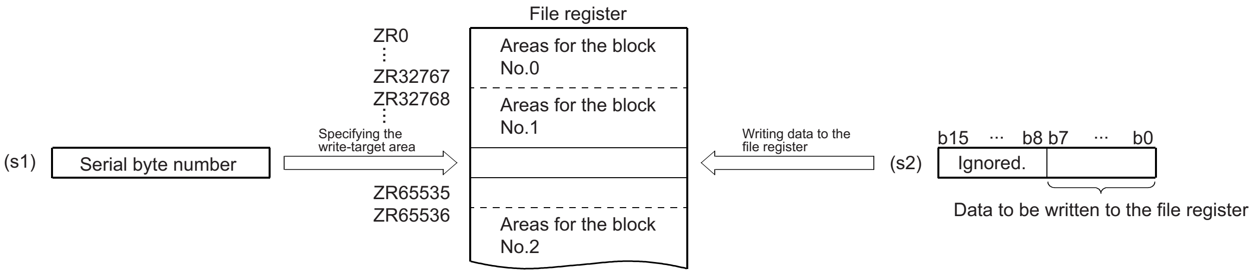
Operand	Description	Range	Data type	Data type (label)
(s1)	Serial byte number of the file register to be written	0 to 4294967295	32-bit unsigned binary	ANY32
(s2)	Device number where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

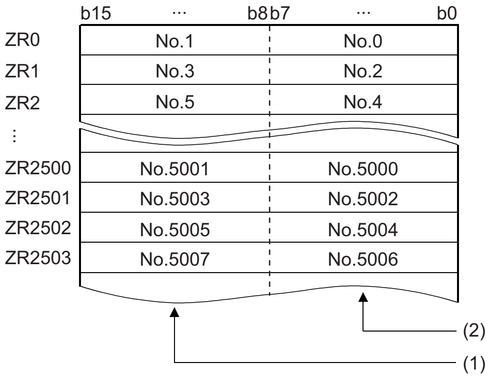
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- Without recognizing block numbers, these instructions write the lower 8-bit data stored in the device specified by (s2) to the file register with the serial byte number specified by (s1). The instructions ignore upper 8-bit data in the device specified by (s2).



- The following figure shows the file register numbers corresponding to serial byte numbers.

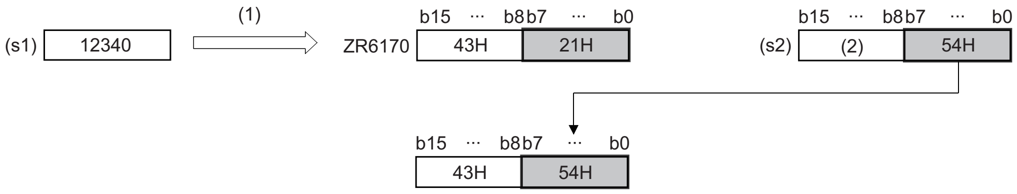


(1) Data areas when an odd number is specified

(2) Data areas when an even number is specified

Ex.

When 12340 is specified in (s1), data is written to the lower 8 bits of ZR11170.

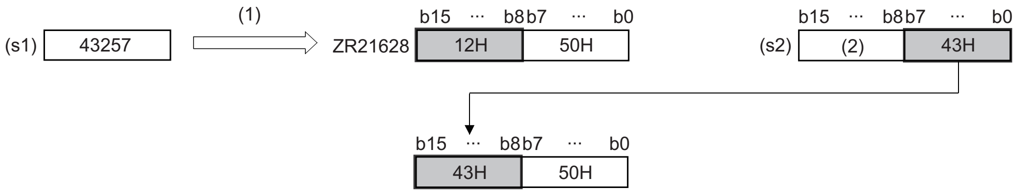


(1) Specifying the write-target area

(2) Ignored

Ex.

When 43257 is specified in (s1), data is written to the upper 8 bits of ZR21628.



(1) Specifying the write-target area

(2) Ignored

Operation error

Error code (SD0)	Description
2820H	The specified device number (serial byte number) is out of range.

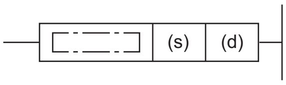
7.24 Indirect Address Read Instructions

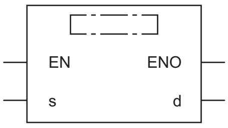
Reading the indirect address

ADRSET(P)

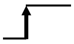
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions read the indirect address of the specified device.

Ladder	ST
	ENO:=ADRSET(EN,s,d); ENO:=ADRSETP(EN,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
ADRSET	
ADRSETP	

Setting data

■Descriptions, ranges, and data types

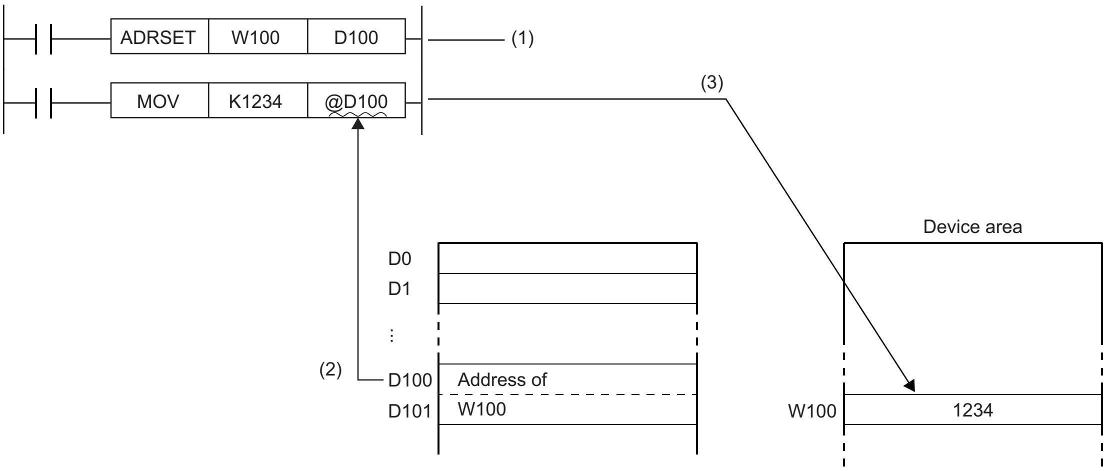
Operand	Description	Range	Data type	Data type (label)
(s)	Number of the device from which the indirect address is to be read	—	Device name	ANY_ELEMENTARY
(d)	Start number of the device for storing the indirect address of the device specified by (s)	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	○	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the indirect address of the device specified by (s), and stores it in the device specified by (d). The address stored in the devices specified by (d)+0 and (d)+1 is used for indirect addressing of the device in the program.



(1) The address of W100 is stored in D100 and D101.
(2) The data (address of W100) stored in D100 and D101 is specified.
(3) "1234" is written to W100.

- Digit specification of bit device or bit specification of word device in (s) is not permitted
- For the indirect specification of devices, refer to the following.

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

- Index specification of labels is not supported. When using a label by specifying it in (d), follow the method below.

Label	Description

Global label specifying a device	When using it as an indirect address, use the indirect specification of the device specified in the global label.
	■Example of ST
	ADRSET(TRUE , intVar , gvAddr); // Read the intVar indirect address to gvAddr.
	INC (TRUE , @D0);// Use the indirect specification of device D0 specified for gvAddr.

Automatic assignment global label/ local label	Transfer the indirect address to the device and use the indirect specification of the transfer destination device.
	■Example of ST
	ADRSET(TRUE , intVar , lvAddr); // Read the intVar indirect address to lvAddr.
	DMOV(TRUE , lvAddr , D0);// Transfer the indirect address, which has been read to lvAddr, to the device.
	INC (TRUE , @D0);// Use the indirect specification of the device to which the indirect address was transferred.

Operation error

There is no operation error.

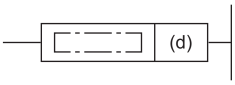
7.25 Clock Instructions

Reading clock data

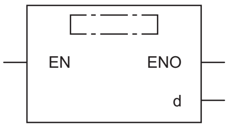
DATERD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

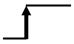
These instructions read "year, month, day, hour, minute, second, and day of week" from the clock element of the CPU module.

Ladder	ST
	ENO:=DATERD(EN,d); ENO:=DATERDP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DATERD	
DATERDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the clock data that has been read	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 7)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read "year, month, day, hour, minute, second, and day of week" from the clock element of the CPU module, and store the read data in binary in the device specified by (d) and later.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6
(Clock element)	Year	Month	Day	Hour	Minute	Second	Day of week
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6

- "Year" stored in the device specified by (d) is a 4-digit year.
- "Day of week" stored in the device specified by (d)+6 is a number from 0 to 6 corresponding to Sunday to Saturday.

Day of week	Day	Month	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

- Data is automatically corrected in leap years.

Operation error

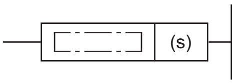
There is no operation error.

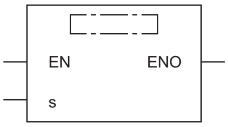
Writing clock data

DATEWR(P)

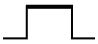
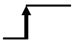
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions write the clock data stored in the specified device and later to the clock element of the CPU module.

Ladder	ST
	ENO:=DATEWR(EN,s); ENO:=DATEWRP(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
DATEWR	
DATEWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the clock data to be written to the clock element is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 7)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions write the clock data stored in the device specified by (s) and later to the clock element of the CPU module.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6
(Clock element)	**Year**	**Month**	**Day**	**Hour**	**Minute**	**Second**	**Day of week**
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6

- Set each data in binary.
- Set the year data in the range from 1980 to 2079 in the device specified by (s).
- Set the month data in the range from 1 to 12 in the device specified by (s)+1.
- Set the day data in the range from 1 to 31 in the device specified by (s)+2.
- Set the hour data in the range from 0 to 23 in the device specified by (s)+3. (Set in 24-hour format.)
- Set the minute data in the range from 0 to 59 in the device specified by (s)+4.
- Set the second data in the range from 0 to 59 in the device specified by (s)+5.
- Set the day of week in the range from 0 to 6 corresponding to Sunday to Saturday in the device specified by (s)+6.

Day of week	Day	Month	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

Operation error

Error code (SD0)	Description
3405H	The data in the device specified (s) is out of range.
3425H	A time value less than an hour from the start time of the daylight-saving time is set in (s) to (s)+6.

Point

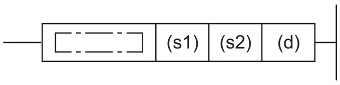
When clock data is changed, "clock setting" (event code: 24000) is saved to the event history. That is, "clock setting" is saved to the event history when this instruction is executed.

Adding clock data

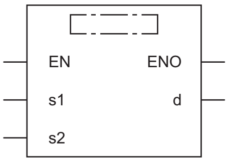
DATE+(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add time data.

Ladder	ST[*1]
	ENO:=DATEPLUS(EN,s1,s2,d); ENO:=DATEPLUSP(EN,s1,s2,d);

FBD/LD

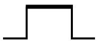
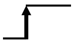


(□ is replaced by either of the following: DATEPLUS, DATEPLUSP.)

---

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DATE+	
DATE+P	

Setting data

■Descriptions, ranges, and data types

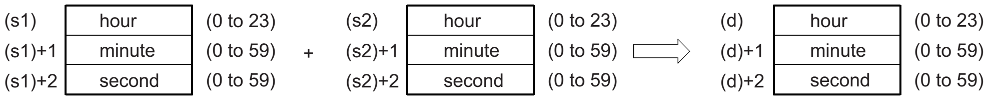
Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the augend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device where the addend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the addition result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

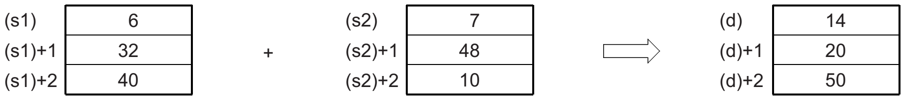
Processing details

- These instructions add the time data in the device specified by (s2) to the time data in the device specified by (s1), and store the addition result in the device number specified by (d) and later.

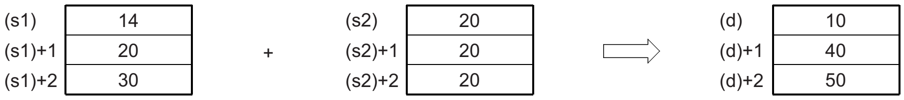


Ex.

7:48:10 is added to 6:32:40.



- If the time obtained as the result of addition exceeds 24 hours, 24 hours are subtracted from the resultant time to produce the operation result. For example, when 20:20:20 is added to 14:20:30, the operation result is 10:40:50 rather than 34:40:50.



Operation error

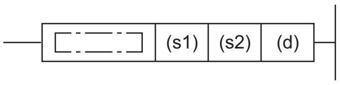
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Subtracting clock data

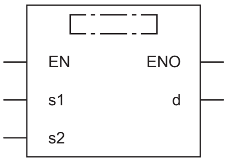
DATE-(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions subtract time data.

Ladder	ST[1]
	ENO:=DATEMINUS(EN,s1,s2,d); ENO:=DATEMINUSP(EN,s1,s2,d);

FBD/LD

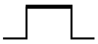
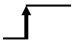


(□ is replaced by either of the following: DATEMINUS, DATEMINUSP.)

---

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DATE-	
DATE-P	

Setting data

■Descriptions, ranges, and data types

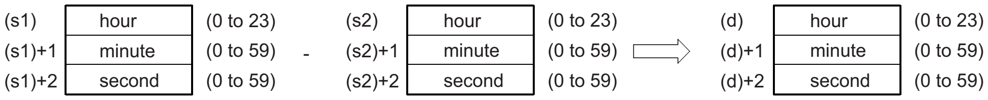
Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where minuend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device where the subtrahend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the subtraction result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

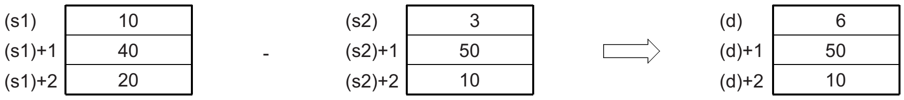
Processing details

- These instructions subtract the time data in the device specified by (s2) from the time data in the device specified by (s1), and store the subtraction result in the device number specified by (d) and later.

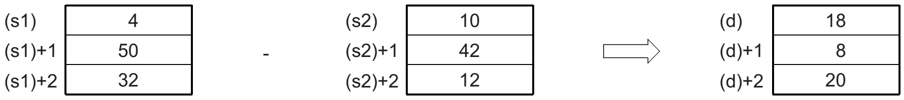


Ex.

3:50:10 is subtracted from 10:40:20.



- If the time obtained as the result of subtraction becomes a negative value, 24 hours are added to the resultant time to produce the operation result. For example, when 10:42:12 is subtracted from 4:50:32, the operation result is 18:8:20 rather than -6:8:20.



Operation error

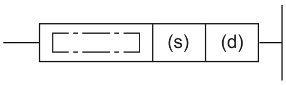
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Converting time data from hour/minute/second to second

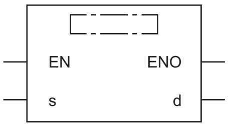
TIME2SEC(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

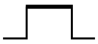
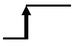
These instructions convert time data (hour, minute, second) to second data.

Ladder	ST
	ENO:=TIME2SEC(EN,s,d); ENO:=TIME2SECP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TIME2SEC	
TIME2SECP	

Setting data

■Description, range, data type

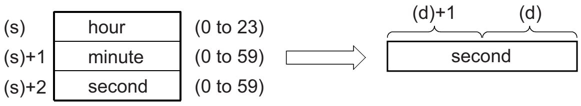
Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the time data to be converted is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the clock data after conversion	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

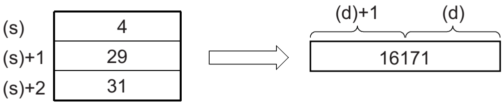
Processing details

- These instructions convert the time data stored in the device number specified by (s) and later to second data, and store the operation result in the device specified by (d).



Ex.

4:29:31 is specified in (s).



Operation error

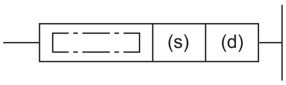
Error code (SD0)	Description
3405H	The data in the device specified (s) is out of range.

Converting time data from second to hour/minute/second

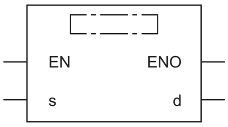
SEC2TIME(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

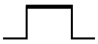
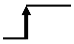
These instructions convert second data to time data (hour, minute, second).

Ladder	ST
	ENO:=SEC2TIME(EN,s,d); ENO:=SEC2TIMEP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SEC2TIME	
SEC2TIMEP	

Setting data

■Descriptions, ranges, and data types

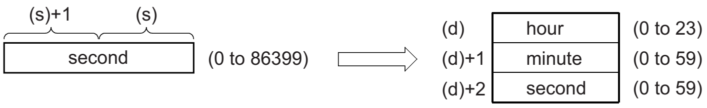
Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the time data to be converted is stored	0 to 86399	32-bit signed binary	ANY32
(d)	Start device for storing the clock data after conversion	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

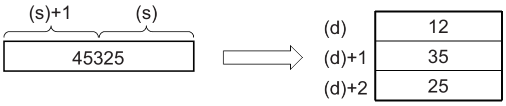
Processing details

- These instructions convert the seconds data stored in the device number specified by (s) and later to time data (hour, minute, second), and store the operation result in the device specified by (d) and later.



Ex.

45325 seconds are specified in (s).



Operation error

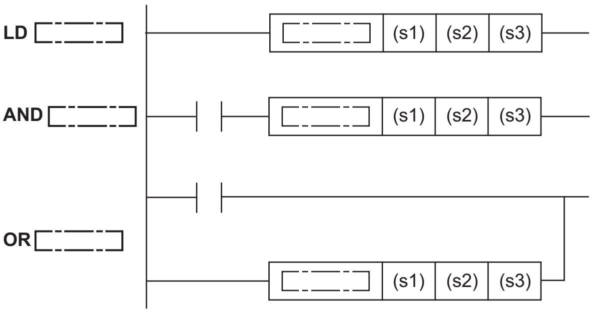
Error code (SD0)	Description
3405H	The data in the device specified (s) is out of range.

Comparing date data

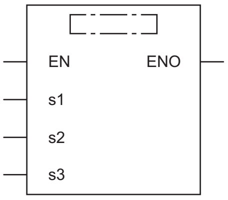
LDDT□, ANDDT□, ORDT□

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions compare the specified date data, or compare the date data with the current date.

Ladder	ST[*1]
 (□ indicates T=, DT<>, DT>, DT<=, DT<, or DT>=.)	ENO:=LDDT_□(EN,s1,s2,s3); ENO:=ANDDT_□(EN,s1,s2,s3); ENO:=ORDT_□(EN,s1,s2,s3); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[*2]

FBD/LD



(□ is replaced by a combination of LDDT_, ANDDT_, or ORDT_ and EQ, NE, GT, LE, LT, or GE.)[*2]

[*1] The engineering tool with version "1.035M" or later supports the ST.

[*2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDDT□, ANDDT□, ORDT□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_DT[*1]
(s2)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_DT[*1]
(s3)	Comparison target setting value or the number of comparison target data	0001H to 0007H, 8001H to 8007H	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

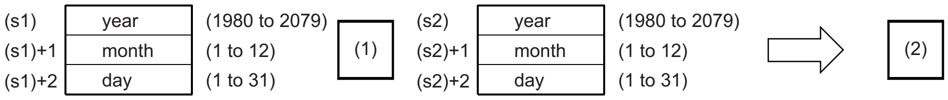
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the date data in the devices specified by (s1) and (s2), or compare the date data in the device specified by (s1) with the current date. Set the comparison target by (s3).

• Comparing two specified date data

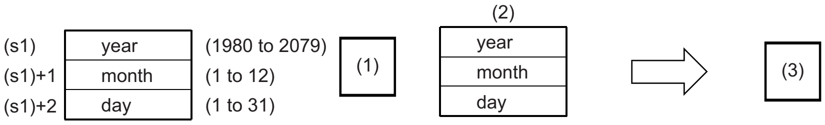
These instructions compare the date data in the device specified by (s1) with the date data in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

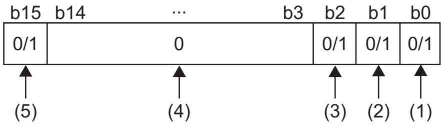
• Comparing the specified date data with the current date

These instructions compare the date data in the device specified by (s1) with the current date data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The date data in the device specified by (s2) is regarded as dummy data and ignored.



(1)Relational operator
(2)Current date data
(3)Comparison result

- Set each data in binary.
- Set the year data in the range from 1980 to 2079 in the devices specified by (s1) and (s2).
- Set the month data in the range from 1 to 12 in the devices specified by (s1)+1 and (s2)+1.
- Set the date data in the range from 1 to 31 in the devices specified by (s1)+2 and (s2)+2.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).



(1) Set "day" as comparison target.
(2) Set "month" as comparison target.
(3) Set "year" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current date in accordance with the conditions set in the 0 to 2 bits.

- When 0 is set to the 0 to 2 bits, the date data are not compared. When 1 is set, the entire date data (year, month, and day) are compared.
- When 0 is set to the 15 bit, the data in the device specified by (s1) and the date data in the device specified by (s2) are compared. When 1 is set, the data in the device specified by (s1) is compared with the current date. The date data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

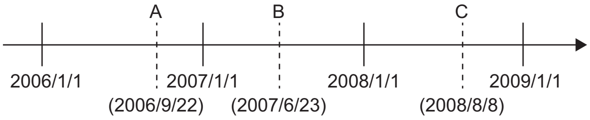
(s3) value when comparing two specified date data	(s3) value when comparing the specified date data with the current date	Comparison target	Description
0001H	8001H	Day	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Month	Only data in the device specified by (s1)+1 is compared.
0003H	8003H	Month, day	Data in the device areas specified by (s1)+1 and (s1)+2 are compared.
0004H	8004H	Year	Only data in the device specified by (s1) is compared.
0005H	8005H	Year, day	Data in the device areas specified by (s1) and (s1)+2 are compared.
0006H	8006H	Year, month	Data in the device areas specified by (s1) and (s1)+1 are compared.
0007H	8007H	Year, month, day	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared.
Other than 0001H to 0007H, 8001H to 8007H		None	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.)

- If the comparison target data in the device are not recognized as date data, SM709 turns on after the instruction is executed and the operation result will be non-continuity. Even if the data are not recognized as date data, SM709 does not turn on if the data are within the setting range. If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the setting area in the device/label memory, SM709 turns on after the instruction is executed and the operation result will be non-continuity as well. Once SM709 turns on, the on state is held until the CPU module is powered off or reset. Turn off SM 709 as needed.
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
DT=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
DT<>, NE	(s1)≠(s2)	
DT>, GT	(s1)>(s2)	
DT<=, LE	(s1)≤(s2)	
DT<, LT	(s1)<(s2)	
DT>=, GE	(s1)≥(s2)	
DT=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
DT<>, NE	(s1)=(s2)	
DT>, GT	(s1)≤(s2)	
DT<=, LE	(s1)>(s2)	
DT<, LT	(s1)>≥(s2)	
DT>=, GE	(s1)<(s2)	

Ex.

The date data A, B, and C are compared.



- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

Comparison target data	**Condition**[*1]		
	A<B	**B<C**	**A<C**
Day	Continuity	Non-continuity	Non-continuity
Month	Non-continuity	Continuity	Non-continuity
Month, day	Non-continuity	Continuity	Non-continuity
Year	Continuity	Continuity	Continuity
Year, day	Continuity	Continuity	Continuity
Year, month	Continuity	Continuity	Continuity
Year, month, day	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

[*1] In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- Even though the specified date does not exist, the comparison operation is performed in accordance with the conditions in the following table as long as the date data are within the valid range.
  - Date A: 2006/02/30 (Even though the date does not exist, this date can be set.)
  - Date B: 2007/03/29
  - Date A: 2008/02/31 (Even though the date does not exist, this date can be set.)

Comparison target data	**Condition**[*2]		
	A<B	**B<C**	**A<C**
Day	Non-continuity	Non-continuity	Continuity
Month	Non-continuity	Non-continuity	Non-continuity
Month, day	Continuity	Non-continuity	Continuity
Year	Continuity	Continuity	Continuity
Year, day	Continuity	Continuity	Continuity
Year, month	Continuity	Continuity	Continuity
Year, month, day	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

[*2] In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- If the LDDT_□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORDT_□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

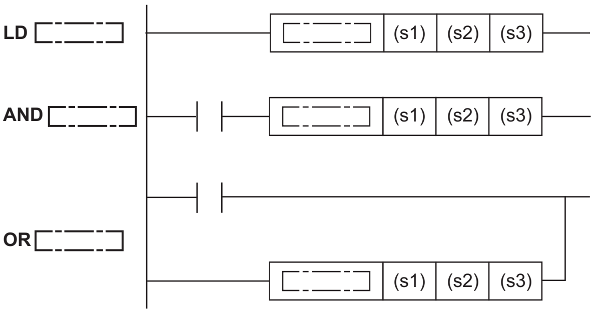
There is no operation error.

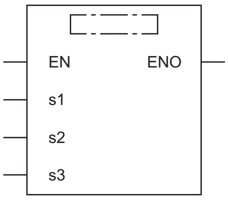
Comparing time data

LDTM□, ANDTM□, ORTM□

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

These instructions compare the specified time data, or compare the specified time data with the current time.

Ladder	ST[1]
 (□ is replaced by TM=, TM<>, TM>, TM<=, TM<, or TM>=.)	ENO:=LDTM_□(EN,s1,s2,s3); ENO:=ANDTM_□(EN,s1,s2,s3); ENO:=ORTM_□(EN,s1,s2,s3); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)[2]

FBD/LD
 (□ is replaced by a combination of LDTM_, ANDTM_, or ORTM_ and EQ, NE, GT, LE, LT, or GE.)[2]

[1] The engineering tool with version "1.035M" or later supports the ST.

[2] EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDTM□, ANDTM□, ORTM□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_TM[1]
(s2)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_TM[1]
(s3)	Comparison target setting value or the number of comparison target data	0001H to 0007H, 8001H to 8007H	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

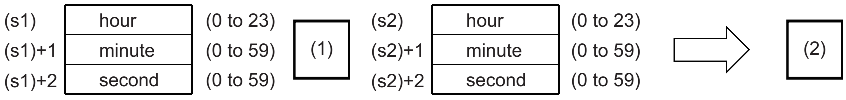
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the time data in the device specified by (s1) and (s2), or compare the time data in the device specified by (s1) with the current time. Set the comparison target by (s3).

• Comparing two specified time data

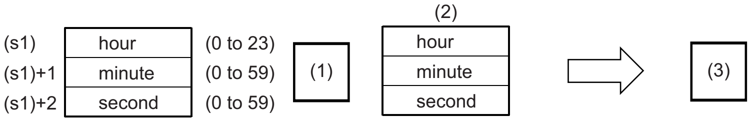
These instructions compare the time data in the device specified by (s1) with the time data (hour, minute, second) in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

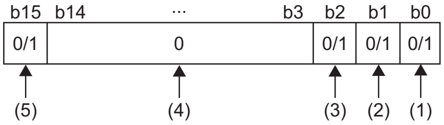
• Comparing the specified time data with the current time data

These instructions compare the time data in the device specified by (s1) with the current time data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The time data in the device specified by (s2) is regarded as dummy data and ignored.



(1)Relational operator
(2)Current time data
(3)Comparison result

- Set each data in binary.
- Set the hour data in the range from 0 to 23 in 24-hour format in the devices specified by (s1) and (s2).
- Set the minute data in the range from 0 to 59 in the devices specified by (s1)+1 and (s2)+1.
- Set the second data in the range from 0 to 59 in the devices specified by (s1)+2 and (s2)+2.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).



(1) Set "second" as comparison target.
(2) Set "minute" as comparison target.
(3) Set "hour" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current time in accordance with the conditions set in the 0 to 2 bits.

- When 0 is set to bits 0 to 2, the time data are not compared. When 1 is set, the comparison target time data (hour, minute, second) are compared.
- When 0 is set to bit 15, the data in the device specified by (s1) and the time data in the device specified by (s2) are compared. When 1 is set, the time data in the device specified by (s1) is compared with the current time. The time data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

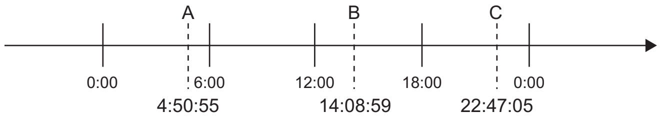
(s3) value when comparing two specified time data	(s3) value when comparing with current time data	Comparison target time	Description
0001H	8001H	Second	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Minute	Only data in the device specified by (s1)+1 is compared.
0003H	8003H	Minute, second	Data in the device areas specified by (s1)+1 and (s1)+2 are compared.
0004H	8004H	Hour	Only data in the device specified by (s1) is compared.
0005H	8005H	Hour, second	Data in the device areas specified by (s1) and (s1)+2 are compared.
0006H	8006H	Hour, minute	Data in the device areas specified by (s1) and (s1)+1 are compared.
0007H	8007H	Hour, minute, second	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared.
Other than 0001H to 0007H, 8001H to 8007H		None	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.)

- If the comparison target data in the device are not recognized as time data, SM709 turns on after the instruction is executed and the operation result will be non-continuity (ENO OFF). If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the setting area in the device/label memory, SM709 turns on after the instruction is executed and the operation result will be non-continuity (ENO OFF) as well. Once SM709 turns on, the on state is held until the CPU module is powered off or reset. Turn off SM 709 as needed.
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
TM=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
TM<>, NE	(s1)≠(s2)	
TM>, GT	(s1)>(s2)	
TM<=, LE	(s1)≤(s2)	
TM<, LT	(s1)<(s2)	
TM>=, GE	(s1)≥(s2)	
TM=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
TM<>, NE	(s1)=(s2)	
TM>, GT	(s1)≤(s2)	
TM<=, LE	(s1)>(s2)	
TM<, LT	(s1)≥(s2)	
TM>=, GE	(s1)<(s2)	

Ex.

The time data A, B, and C are compared.



- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

Comparison target data	**Condition**[*1]		
	A<B	**B<C**	**A<C**
Second	Continuity	Non-continuity	Non-continuity
Minute	Non-continuity	Continuity	Non-continuity
Minute, second	Non-continuity	Continuity	Non-continuity
Hour	Continuity	Continuity	Continuity
Hour, second	Continuity	Continuity	Continuity
Hour, minute	Continuity	Continuity	Continuity
Hour, minute, second	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

[*1] In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- If the LDTM_☐ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORTM_☐ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

There is no operation error.

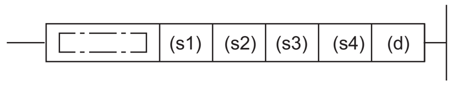
Outputting a comparison result of time data

TCMP(P)

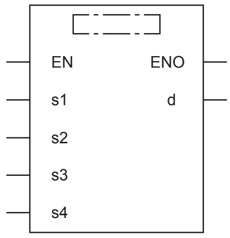
RnCPU **RnENCPU** ✕ **RnPCPU (Process)** ✕ **RnPCPU (Redundant)** ✕ **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

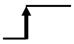
These instructions compare the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
 (s1) (s2) (s3) (s4) (d)	ENO:=TCMP(EN,s1,s2,s3,s4,d); ENO:=TCMPP(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TCMP	
TCMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device for storing the time data (hour) to be compared	0 to 23	16-bit signed binary	ANY16
(s2)	Start device for storing the time data (minute) to be compared	0 to 59	16-bit signed binary	ANY16
(s3)	Start device for storing the time data (second) to be compared	0 to 59	16-bit signed binary	ANY16
(s4)	Start device for storing the time data (hour, minute, second) to be compared	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the comparison result	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

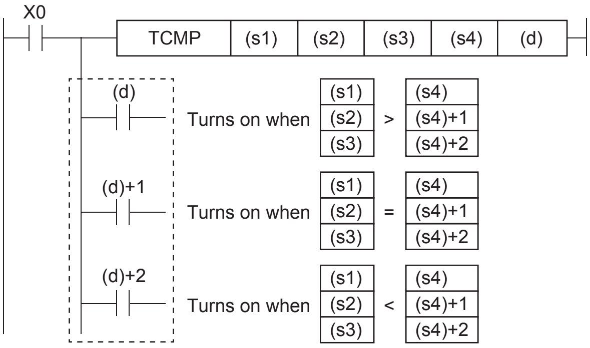
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s3)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.



- (s1): hour, (s2): minute, (s3): second
- (s4): hour, (s4)+1: minute, (s4)+2: second

Operation error

Error code (SD0)	Description
3405H	The value specified by (s1) and (s4) is outside the following range. 0 to 23
	The value specified by (s2), (s3), (s4)+1, and (s4)+2 is outside the following range. 0 to 59

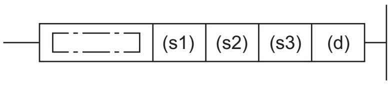
Outputting a band comparison result of time data

TZCP(P)

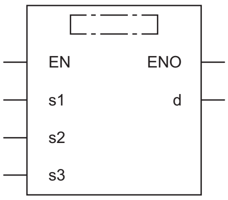
RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support these instructions. (Use an engineering tool with version "1.020W" or later.)

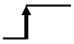
These instructions compare the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=TZCP(EN,s1,s2,s3,d); ENO:=TZCPP(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TZCP	
TZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device for storing the lower limit value of the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device for storing the upper limit value of the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s3)	Start device for storing the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

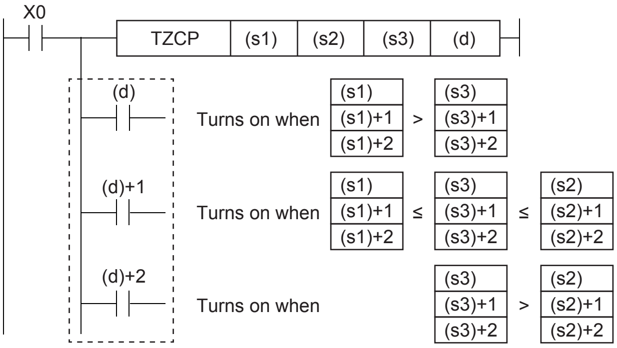
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○[1]	—	—	—	—	○	—	—	—	—

[1] T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Device	Clock data	Data range
(s1), (s2), (s3)	Hour	0 to 23
(s1)+1, (s2)+1, (s3)+1	Minute	0 to 59
(s1)+2, (s2)+2, (s3)+2	Second	0 to 59

Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

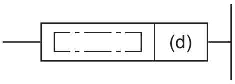
Error code (SD0)	Description
3405H	The value specified by (s1), (s2), or (s3) is outside the following range. 0 to 23
	The value specified by (s1)+1, (s2)+1, (s3)+1, (s1)+2, (s2)+2, or (s3)+2 is outside the following range. 0 to 59

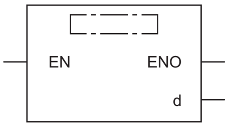
Reading expansion clock data

S(P).DATERD

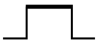
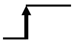
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions read clock data including millisecond from the clock elements in the CPU module.

Ladder	ST
	ENO:=S_DATERD(EN,d); ENO:=SP_DATERD(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
S.DATERD	
SP.DATERD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the clock data that has been read	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 8)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read "year, month, day, hour, minute, second, day of week, and millisecond" from the clock element of the CPU module, and store the read data in binary in the device specified by (d) and later.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6	(d)+7
(Clock element)	**Year**	**Month**	**Day**	**Hour**	**Minute**	**Second**	**Day of week**	**Millisecond**
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6	0 to 999

- "Year" stored in the device specified by (d) is a 4-digit year.
- "Day of week" stored in the device specified by (d)+6 is a number from 0 to 6 corresponding to Sunday to Saturday.

Day of week	Day	Month	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

- Data is automatically corrected in leap years.

Precautions

- These instructions read clock data and store it in the device even when incorrect click data is set in the CPU module. (Example: February 30) When setting clock data with the DATEWR(P) instruction or engineering tool, be careful not to set incorrect clock data.
- When millisecond clock data is read, the maximum error is 2ms. (This error means the difference between the data stored in clock elements in the CPU module and the data read by the S(P).DATERD instruction.)

Operation error

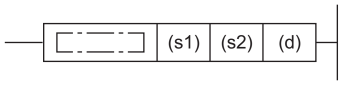
There is no operation error.

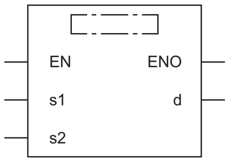
Adding expansion clock data

S(P).DATE+

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These instructions add time data.

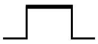
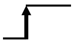
Ladder	ST[1]
	ENO:=S_DATEPLUS(EN,s1,s2,d); ENO:=SP_DATEPLUS(EN,s1,s2,d);

FBD/LD


(□ is replaced by either of the following: S_DATEPLUS, SP_DATEPLUS.)

[1] The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
S.DATE+	
SP.DATE+	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the augend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Start device where the addend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(d)	Start device for storing the addition result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions add the time data in the device specified by (s2) to the time data in the device specified by (s1), and store the addition result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)		(s2)	hour	(0 to 23)		(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)		(s2)+1	minute	(0 to 59)		(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)	+	(s2)+2	second	(0 to 59)	⇨	(d)+2	second	(0 to 59)
(s1)+3	–			(s2)+3	–			(d)+3	–	
(s1)+4	1/1000second	(0 to 999)		(s2)+4	1/1000second	(0 to 999)		(d)+4	1/1000second	(0 to 999)

Ex.

7:48:10:500 is added to 6:32:40:875.

(s1)	6		(s2)	7		(d)	14
(s1)+1	32		(s2)+1	48		(d)+1	20
(s1)+2	40	+	(s2)+2	10	⇨	(d)+2	51
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

- If the time obtained as the result of addition exceeds 24 hours, 24 hours are subtracted from the resultant time to produce the operation result. For example, when 20:20:20:500 is added to 14:20:30:875, the operation result is 10:40:51:375 rather than 34:40:51:375.

(s1)	14		(s2)	20		(d)	10
(s1)+1	20		(s2)+1	20		(d)+1	40
(s1)+2	30	+	(s2)+2	20	⇨	(d)+2	51
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

Point

- Devices (s1)+3, (s2)+3, and (d)+3 are not used for operation.
- The clock data that has been read by the S(P).DATERD instruction can be added without conversion.

(d)	Hour	
(d)+1	Minute	
(d)+2	Second	
(d)+3	Day of week	⬅
(d)+4	Millisecond	

When clock data is read by the S(P).DATERD instruction, "day of week" is inserted between "second" and "millisecond".

If the S(P)DATE+ instruction is used to read clock data, the data can be directly used for addition since it does not perform calculation for the day of week.

Operation error

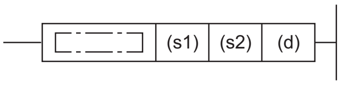
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Subtracting expansion clock data

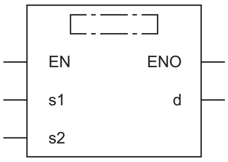
S(P).DATE-

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions subtract time data.

Ladder	ST[*1]
	ENO:=S_DATEMINUS(EN,s1,s2,d); ENO:=SP_DATEMINUS(EN,s1,s2,d);

FBD/LD

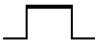
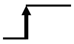


(□ is replaced by either of the following: S_DATEMINUS, SP_DATEMINUS.)

---

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
S.DATE-	
SP.DATE-	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where minuend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Start device where the subtrahend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(d)	Start device for storing the subtraction result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions subtract the time data in the device specified by (s2) from the time data in the device specified by (s1), and store the subtraction result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)		(s2)	hour	(0 to 23)		(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)		(s2)+1	minute	(0 to 59)		(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)	-	(s2)+2	second	(0 to 59)	⇒	(d)+2	second	(0 to 59)
(s1)+3	–			(s2)+3	–			(d)+3	–	
(s1)+4	1/1000second	(0 to 999)		(s2)+4	1/1000second	(0 to 999)		(d)+4	1/1000second	(0 to 999)

Ex.

3:50:10:500 is subtracted from 10:40:20:875.

(s1)	10		(s2)	3		(d)	6
(s1)+1	40		(s2)+1	50		(d)+1	50
(s1)+2	20	-	(s2)+2	10	⇒	(d)+2	10
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

- If the time obtained as the result of subtraction becomes a negative value, 24 hours are added to the resultant time to produce the operation result. For example, when 10:42:12:500 is subtracted from 4:50:32:875, the operation result is 18:8:20:375 rather than -6:8:20:375.

(s1)	4		(s2)	10		(d)	18
(s1)+1	50		(s2)+1	42		(d)+1	8
(s1)+2	32	-	(s2)+2	12	⇒	(d)+2	20
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

Point

- Devices (s1)+3, (s2)+3, and (d)+3 are not used for operation.
- The clock data that has been read by the S(P).DATERD instruction can be subtracted without conversion.

(d)	Hour	
(d)+1	Minute	
(d)+2	Second	
(d)+3	Day of week	⬅
(d)+4	Millisecond	

When clock data is read by the S(P).DATERD instruction, "day of week" is inserted between "second" and "millisecond".

If the S(P).DATE- instruction is used to read clock data, the data can be directly used for subtraction since it does not perform calculation for the day of week.

Operation error

Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

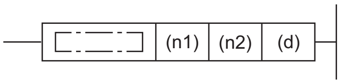
7.26 Timing Check Instructions

Generating timing pulses

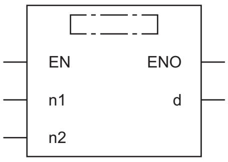
DUTY

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

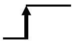
This instruction turns on the user timing clock for the specified number of scans and off for the specified number of scans.

Ladder	ST
	ENO:=DUTY(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DUTY	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(n1)	Number of scans during which the clock is turned on	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of scans during which the clock is turned off	0 to 65535	16-bit unsigned binary	ANY16
(d)	Special relay device number of user timing clock to be operated	SM420 to SM424	Bit	ANY_BOOL[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Only labels assigned to SM420 to SM424 can be used.

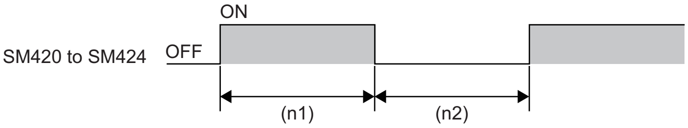
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○[1]	—	—	—	—	—	—	○	—	—	—	—

[1] Only SM420 to SM424 can be used.

Processing details

- This instruction turns on SM420 to SM424 in the device specified by (d) for the number of scans specified by (n1) and turns it off for the number of scans specified by (2).



(n1): (n1) scans

(n2): (n2) scans

- The scan execution type program uses SM420 to SM424.
- When 0 is specified in (n1) and (n2) is equal to or greater than 0, SM420 to SM424 stay off. When (n1) is greater than 0 and (n2) is 0, SM420 to SM424 stay on.
- When the DUTY instruction is executed, the data specified by (n1), (n2), and (d) is stored in the system, and the timing pulses are turned on or off by the END processing.

Operation error

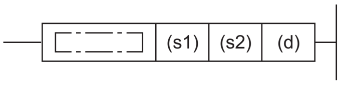
Error code (SD0)	Description
2820H	The device other than SM420 to SM424 is specified by (d).

Measuring time of the specified data

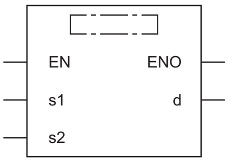
TIMCHK

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✕

This instruction measures the on time of the device and, if the on time has continued as specified or longer, turns on the specified device.

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

FBD/LD



■Execution condition

Instruction	Execution condition
TIMCHK	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Device for storing the current value measured (unit: 100ms)	—	16-bit signed binary	ANY16
(s2)	Set value for measurement or the device where the set value for measurement is stored (unit: 100ms)	0 to 32767	16-bit signed binary	ANY16
(d)	Device to be turned on at time-up	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction measures the on time of the device specified by (s1) and, if the on time has continued as specified in the device specified by (s2) or longer, turns on the device specified by (d).
- The current value in the device specified by (s1) is cleared to 0 and the device specified by (d) is turned off on the rising edge of the execution command. The current value in the device specified by (s1) and the on state of the device specified by (d) are retained even after the execution command turns off.
- The current value measured is stored in units of 100ms. Set the measurement time in increments of 100ms.
- If 0 is specified in (s2), the current value in the device specified by (s1) is cleared to 0 and the device specified by (d) is turned off on the rising edge of the execution command.
- If a number other than 0 to 32767 is specified in (s2), (d) is turned on at the next scan after the execution command turns on.

Operation error

There is no operation error.

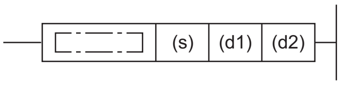
Hour meter

HOURM

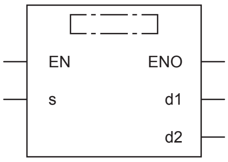
RnCPU **RnENCPU** ✗ **RnPCPU (Process)** ✗ **RnPCPU (Redundant)** ✗ **RnSFCPU (Standard)** ✗ **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).

Ladder	ST
 (s) (d1) (d2)	ENO:=HOURM(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
HOURM	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Time after which the alarm (d2) is set to on (unit: hour)	0 to 32767	16-bit signed binary	ANY16
(d1)	Device where the measured current value is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(d2)	Device to be turned on when a timeout occurs (alarm output)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[*1]	○	○	○	○	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○[*2]	—	—	—	—	—	—	—	—	—

[*1] FX and FY cannot be used.

[*2] T, ST, C, and FD cannot be used.

Processing details

- This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).
- In (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1).
- The measured current value of less than one hour (in units of second) is stored in (d1)+1.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When (d1) reaches the maximum value (32767) and also (d1)+1 reached reaches the maximum value (3599), the measurement is stopped.
- This instruction operates even if a negative value is set in the device specified by (d1).
- Changing clock data (including time adjustment by the daylight-saving time function) does not affect the operation of the HOURM instruction.

Precautions

- In cases such as measuring the ON time from initial value or continuing to the measurement even after the current value reaches the maximum value of 16 bits, clear (d1) to (d1)+1 if (d1) is specified with the device, or clear two elements if (d1) is specified with the label.
- To avoid that timer measurement does not work normally, do not use this instruction in the initial execution type program, interrupt program, fixed scan execution type program, and event execution type program.

Operation error

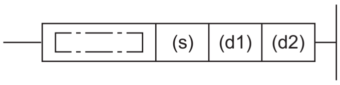
Error code (SD0)	Description
3405H	The range specified by (s) is outside the following range. 0 to 32767

DHOURM

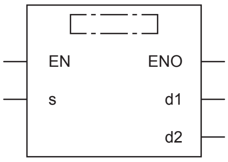
RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).

Ladder	ST
	ENO:=DHOURM(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DHOURM	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Time after which the alarm (d2) is set to on (unit: hour)	0 to 2147483647	32-bit signed binary	ANY32
(d1)	Device where the measured current value is stored	—	32-bit signed binary	ANY32_ARRAY (Number of elements: 2)
(d2)	Device to be turned on when a timeout occurs (alarm output)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, C, and FD cannot be used.

Processing details

- This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).
- In (s)+1 and (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1)+1 and (d1). ((d1)+1: upper value, (d1): lower value)
- The measured current value of less than one hour (in units of second) is stored in (d1)+2.
- No value is stored in (d1)+3.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When (d1)+1 and (d1) reaches the maximum value (2147483647) and also (d1)+2 reached reaches the maximum value (3599), the measurement is stopped.
- This instruction operates even if a negative value is set in the device specified by (d1).
- Changing clock data (including time adjustment by the daylight-saving time function) does not affect the operation of the DHOURM instruction.

Precautions

- In cases such as measuring the ON time from initial value or continuing to the measurement even after the current value reaches the maximum value of 32 bits, clear (d1) to (d1)+2 if (d1) is specified with the device, or clear two elements if (d1) is specified with the label.
- To avoid that timer measurement does not work normally, do not use this instruction in the initial execution type program, interrupt program, fixed scan execution type program, and event execution type program.

Operation error

Error code (SD0)	Description
3405H	The range specified by (s) is outside the following range. 0 to 2147483647

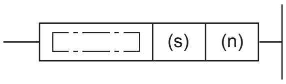
7.27 Module Access Instructions

Performing I/O refresh

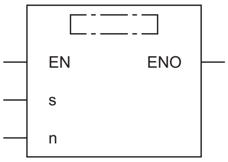
RFS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕RnSFCPU (Safety)

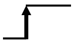
These instructions refresh the n points of data from the specified device, and import external inputs or outputs data to the output module.

Ladder	ST
	ENO:=RFS(EN,s,n); ENO:=RFSP(EN,s,n);

FBD/LD



■Execution condition

Instruction	Execution condition
RFS	
RFSP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device to be refreshed	—	Bit	ANY_BOOL[1]
(n)	Number of refreshed points	1 to 4096	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Only labels assigned to devices (X, Y) can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○[1]	—	—	—	—	—	—	—	—	—	—	—
(n)	—	○	○	○	○	—	—	○	○	—	—	—

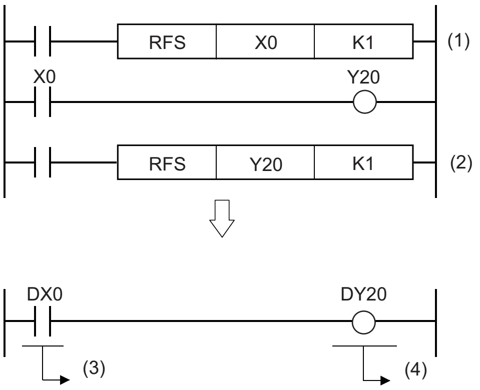
[1] Only X and Y can be used.

Processing details

- This instruction refreshes only the relevant device during one scan, and imports external inputs or outputs data to the output module.
- The instruction imports external inputs and outputs data to the outside altogether only after execution of the END instruction of the program, and therefore cannot output pulse signals to the outside during one scan. When executed, the I/O refresh instruction forcibly refreshes the relevant input (X) and output (Y) during program execution and therefore can output pulse signals to the outside during one scan.
- To refresh the input (X) or output (Y) in units of points, use the direct access input (DX) or direct access output (DY).

Ex.

When a program using the RFS instruction is changed to a program using direct access input/output



(1)Refresh X0.
(2)Refresh Y20.
(3)Direct access input
(4)Direct access output

Operation error

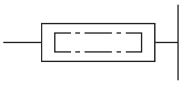
Error code (SD0)	Description
2820H	The range of (n) points from the device specified by (s) exceeds the range of the proximal I/O.

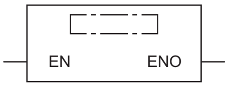
Selecting refresh to be performed

COM(P)

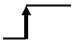
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions perform I/O refresh, link refresh of the network module, and device/label access service processing.

Ladder	ST
	ENO:=COM(EN); ENO:=COMP(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
COM	
COMP	

Processing details

- The COM(P) instructions are used to perform processing such as I/O refresh at any time during execution of the sequence program.
- The processing performed by the COM(P) instruction includes the following.
  - I/O refresh
  - Link refresh of the CC-Link module
  - Link refresh of the CC-Link IE Controller Network module
  - Link refresh of the CC-Link IE Field Network module
  - Link refresh of CC-Link IE Field Network Basic[*1]
  - Link refresh of the MELSECNET/H module
  - Intelligent function module refresh
  - Refresh of multiple CPU system using the CPU buffer memory (in END processing)
  - Import of input/output outside the group of multiple CPU system
  - Device/label access service processing (communication with the engineering tool, GOT, or other external devices)

[*1] Programmable controller CPU with firmware version "25" or later supports this processing.

- When SM775 is turned off, every processing except I/O refresh is performed.

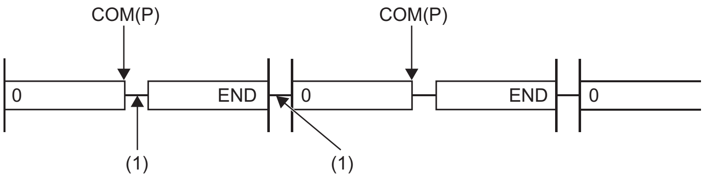
Description	SM775 is off	SM775 is on
I/O refresh and import of input/output outside the group of multiple CPU system	Non-execution	Execution or non-execution can be selected.
Link refresh of the CC-Link module	Execution	
Link refresh of the CC-Link IE Controller Network module		
Link refresh of the CC-Link IE Field Network module		
Link refresh of CC-Link IE Field Network Basic		
Link refresh of the MELSECNET/H module		
Intelligent function module refresh		
Refresh of multiple CPU system using the CPU buffer memory (in END processing)		
Device/label access service processing (communication with the engineering tool, GOT, or other external devices)		

- Select execution or non-execution for b0 to b4, b6, b13, and b15 of SD775 (Selection of refresh processing during the COM instruction execution), and then turn on SM775.

	b15	b14	b13	b12···b7	b6	b5	b4	b3	b2	b1	b0
SD775	1/0	0	1/0	0	1/0	0	1/0	1/0	1/0	1/0	1/0
	(8)		(7)		(6)		(5)	(4)	(3)	(2)	(1)

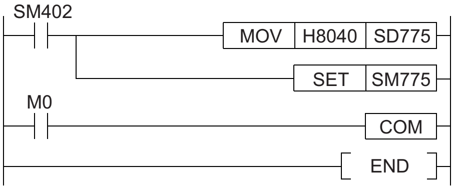
(1) I/O refresh, import of input/output of non-controlled modules in a multiple CPU system
(2) Link refresh of the CC-Link module
(3) Link refresh of the CC-Link IE Controller Network module and MELSECNET/H module
(4) Intelligent function module refresh
(5) Refresh of multiple CPU system using the CPU buffer memory (in END processing)
(6) Link refresh of the CC-Link IE Field Network module
(7) Link refresh of CC-Link IE Field Network Basic
(8) Device/label access service processing (communications with the engineering tool, GOT, or other external devices)

- When executed, the COM(P) instruction performs the specified refresh processing.



(1) Specified processing

- In the following program example, link refresh of the CC-Link IE Field Network module is executed when M0 turns on.



Precautions

- The COM(P) instruction can be used as many times as needed in the program. Note, however, that the scan time of the program is extended by the time of the processing selected by SD775.
- Interrupts are enabled during execution of the COM(P) instruction. If refresh data is used by an interrupt program, data inconsistency may occur.
- If device/label access service processing is performed by the COM(P) instruction while an Ethernet device is connected to the Ethernet port, the processing time of the instruction may be extended.

Point

The COM(P) instruction cannot be used in the interrupt program.

Operation error

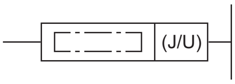
There is no operation error.

Performing module refresh

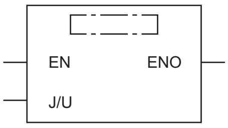
S(P).ZCOM

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

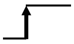
These instructions perform refresh processing for the specified module.

Ladder	ST
	ENO:=S_ZCOM(EN,J); ENO:=S_ZCOM(EN,U); ENO:=SP_ZCOM(EN,J); ENO:=SP_ZCOM(EN,U);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ZCOM	
SP.ZCOM	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(J)	Own station network number	1 to 239	Device name	ANY16
(U)	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	0H to FFH		
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

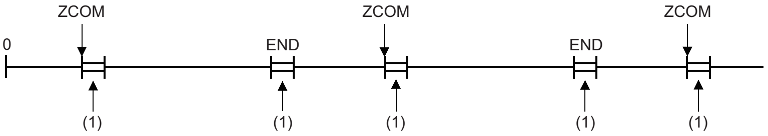
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J/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	$	
(J/U)	—	—	—	—	—	—	—	—	—	—	—	○

- The S(P).ZCOM instructions are used to perform refresh at any time during execution of the sequence program. The following lists the targets of refresh by the S(P).ZCOM instructions.
  - Link refresh of the CC-Link IE Controller Network module (in refresh setting)
  - Link refresh of the CC-Link IE Field Network module (in refresh setting)
  - Link refresh of the MELSECNET/H module (in refresh setting)
  - Link refresh of the CC-Link module (in refresh setting)
  - Intelligent function module refresh (when a refresh parameter is specified)

Processing details

- When executed, the S(P).ZCOM instruction temporarily stops sequence program processing by the CPU module and perform refresh processing for the module specified by (J/U).

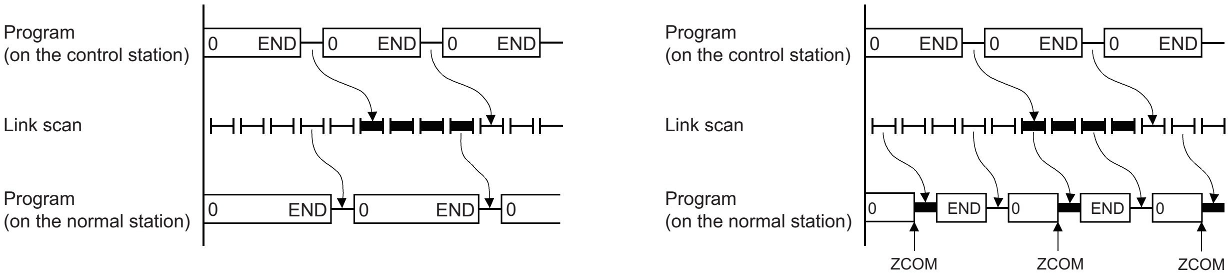


(1) Refresh processing

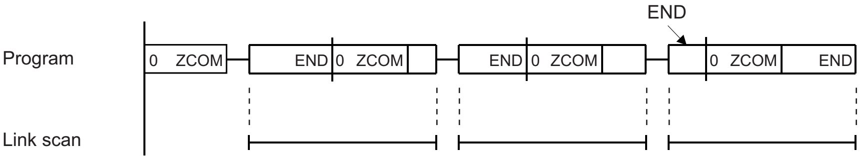
- The following is applicable when refresh processing of the CC-Link IE Controller Network module or MELSECNET/H module (network between programmable controllers) is performed.
  - When the scan time of the sequence program of the host station is longer than that of another station, the S(P).ZCOM instruction is used to ensure the import of data from the other station.

[When the S(P).ZCOM instruction is not used]

[When the S(P).ZCOM instruction is used]



  - When the link scan time is longer than the sequence program scan time, using the S(P).ZCOM instruction will not make data communication faster.



Precautions

- The S(P).ZCOM instruction can be used as many times as needed in the program. Note, however, that the scan time of the program is extended by the refresh time.
- Interrupts are enabled during execution of the S(P).ZCOM instruction. If refresh data is used by an interrupt program, data separation may occur.

Operation error

Error code (SD0)	Description
2800H	The specified start I/O number is out of the range, 0 to FFH.
2801H	No module exists at the position specified by the start I/O number.
2804H	The network number set to (J) is out of the range, 1 to 239.
2820H	The specified network number is not connected to the host station.

Point

- The S(P).ZCOM instruction cannot be used in the interrupt program.
- To communicate only with external devices, use the COM(P) instruction.

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

FROM(P), DFROM(P)

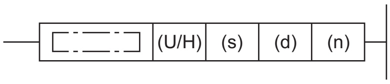
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- FROM(P):

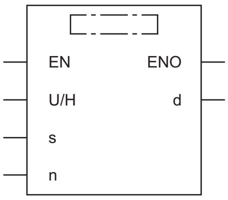
These instructions read n words of data from the buffer memory address in the specified module or another CPU module.

- DFROM(P):

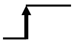
These instructions read n×2 words of data from the buffer memory address in the specified module or another CPU module.

Ladder	ST
	ENO:=FROM(EN,U/H,s,n,d); ENO:=FROMP(EN,U/H,s,n,d); ENO:=DFROM(EN,U/H,s,n,d); ENO:=DFROMP(EN,U/H,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FROM DFROM	
FROMP DFROMP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module or CPU module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(s)		Start address of buffer memory or CPU built-in memory containing the data to be read	0 to 65535	16-bit unsigned binary	ANY16
(d)	FROM(P)	Start device for storing the data that has been read	—	16-bit signed binary	ANY16[1]
	DFROM(P)			32-bit signed binary	ANY32[1]
(n)		Number of read data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

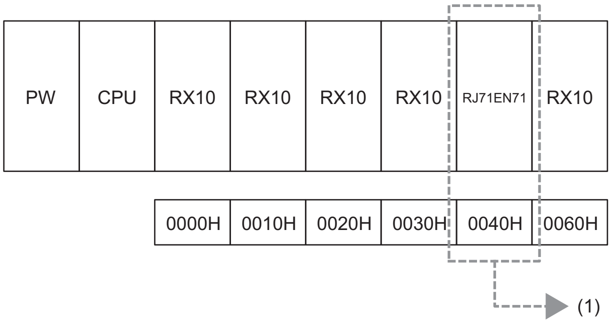
■Applicable devices

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	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	○*1	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only the DFROM(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



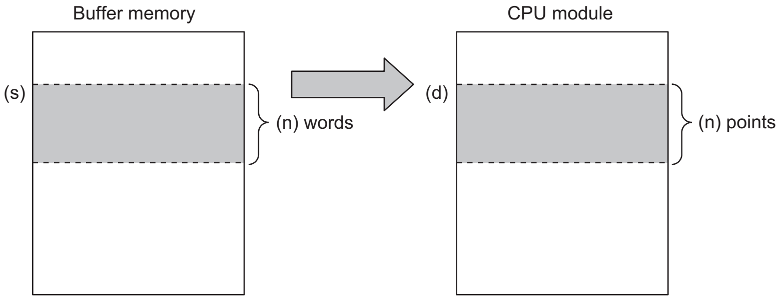
(1) Specify K4 or H4 as the start I/O number of the read-target module.

To read the module name of a CPU module, specify the read start I/O number as in the following table.

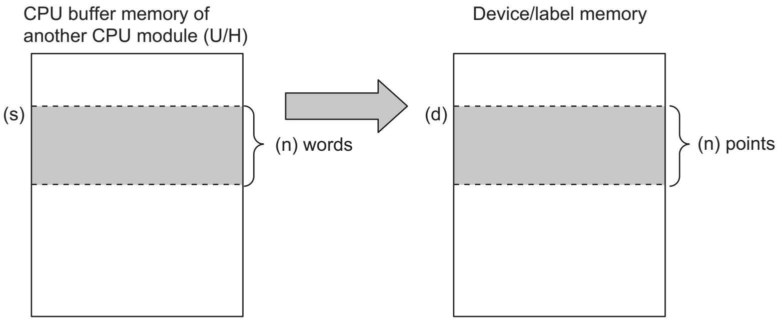
CPU module	Read start I/O number
CPU No. 1	3E0H
CPU No. 2	3E1H
CPU No. 3	3E2H
CPU No. 4	3E3H

■FROM(P)

- These instructions read (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.
  - Reading word data from module



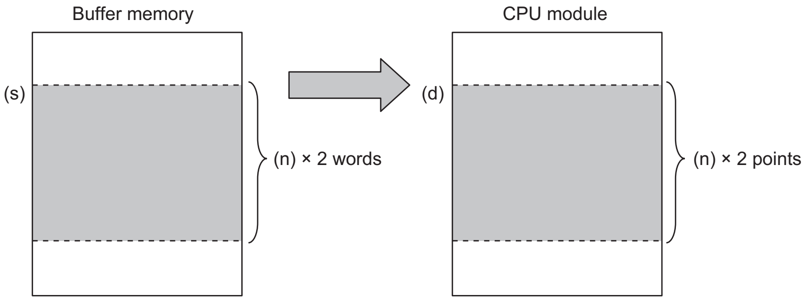
  - Reading word data from another CPU module



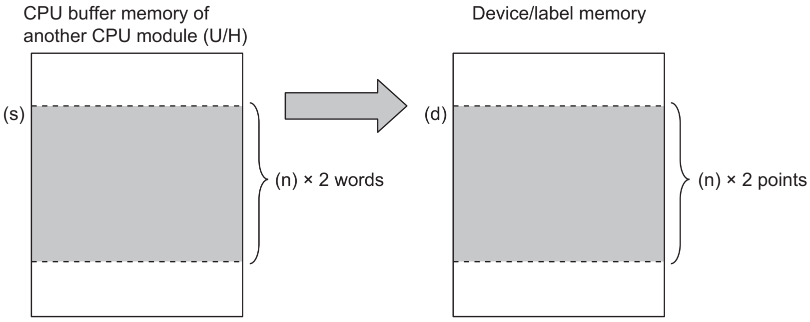
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

■DFROM(P)

- These instructions read (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.
  - Reading double word data from module



  - Reading double word data from another CPU module



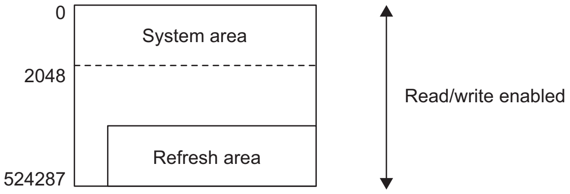
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s) is outside the range of buffer memory or CPU buffer memory.
	The (n) points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (FROM(P) instruction)
	The (n)×2 points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (DFROM(P) instruction)

Point

- Module data can also be read using the module access device. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later.



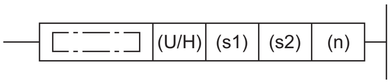
- A CPU buffer memory access device can be used to read data from the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The FROM(P) and DFROM(P) instructions can read data from the buffer memory address with a capacity of 64K or less. To read data from the buffer memory address with a capacity exceeding 64K, use the FROMD(P) or DFROMD(P) instruction. (☞ Page 1054 FROMD(P), DFROMD(P))

Writing 1-word/2-word data to a module (16-bit specification)

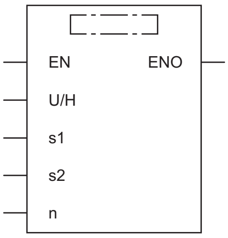
TO(P), DTO(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

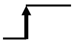
- TO(P): These instructions write the n points of data from the specified device to the buffer memory in the module/host CPU module.
- DTO(P): These instructions write the n×2 points of data from the specified device to the buffer memory in the module/host CPU module.

Ladder	ST
	ENO:=TO(EN,U/H,s1,s2,n); ENO:=TOP(EN,U/H,s1,s2,n); ENO:=DTO(EN,U/H,s1,s2,n); ENO:=DTOP(EN,U/H,s1,s2,n);

FBD/LD



■Execution condition

Instruction	Execution condition
TO DTO	
TOP DTOP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module or CPU module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)		Start address of buffer memory or CPU built-in memory to which data is to be written	0 to 65535	16-bit unsigned binary	ANY16
(s2)	TO(P)	Write data or the start device where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16[1]
	DTO(P)		-2147483648 to 2147483647	32-bit signed binary	ANY32[1]
(n)		Number of write data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

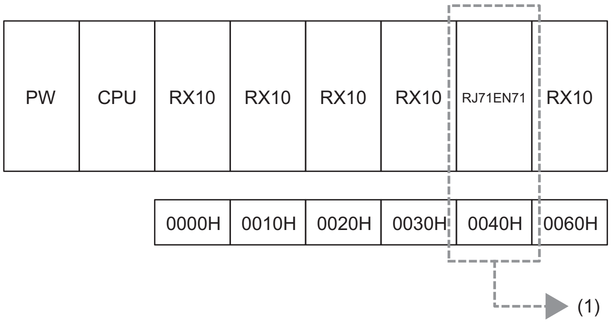
■Applicable devices

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	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	—	○	—	—	○[1]	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

[1] Only the DTO(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



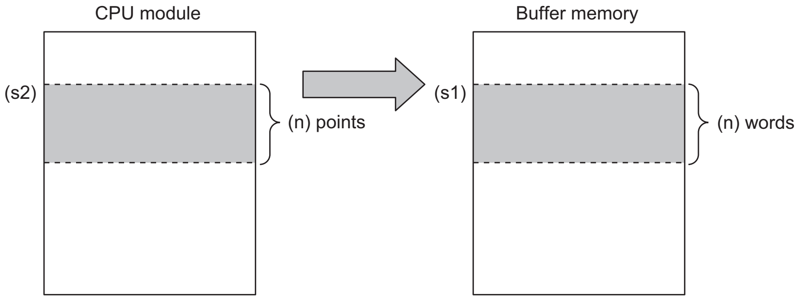
(1) Specify K4 or H4 as the start I/O number of the write-target module.

To read the module name of a CPU module, specify the read start I/O number as in the following table.

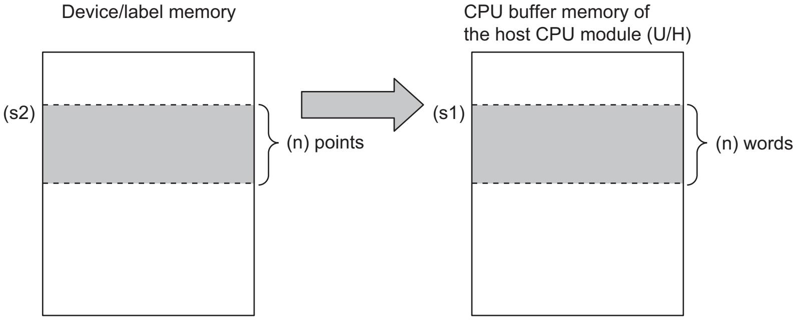
CPU module	Read start I/O number
CPU No. 1	3E0H
CPU No. 2	3E1H
CPU No. 3	3E2H
CPU No. 4	3E3H

■TO(P)

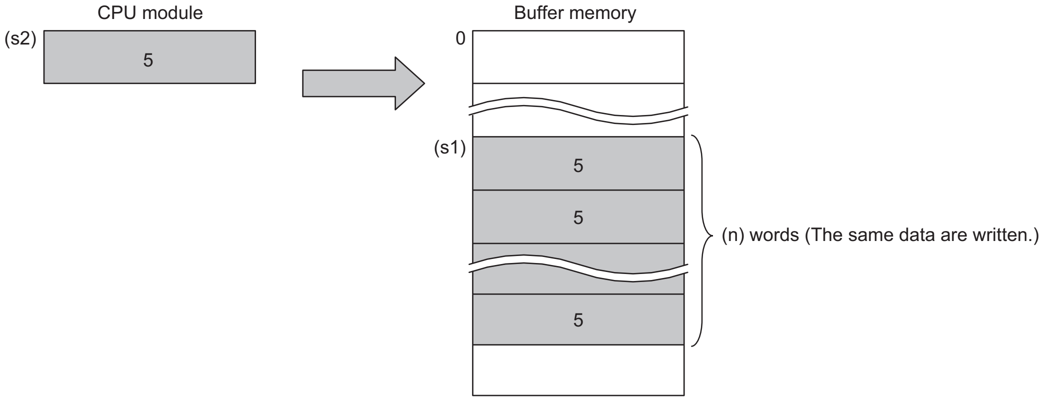
- These instructions write the n points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host CPU module specified by (U/H).
  - Writing word data to a module



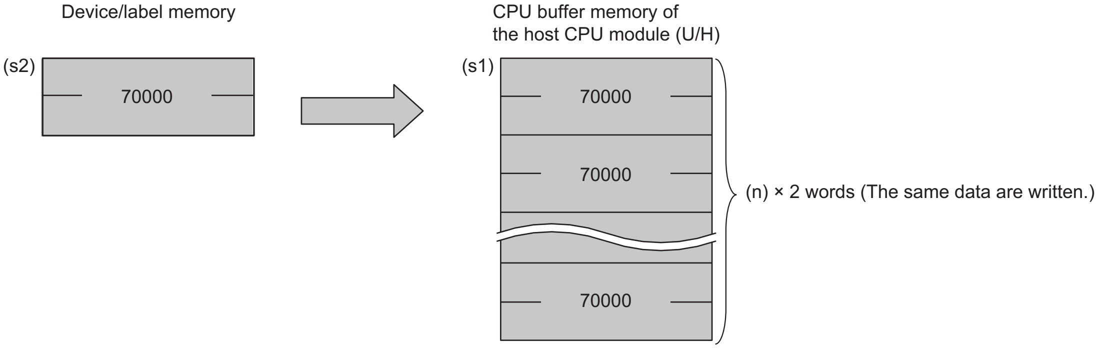
  - Writing word data to the host CPU module



- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.
  - Writing word data to a module



  - Writing word data to the host CPU module

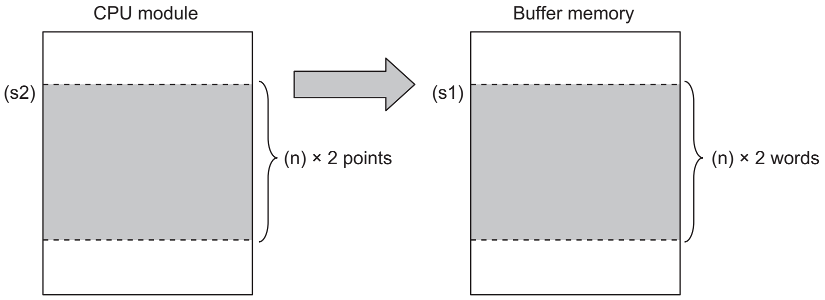


- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

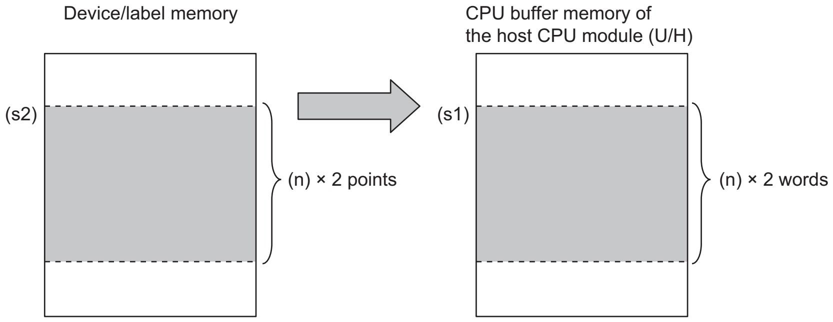
■DTO(P)

- These instructions write the (n)×2 points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host CPU module specified by (U/H).

• Writing double word data to a module

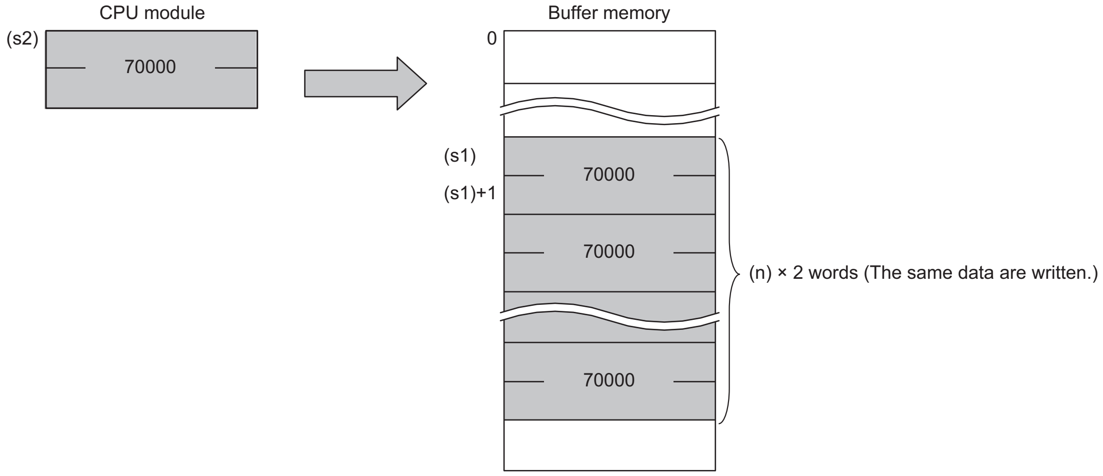


• Writing double word data to the host CPU module

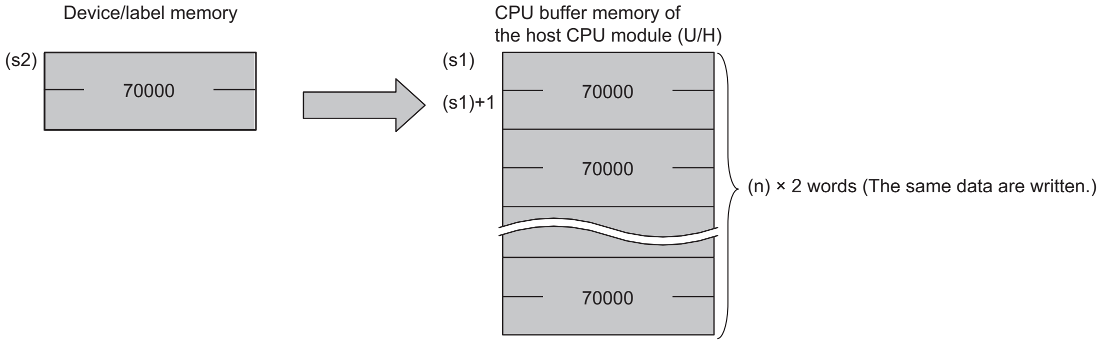


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n)×2 words from the specified buffer memory address.

• Writing double word data to a module



• Writing double word data to the host CPU module



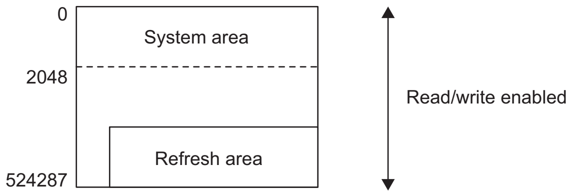
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s1) is outside the range of buffer memory or CPU buffer memory.
	The (n) points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (TO(P) instruction)
	The (n)×2 points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (DTO(P) instruction)

Point

- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later.



- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The TO(P) and DTO(P) instructions can write data to the buffer memory address with a capacity of 64K or less. To write data to the buffer memory address with a capacity exceeding 64K, use the TOD(P) or DTOD(P) instruction. (☞ Page 1058 TOD(P), DTOD(P))

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

FROMD(P), DFROMD(P)

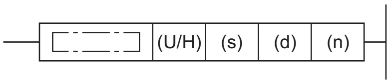
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

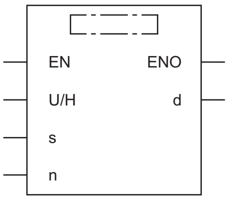
- FROMD(P):

These instructions read n words of data from the buffer memory address in the specified module or another CPU module.

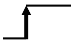
- DFROMD(P):

These instructions read n×2 words of data from the buffer memory address in the specified module or another CPU module.

Ladder	ST
 (U/H) (s) (d) (n)	ENO:=FROMD(EN,U/H,s,n,d); ENO:=FROMDP(EN,U/H,s,n,d); ENO:=DFROMD(EN,U/H,s,n,d); ENO:=DFROMDP(EN,U/H,s,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
FROMD DFROMD	
FROMDP DFROMDP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module or CPU module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(s)		Buffer memory from which the data is read or the start device where the start address of the CPU memory is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	FROMD(P)	Start device for storing the data that has been read	—	16-bit signed binary	ANY16[*1]
	DFROMD(P)			32-bit signed binary	ANY32[*1]
(n)		Number of read data	0 to 4294967295	32-bit unsigned binary	ANY32
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

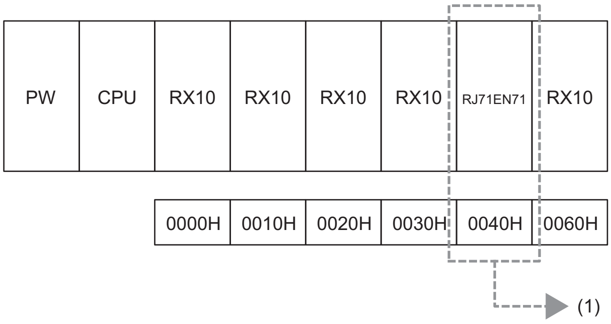
■Applicable devices

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	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	—	○	—	—	○*1	—	○	○	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

*1 Only the DFROMD(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the read-target module.

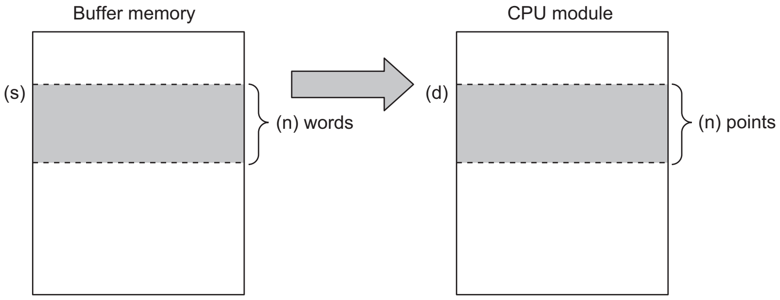
To read the module name of a CPU module, specify the read start I/O number as in the following table.

CPU module	Read start I/O number
CPU No. 1	3E0H
CPU No. 2	3E1H
CPU No. 3	3E2H
CPU No. 4	3E3H

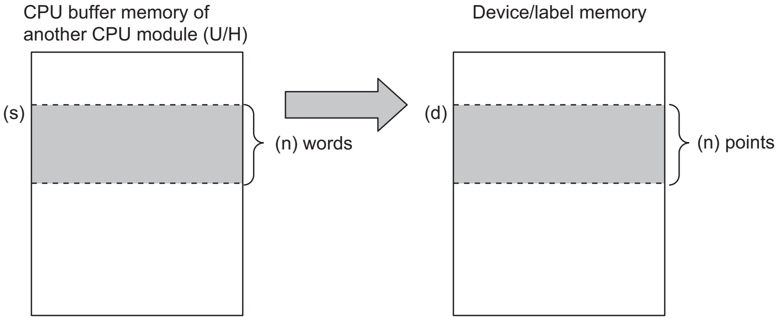
■FROMD(P)

- These instructions read (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading word data from module



• Reading word data from another CPU module

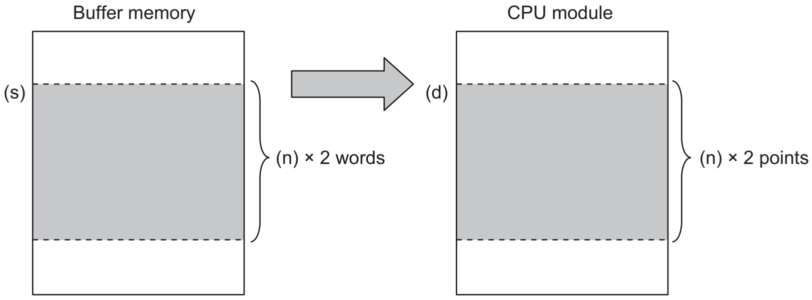


- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

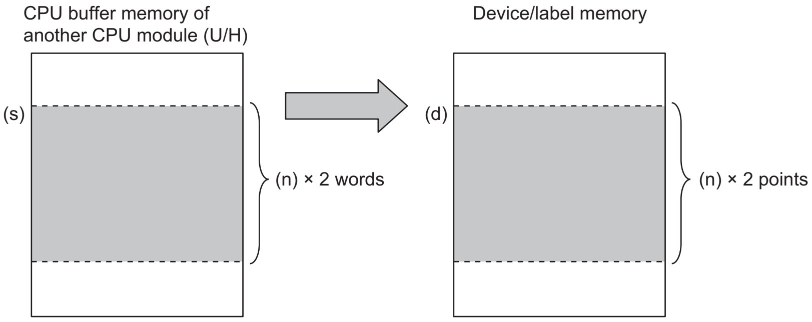
■DFROMD(P)

- These instructions read (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading double word data from module



• Reading double word data from another CPU module



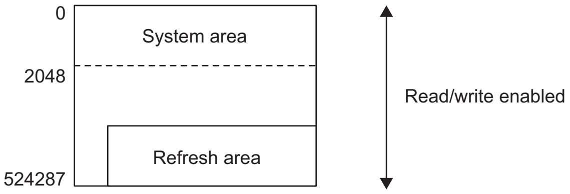
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s) is outside the range of buffer memory or CPU buffer memory.
	The (n) points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (FROMD(P) instruction)
	The (n)×2 points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (DFROMD(P) instruction)

Point

- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later.



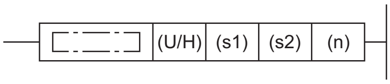
- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The FROMD(P) and DFROMD(P) instructions can read data from the buffer memory address with a capacity exceeding 64K.

Writing 1-word/2-word data to a module (32-bit specification)

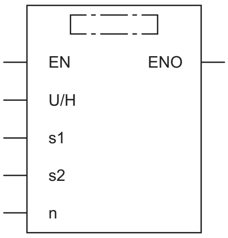
TOD(P), DTOD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

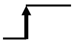
- TOD(P): These instructions write the n points of data from the specified device to the buffer memory in the module/host CPU module.
- DTOD(P): These instructions write the n×2 points of data from the specified device to the buffer memory in the module/host CPU module.

Ladder	ST
	ENO:=TOD(EN,U/H,s1,s2,n); ENO:=TODP(EN,U/H,s1,s2,n); ENO:=DTOD(EN,U/H,s1,s2,n); ENO:=DTODP(EN,U/H,s1,s2,n);

FBD/LD



■Execution condition

Instruction	Execution condition
TOD DTOD	
TODP DTODP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module or CPU module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)		Buffer memory to which the data is written or the start device where the start address of CPU memory is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(s2)	TOD(P)	Write data or the start device where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16[1]
	DTOD(P)		-2147483648 to 2147483647	32-bit signed binary	ANY32[1]
(n)		Number of write data	0 to 4294967295	32-bit unsigned binary	ANY32
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

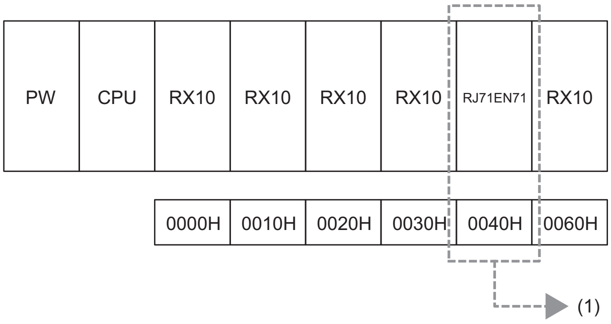
■Applicable devices

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	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	—	○	—	—	○[*1]	—	○	○	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

*1 Only the DTOD(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



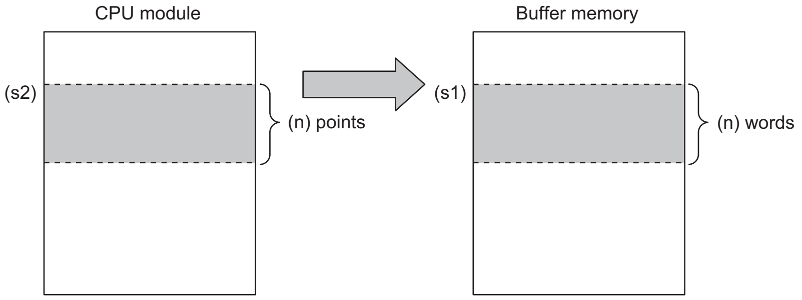
(1) Specify K4 or H4 as the start I/O number of the write-target module.

To read the module name of a CPU module, specify the read start I/O number as in the following table.

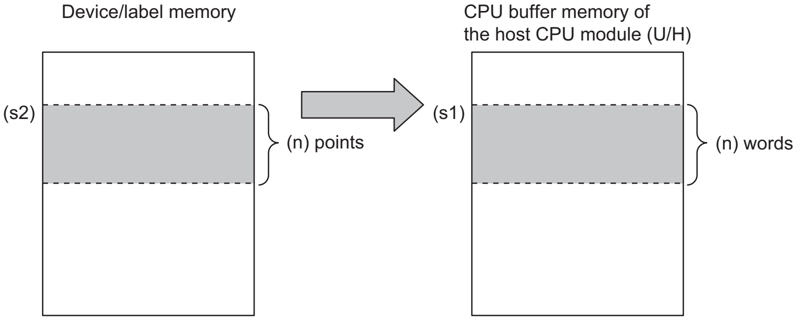
CPU module	Read start I/O number
CPU No. 1	3E0H
CPU No. 2	3E1H
CPU No. 3	3E2H
CPU No. 4	3E3H

■TOD(P)

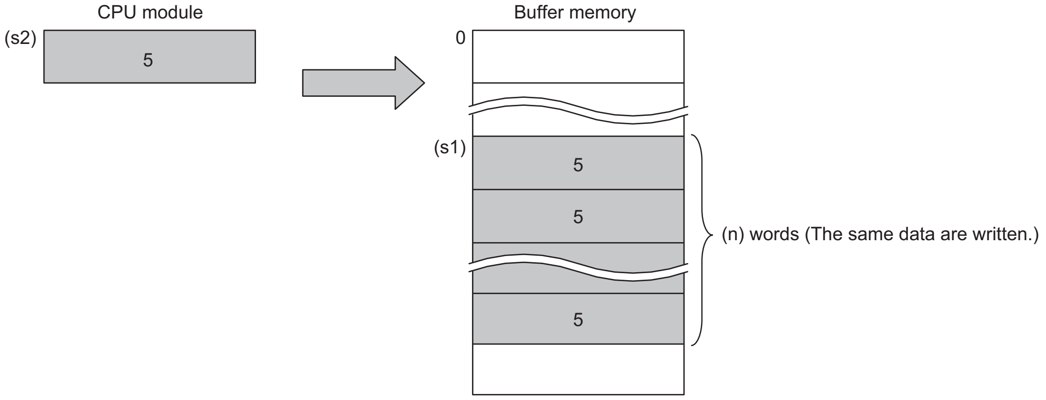
- These instructions write the n points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host CPU module specified by (U/H).
  - Writing word data to a module



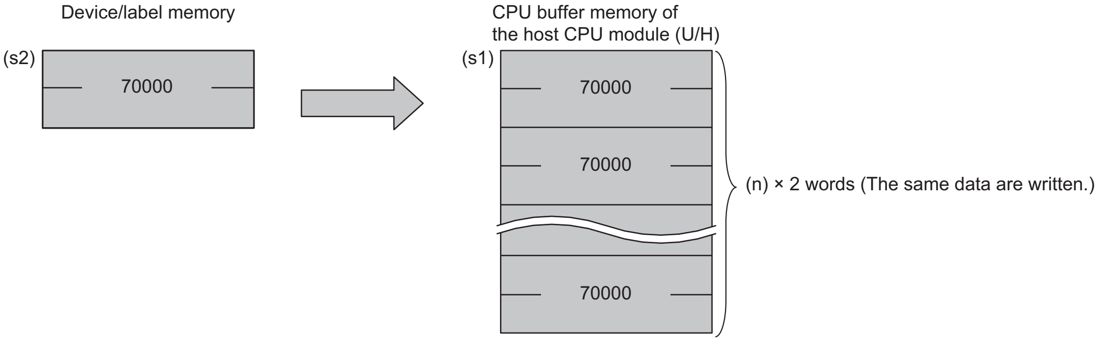
  - Writing word data to the host CPU module



- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.
  - Writing word data to a module



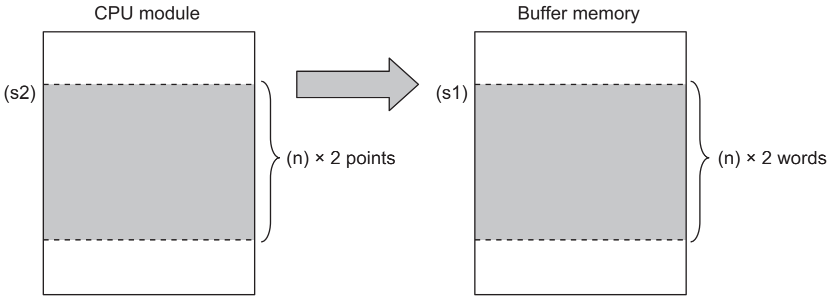
  - Writing word data to the host CPU module



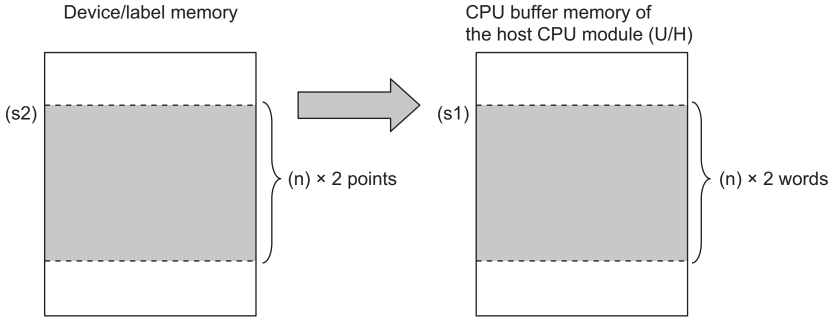
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

■DTOD(P)

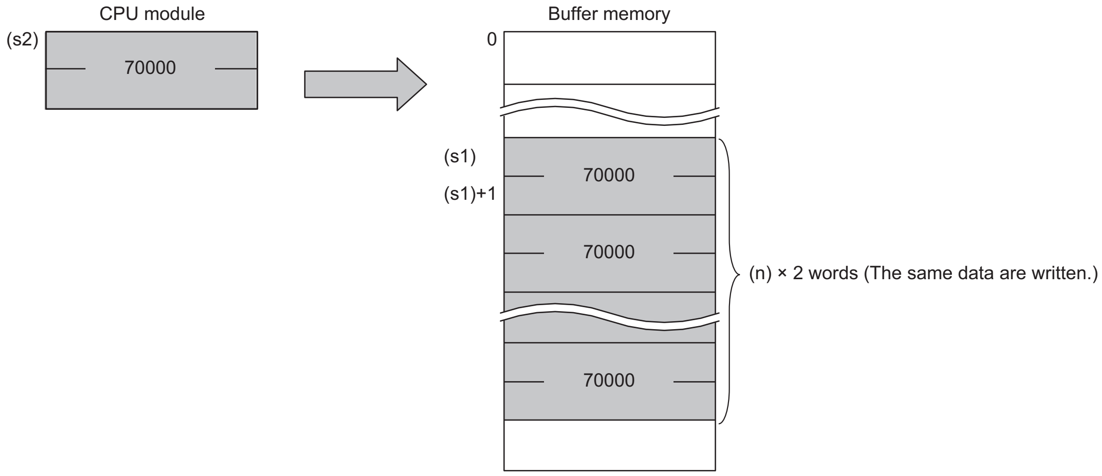
- These instructions write the (n)×2 points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host CPU module specified by (U/H).
  - Writing double word data to a module



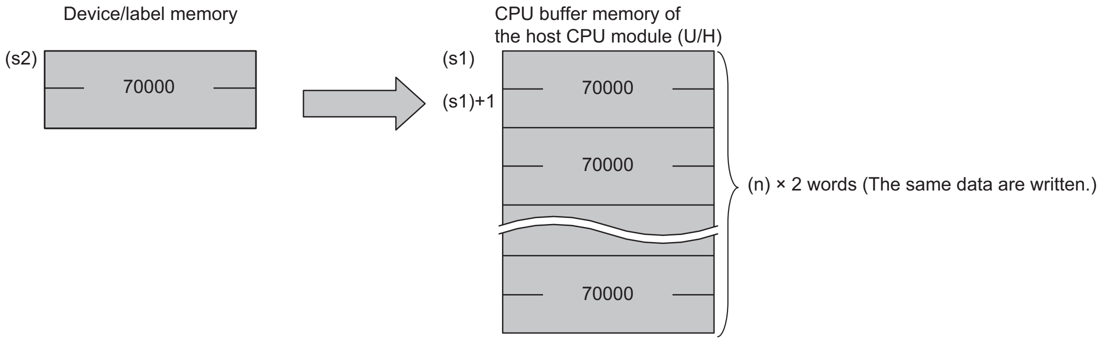
  - Writing double word data to the host CPU module



- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n)×2 words from the specified buffer memory address.
  - Writing double word data to a module



  - Writing double word data to the host CPU module



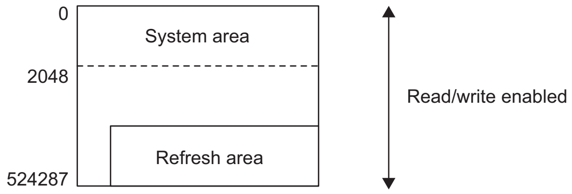
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s1) is outside the range of buffer memory or CPU buffer memory.
	The (n) points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (TOD(P) instruction)
	The 2×(n) points of data from the address specified by (s1) is outside the range of buffer memory or CPU buffer memory. (DTOD(P) instruction)

Point

- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later.



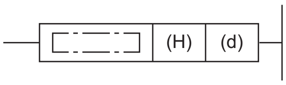
- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The TO(P) and DTO(P) instructions can write data to the buffer memory address with a capacity exceeding 64K.

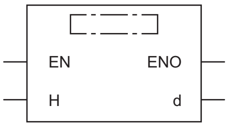
Reading the module model name

TYPERD(P)

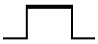
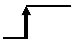
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read the module name of the specified slot.

Ladder	ST
	ENO:=TYPERD(EN,H,d); ENO:=TYPERDP(EN,H,d);

FBD/LD


■Execution condition

Instruction	Execution condition
TYPERD	
TYPERDP	

Setting data

■Description, range, data type

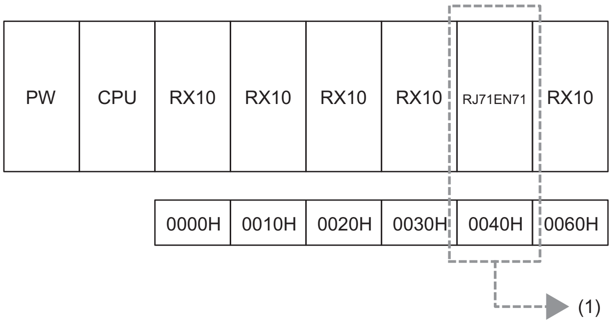
Operand	Description	Range	Data type	Data type (label)
(H)	Start I/O number (first three digits in four-digit hexadecimal representation) of a model read target module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(d)	(d)+0: Instruction execution result (d)+1 to (d)+9: Module name	—	Word	ANY16_ARRAY (Number of elements: 10)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(H)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the module name in the slot specified by (H), and store the model name in the device specified by (d) and later. The target modules are as follows.
  - CPU module
  - Input module
  - Output module
  - I/O combined module
  - Intelligent function module
- For (H), specify the start I/O number of the target module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the read-target module.

- For the slot to be specified for the read target when specifying a module that occupies two slots, refer to the number of occupied I/O points described in the manual for each module.
- To read the module name of a CPU module, specify the read start I/O number as in the following table.

CPU module	Read start I/O number
CPU No. 1	3E0H
CPU No. 2	3E1H
CPU No. 3	3E2H
CPU No. 4	3E3H

- The result of instruction execution is stored in (d)+0, and a module name is stored in (d)+1 to (d)+9. The following table lists the values to be stored in (d).

Condition	(d)+0	(d)+1 to (d)+9
The read target module has a module name.	0	Module name held by a module
The read target module does not have a module name.	1	Character string consisting of module type and the number of points
The read target slot is an empty slot.	-1	0000H
The read target module is in the course of online exchange.		
The I/O number specified in (H) is not the start number of a module.		

- When the read target module has a module name, the module name to be stored in (d)+1 and later is as follows.
  - Nine words are used.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - The module name held by a module is stored. (Note that it may differ from the module name written to the rating plate.)

Ex.

The following table lists module name examples that are stored.

Target module	Module name example stored
CPU module	R04CPU
I/O module	INPUT_16
Network module	RJ71GP21-SX

Point

If the module name in the I/O assignment setting differs from that of the mounted module, the module name held by the mounted module is stored.

- When the read target module does not have a module name, the character string to be stored in (d)+1 and later is as follows.
  - Nine words are used.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - A character string consisting of a combination of "character string indicating the module type" and "character string indicating the number of points" is stored.

Ex.

The following table lists character string examples that are stored.

Target module	Character string example stored
Input module	INPUT_16
Output module	OUTPUT_32
I/O combined module	MIXED_64
Intelligent function module	INTELLIGENT_128

The following table lists character string examples that indicate the numbers of points.

Number of points	Character string example that indicates the number of points.
16 point	_16
32 point	_32
48 point	_48
64 point	_64
128 point	_128
256 point	_256
512 point	_512
1024 point	_1024

Point

If the number of points in the I/O assignment setting differs from that of the mounted module, the number of points of the mounted module is stored.

- When reading the module name of the RnENCPU (CPU part) by specifying the I/O number, the following chracter string is stored.

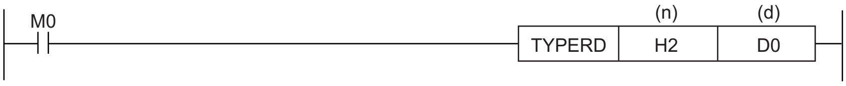
Ex.

The following table lists character string examples that are stored.

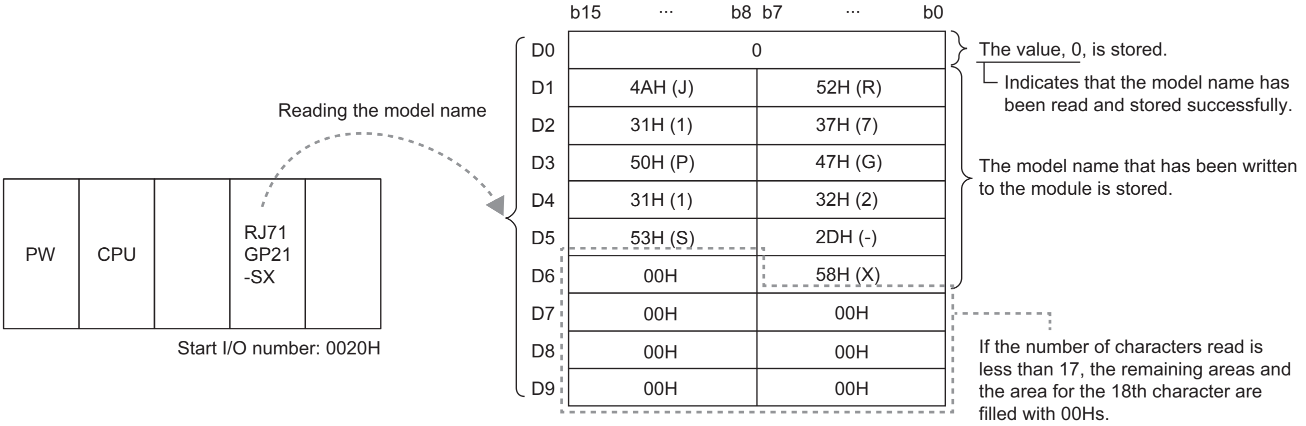
I/O number of the module	Character string example stored
3E00H	R120ENCPU
0000H	_RJ71EN71(E+CCIEF)[*1]

[*1] For the RnENCPU (network part), the character string of the module selected on the Element Selection window for the Module Configuration of the engineering tool is stored.

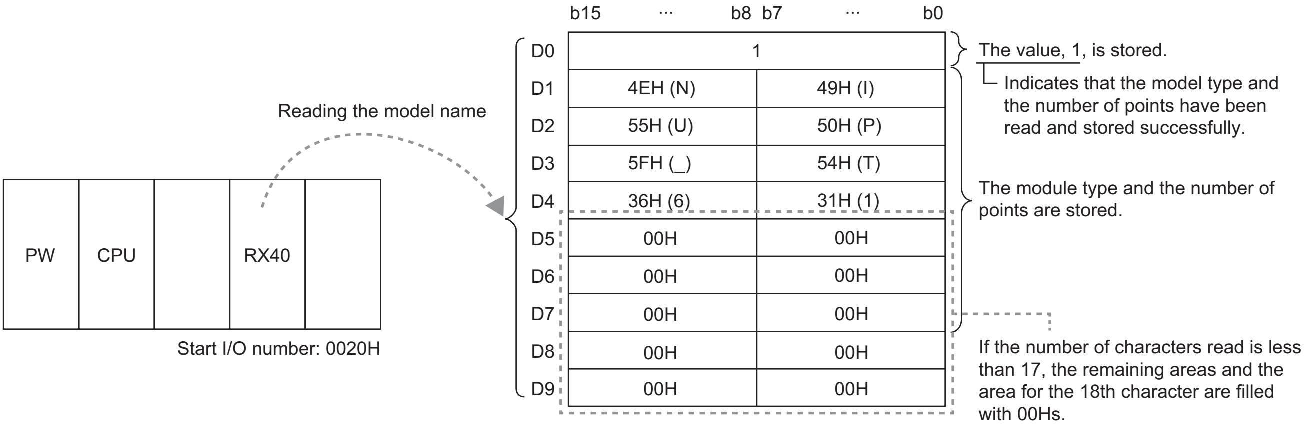
- In a multiple CPU system, the module name can also be read by specifying the module controlled by the CPU module of another CPU.
- In the following program example, when M0 turns on, the module name of the module mounted at I/O number 0020H is stored in D0 and later.



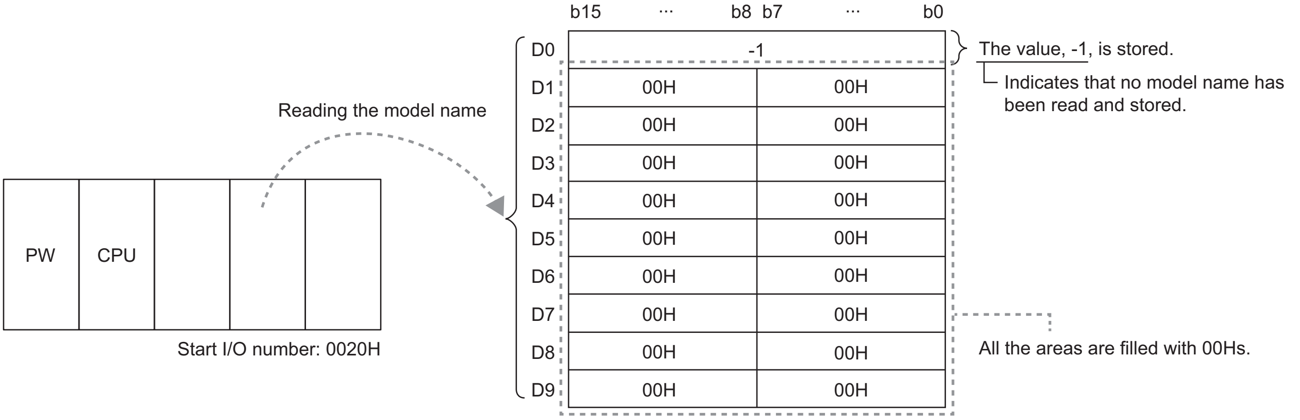
- Module having a module name (example: RJ71GP-SX)



- Module having no module name (example: RX40)



- Empty slot



Operation error

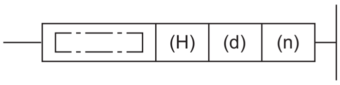
Error code (SD0)	Description
2800H	The value set to (H) is out of the range, 0 to FFH and 3E0 to 3E3H.
2810H	Communications with the read target module is disabled due to a failure of the module.

Reading module specific information

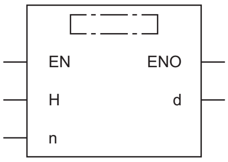
UNIINFRD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

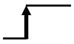
These instructions read the specified points of module information.

Ladder	ST
	ENO:=UNIINFRD(EN,H,n,d); ENO:=UNIINFRDP(EN,H,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UNIINFRD	
UNIINFRDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(H)	Start I/O number (first three digits in four-digit hexadecimal representation) of an information read target module	0H to FFH	16-bit unsigned binary	ANY16
(d)	Start device for storing module information	—	Word	ANY16[*1]
(n)	Number of read data points	0 to 256	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

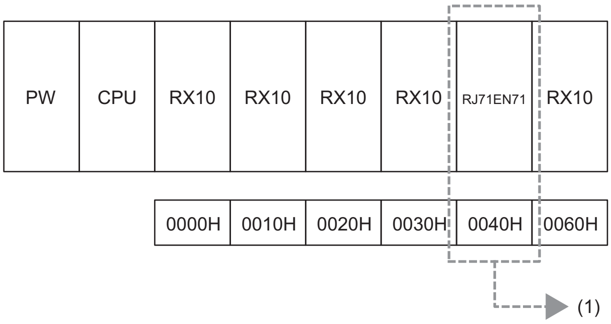
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(H)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions read the module information by the number of points specified by (n) from the module specified by (H), and store the information in the device areas specified by (d) and later. Even if the module type or the number of points is changed in I/O assignment, the status of the mounted module is read.
- For (H), specify the start I/O number of the module, whose information is to be read, with upper 3 digits when it is represented by 4 hexadecimal digits.
- If an I/O number other than the start I/O number of the read target module is specified, module information in which only the module mount status is on and any other status is off is stored.



(1) Specify K4 or H4 as the start I/O number of the read-target module.

- The following shows detailed module information.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(d)																
(d)+1																

Device	Bit	Item name	Description
(d)	b0	Number of I/O points	0000: 16 points 0001: 32 points 0010: 48 points 0011: 64 points 0100: 128 points 0101: 256 points 0110: 512 points 0111: 1024 points
	b1		
	b2		
	b3		
	b4	Module type	0000: Input module, or not set 0010: Output module 0100: I/O combined module (different numbers) 0110: I/O combined module (same number) 1000: Intelligent function module[3]
	b5		
	b6		
	b7		
	b8	Series type	010: MELSEC-Q series module 011: MELSEC iQ-R series module 111: Unknown
	b9		
	b10		
	b11	Group number	000: CPU No. 1 control 001: CPU No. 2 control 010: CPU No. 3 control 011: CPU No. 4 control
	b12		
	b13		
	b14	Start of slot	0: Not start of slot 1: Start of slot
	b15	Module mounted	0: Module not mounted 1: Module mounted[2]
(d)+1	b0	Error	00: Normal 01: Minor error 10: Medium error 11: Major error
	b1		
	b2	Module ready	0: Not ready 1: Ready
	b3	Reserved	0: Fixed
	b4	Reserved	0: Fixed
	b5	Reserved	0: Fixed
	b6	Inter-module synchronization	00: Not subject to synchronization 01: Preparing for synchronization 10: Synchronized 11: Synchronization error
	b7		
	b8	Reserved	0: Fixed
	b9	Reserved	0: Fixed
	b10	External power supply	0: During normal operation, or no external power supply 1: Power off
	b11	Fuse status	0: Normal operation 1: Fuse blown
	b12	Reserved	0: Fixed
	b13	Online module change	0: Not during online module change 1: During online module change
	b14	Reserved	0: Fixed
	b15	Module access	0: Access disabled[1] 1: Access enabled

[1] For example, the module is being connected or removed.

[2] For a module which occupies 32 points or more, information is stored in (d) + 0 and (d) + 1, and ON information is stored in the later devices only when the module is mounted.

[3] When the I/O number of the RnENCPU (network part) is specified, this information is read and stored.

- In the following program example, when M0 turns on, the information on the module mounted at I/O number 0040H is stored in D0 and later.

PW	CPU	RX10	RX10	RX10	RX10	RJ71EN71	RX10
		0000H	0010H	0020H	0030H	0040H	0060H

[Result of reading]

- Module information of the RJ71EN71 is read.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1
	(1)	(2)	(3)			(4)			(5)				(6)			

(1) Module mounting status
(2) Start of slot
(3) CPU No. 1 control
(4) MELSEC iQ-R series module
(5) Intelligent function module
(6) 32-point module

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
	(1)	(2)	(3)	(4)	(5)	(6)	(7)		(8)		(9)			(10)	(11)	

(1) Module access enabled
(2) Fixed to 0
(3) Not during online module change
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Inter-module synchronized
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

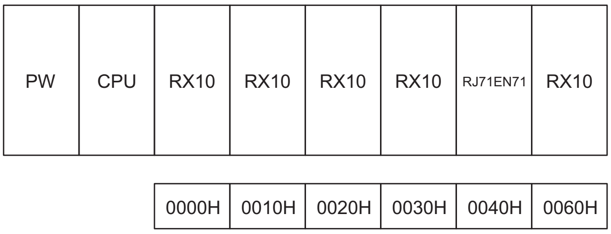
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(1)	(2)														

(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because information is stored in D0 and D1.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(1)															

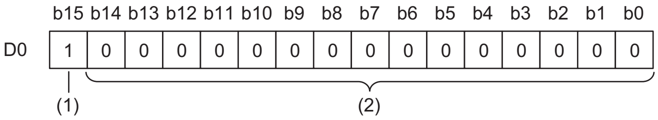
(1) All 0s because information is stored in D0 and D1.

- If an I/O number other than the start I/O number is specified in (H) for a module having 32 or more I/O points, module information in which only the module mount status is on and any other status is off is stored. In the following program example, when M0 turns on, the information on the module mounted at I/O number 0050H is stored in D0 and later.

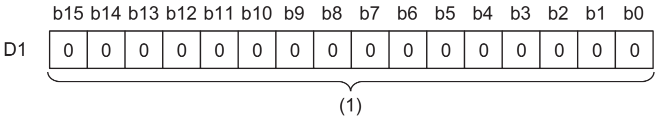


[Result of reading]

- Only the module mount status is on because the I/O number in the latter 16 points of the RJ71EN71 is specified by (H).

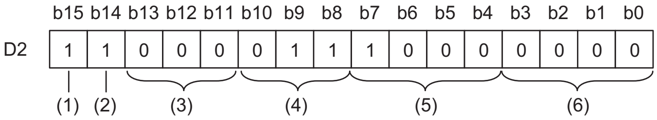


(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because the latter 16 points of a 32-point module is specified in n1.

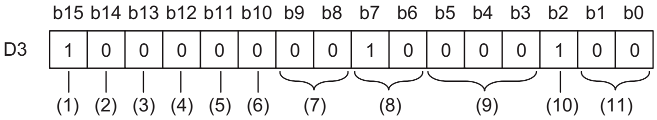


(1) All 0s because the latter 16 points of a 32-point module is specified in n1.

- Module information of RX10 is read.



(1) Module mounting status
(2) Start of slot
(3) CPU No. 1 control
(4) MELSEC iQ-R series module
(5) Input module
(6) 16-point module



(1) Module access enabled
(2) Fixed to 0
(3) Not during online module change
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Inter-module synchronized
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

Operation error

Error code (SD0)	Description
2800H	The value set to (H) is out the range, 0 to FFH.
3405H	The value set to (n) is out of the range, 0 to 256.
	The total of the values in (H) and (n) is 257 or greater.

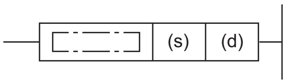
7.28 Routing Information Instructions

Reading routing information

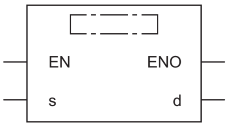
S(P).RTREAD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

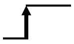
These instructions read the data of the specified transfer destination network number from the routing information set in parameter.

Ladder	ST
	ENO:=S_RTREAD(EN,s,d); ENO:=SP_RTREAD(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.RTREAD	
SP.RTREAD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer destination network number	1 to 239	16-bit signed binary	ANY16
(d)	Start device for storing the read data	—	Word	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the data of the transfer destination network number specified by (s) from the routing information set in parameter, and store the information in the device specified by (d) and later.
- If the data of the transfer destination network number specified by (s) is not set in parameter, 0 is stored in the device specified by (d) and later.
- The following figure shows the data stored in the device specified by (d) and later.

Start device	Item	Range
(d)	Relay network number	1 to 239
(d)+1	Relay station number	Refer to the following table.
(d)+2	Dummy	—

- The specification ranges of relay station number are as follows.

Network type	Specification range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network (master station)	Fixed to 125 (The fixed value is stored.)
CC-Link IE Field Network (local station)	1 to 120 (The relevant station number is stored.)

Operation error

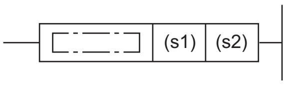
Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 239.

Registering routing information

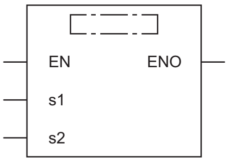
S(P).RTWRITE

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

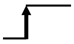
These instructions write routing information in the area with the specified transfer destination network number.

Ladder	ST
	ENO:=S_RTWRITE(EN,s1,s2); ENO:=SP_RTWRITE(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.RTWRITE	
SP.RTWRITE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Transfer destination network number	1 to 239	16-bit signed binary	ANY16
(s2)	Start device where the write data is stored	—	Word	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions write the routing information stored in the device specified by (s2) and later to the area with the transfer destination network number specified by (s1).
- The following figure shows the data stored in the device specified by (s2) and later.

Start device	Item	Range
(s2)	Relay network number	1 to 239
(s2)+1	Relay station number	Refer to the following table.
(s2)+2	Dummy	—

- The specification ranges of relay station number are as follows.

Network type	Specification range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network (master station)	Fixed to 125
CC-Link IE Field Network (local station)	1 to 120

- If the data of the transfer destination network number specified by (s1) has already been set in parameter, the data is overwritten with the data stored in the device specified by (s2) and later.
- If the data in the device areas specified by (s2) to (s2)+2 are all 0, the data of the transfer destination network number specified by (s1) is deleted from parameter.

Operation error

Error code (SD0)	Description
3405H	The value set to (s1) is out of the range, 1 to 239.
	Any of the data in the device areas specified by (s2) and later exceeds the following range. • (s2)+0: 1 to 239 • (s2)+1: 1 to 120, 125
	The total number of routing information registered in parameter of the network module and registered by using the RTWRITE instruction exceeds 238.
	A transfer destination network number which is not registered in parameter is specified as a deletion target.
	A zero is specified in only either of the device areas specified by (s2) and (s2)+1.

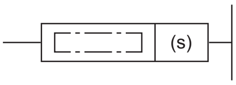
7.29 Logging Instructions

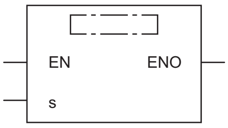
Setting trigger logging

LOGTRG

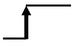
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

This instruction generates a trigger condition for the specified logging setting number in trigger logging.

Ladder	ST
	ENO:=LOGTRG(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
LOGTRG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logging setting number	1 to 10	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction generates a trigger for trigger logging with the logging setting number specified by (s).
- Specify a value 1 to 10 in (s).
- The LOGTRG instruction turns on the special relay (logging trigger) with the logging setting number in the device specified by (s), executes trigger logging for the specified number of records, latches data, and stops trigger logging.
- The instruction is enabled when "When trigger instruction executed" in the "Trigger condition".
- Even if the LOGTRG instruction is executed, no processing is performed in the following cases.
  - A logging setting number in which an item other than "When trigger instruction executed" is selected in the "Trigger condition" is specified.
  - A logging setting number with no setting is specified.
  - A logging setting number specifying the execution of continuous logging is specified.
  - Another LOGTRG instruction is executed without executing the LOGTRGR instruction after a LOGTRG instruction was executed once.

Operation error

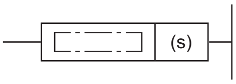
Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 10.

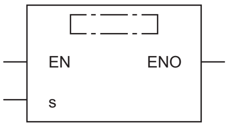
Resetting trigger logging

LOGTRGR

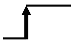
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction resets the trigger condition of the specified logging setting number.

Ladder	ST
	ENO:=LOGTRGR(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
LOGTRGR	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logging setting number	1 to 10	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction resets the LOGTRG instruction of the logging setting number specified by (s). The instruction disables the LOGTRG instruction of the specified trigger logging setting number.
- The LOGTRGR instruction turns off the special relays (logging completion, logging trigger, and after logging trigger) with the logging setting number in the device specified by (s).
- If the instruction is executed while buffer data is saved to an SD memory card, the execution of the instruction is made to wait until all data is saved completely.

Operation error

Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 10.

7.30 Program Control Instructions

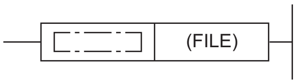
Changing the program execution type to standby type

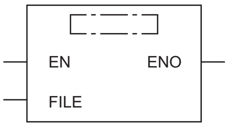
PSTOP(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

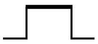
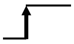
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the execution type of the program with the file name stored in the specified device to a standby type.

Ladder	ST
 FILE: File name	ENO:=PSTOP(EN,filename); ENO:=PSTOPP(EN,filename);

FBD/LD
 FILE: File name

■Execution condition

Instruction	Execution condition
PSTOP	
PSTOPP	

Setting data

■Description, range, data type

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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(file name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the execution type of the program with the file name stored in the device specified by (file name) to a standby type.
- Only programs stored in program memory can be changed to a standby type.
- The execution type of the specified program changes to a standby type during END processing.
- The PSTOP(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is the SFC program.

Point

For how to change the program execution type, refer to the following.

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

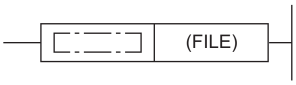
Changing the program execution type to standby type (output off)

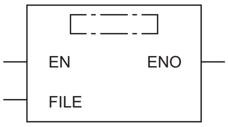
POFF(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

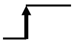
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the program execution type of the program with the file name stored in the specified device.

Ladder	ST
 FILE: File name	ENO:=POFF(EN,filename); ENO:=POFFP(EN,filename);

FBD/LD
 FILE: File name

■Execution condition

Instruction	Execution condition
POFF	
POFFP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(file name)	File name of the program to be changed to the standby type (with output set to off), or the device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(file name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

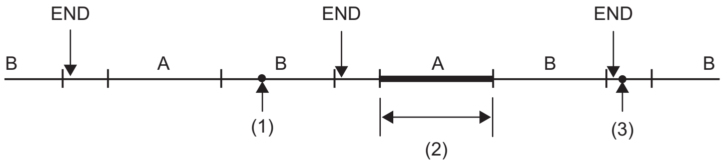
- These instructions change the execution type of the program with the file name stored in the device specified by (file name). If the program is a scan execution type, the output is turned off (non-execution processing) in the next scan. The program will be a standby type in the following scan and later. When the program is a fixed scan execution type or event execution type, it becomes a scan execution type in the next scan and turns off (non-execution processing) the output. The program will be a standby type in the following scan and later.
- Only programs stored in program memory can be changed to a standby type.
- The POFF(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is not supported.

Point

Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



A, B: Program name

(1)Specify the program A, and execute the POFF instruction.

(2)The program A is not executed.

(3)The program A changes to a standby type program.

The operation results of each coil instruction after the non-execution processing will be as follows, regardless of the on/off state of the condition contact.

- OUT instruction: The output is forcibly turned off.
- SET, RST, SFT, basic, and application instructions: Status is held.
- PLS and PLS conversion instructions (□P): Same processing as when the condition contact is set to off
- OUT T instruction: The current value of the low-speed/high-speed timer is 0.
- OUT ST and OUT C instructions: Current value is held.

For how to change the program execution type, refer to the following.

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

For the operation when the SFC program is specified, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

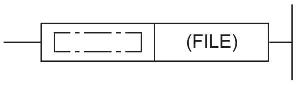
Changing the program execution type to scan execution type

PSCAN(P)

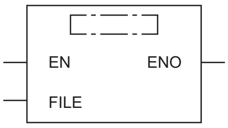
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the execution type of the program with the file name stored in the specified device to a scan execution type.

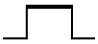
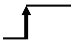
Ladder	ST
 FILE: File name	ENO:=PSCAN(EN,filename); ENO:=PSCANP(EN,filename);

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
PSCAN	
PSCANP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(file name)	File name of the program to be changed to a scan execution type, or the start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

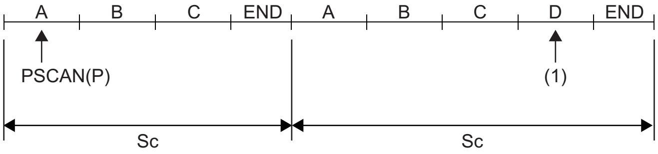
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(file name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the execution type of the program with the file name stored in the device specified by (filename) to a scan execution type.
- Only programs stored in program memory can be changed to a scan execution type.
- The execution type of the specified program changes to a scan execution type during END processing.

Ex.

While there are programs A, B, and C, the PSCAN(P) instruction is executed for program D within program A.



Sc: Scan

(1) The program D is executed.

- The PSCAN(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is not supported.
3204H	The file name specified by (file name) is the SFC program and the SFC program with another file name is already operating.

Point

For how to change the program execution type, refer to the following.

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

For the operation when the SFC program is specified, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

8 BUILT-IN ETHERNET FUNCTION INSTRUCTIONS

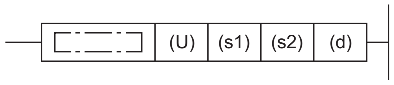
8.1 Open/Close Processing Instructions

Opening a connection

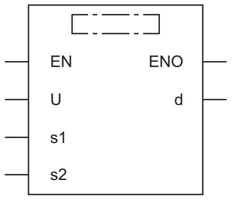
SP.SOCOPEN

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction opens the specified connection.

Ladder	ST
	ENO:=SP_SOCOPEN(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCOPEN	

Setting data

■Description, range, data type

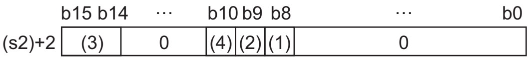
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 10)
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	Execution/completion type	Specify whether to use the parameter value set by the engineering tool or the value set in (s2)+2 to (s2)+9 of control data for opening a connection. • 0000H: Performs open processing according to the "open setting" by the engineering tool. • 8000H: Performs open processing according to the setting in (s2)+2 to (s2)+9 of control data.	0000H 8000H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Application setting area	Specify the application of a connection.  (1) Communication method (protocol) (b8) • 0: TCP/IP • 1: UDP/IP (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open (4) Predefined protocol setting (b10) • 0: Do not use the predefined protocol support function. (Use the socket communications function.) • 1: Use the predefined protocol support function.	—	User
+3	Own station port number.	Specify the port number of the own station.	0001H to 1387H, 1392H to FFFEH (0400H or later recommended)	User
+4 +5	IP address of external device[*1]	Specify the IP address of an external device.	00000001H to FFFFFFFFH (FFFFFFFFH: simultaneous broadcast)	User
+6	Destination port number	Specify the destination port number.	0001H to FFFFH (FFFFH: simultaneous broadcast)	User
+7 to +9	—	Reserved	—	System

[*1] In Unpassive open mode, the IP address of the external device and destination port number are ignored.

Point

Port numbers 0001H to 03FFH are generally reserved port numbers (WELL KNOWN PORT NUMBERS), and therefore port numbers 0400H or later should be used.

Processing details

- This instruction opens the connection specified by (s1). The setting value used for open processing is selected by (s2)+0.
- The execution status and the completion status of the SP.SOCOPEN instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the SP.SOCOPEN instruction completes, and turns off during the next END processing.

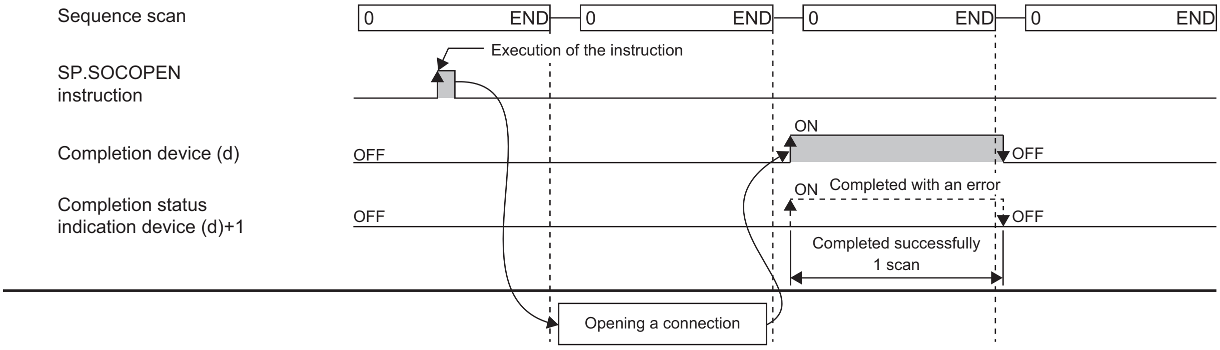
  - Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.SOCOPEN instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCOPEN instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCOPEN instruction.



- A connection which has not been set by a parameter (a connection whose protocol field is left blank) can be opened and used. To do so, set (s2)+0 to 8000H and specify the details of open in (s2)+1 to (s2+9 of control data.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

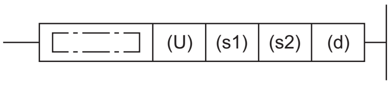
📖 MELSEC iQ-R Ethernet User's Manual (Application)

Closing a connection

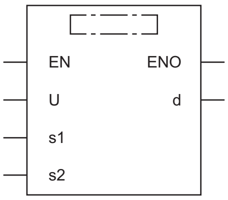
SP.SOCCLOSE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

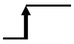
This instruction closes the specified connection.

Ladder	ST
	ENO:=SP_SOCCLOSE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCLOSE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Processing details

- This instruction performs close processing for the connection specified by (s1). The setting value used for open processing is selected by (s2)+0.
- The execution status and the completion status of the SP.SOCCLOSE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the SP.SOCCLOSE instruction completes, and turns off during the next END processing.

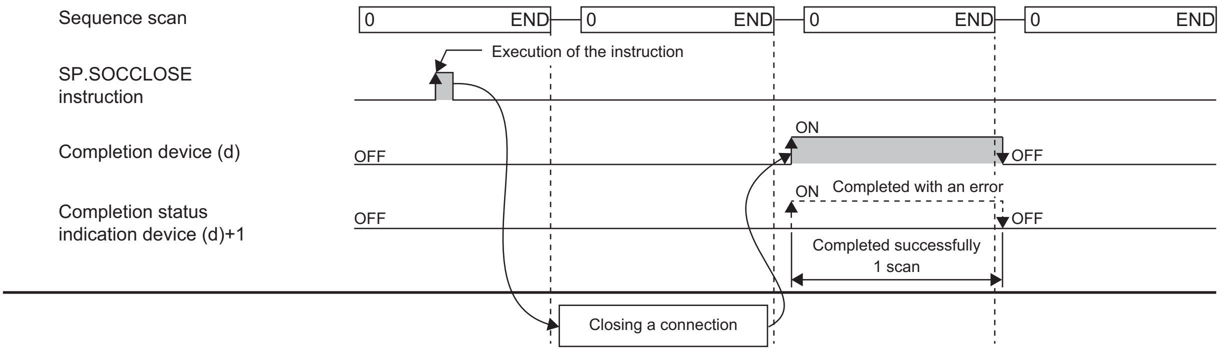
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.SOCCLOSE instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCCLOSE instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCCLOSE instruction.



Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

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

Point

When a connection waiting for opening the SP.SOCCLOSE instruction is specified in TCP Passive mode, a successful completion occurs when the SP.SOCOPEN or SP.SOCCLOSE instruction is issued and the connection is closed.

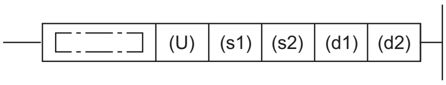
8.2 Socket Communications Instructions

Reading receive data during the END processing

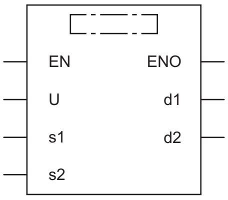
SP.SOCRCV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction reads the receive data of the specified connection, during END processing after instruction execution, from the socket communications receive data area.

Ladder	ST
	ENO:=SP_SOCRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCRCV	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d1)	Start device for storing the receive data	—	Word	ANY16[*1]
(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] 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

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (d1)

Device	Item	Description	Setting range	Set by
+0	Receive data length	The length of the data read from the socket communications receive data area is stored. (Number of bytes)	0 to 10238	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.[1]	—	System

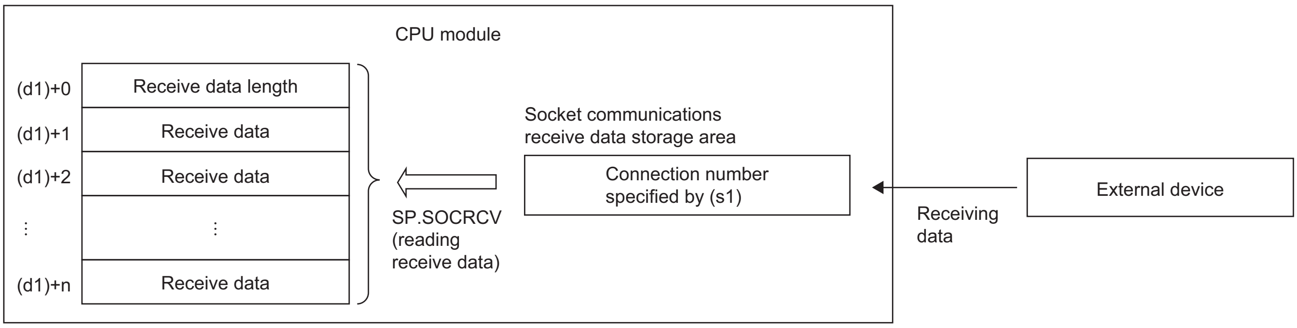
[1] The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

- When the SP.SOCRCV instruction is executed, receive data is read from the socket communications receive data area during END processing. For this reason, executing the SP.SOCRCV instruction prolongs the scan time.
- When an odd number of bytes of data is received, invalid data is stored in the higher byte of the device where the last receive data is stored.

Processing details

- The SP.SOCRCV instruction reads the receive data of the connection specified by (s1) from the socket communications receive data area (where the data received from an external device in each connection is stored) by the END processing after the instruction execution.



- The execution status and the completion status of the SP.SOCRCV instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the SP.SOCRCV instruction completes, and turns off during the next END processing.

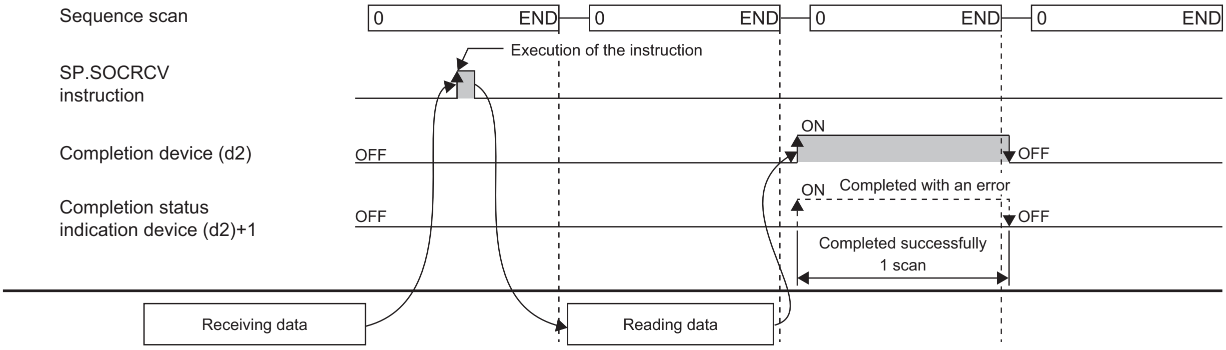
  - Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the SP.SOCRCV instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCRCV instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCRCV instruction.



Precautions

When reading receive data from the same connection, do not use this command together with the S.SOCRCVS instruction.

Operation error

Error code (SD0)	Description
2820H	The amount of data received exceeds the relevant setting area in the device/label memory in the receive data storage device.
3405H	The connection number specified by (s1) is a value other than 1 to 16.

Upon completion with an error, the completion status indication device (d2)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

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

Point

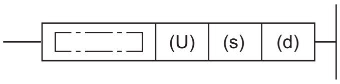
- To avoid receiving too much amount of data, the SP.SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.
- By connecting the completion device of the SP.SOCRCV instruction to the execution instruction through a normally closed contact, data can be read continuously even when it is received continuously.

Reading receive data when the instruction is executed

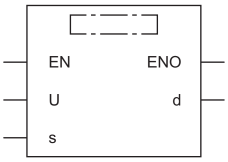
S.SOCRCVS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction reads the receive data of the specified connection from the socket communications receive data area.

Ladder	ST
	ENO:=S_SOCRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SOCRCVS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(d)	Start device for storing the receive data	—	Word	ANY16[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(U)
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (d)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the socket communications receive data area is stored. (Number of bytes)	0 to 10238	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.[1]	—	System

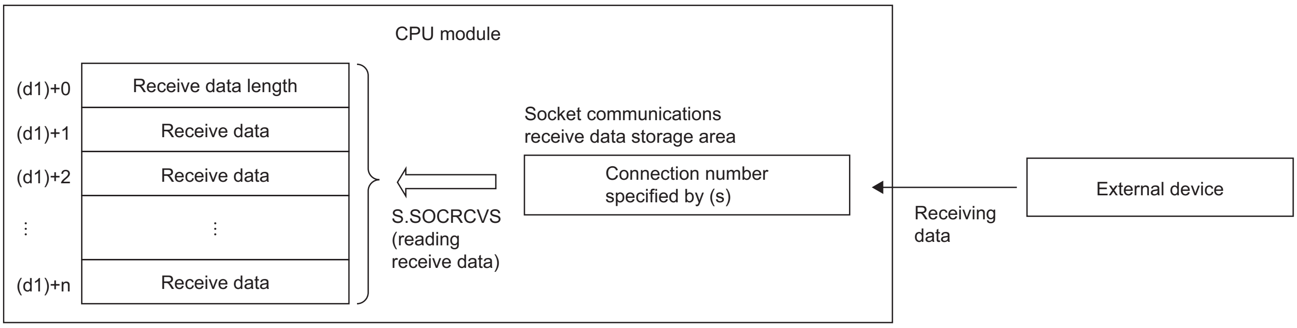
[1] The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

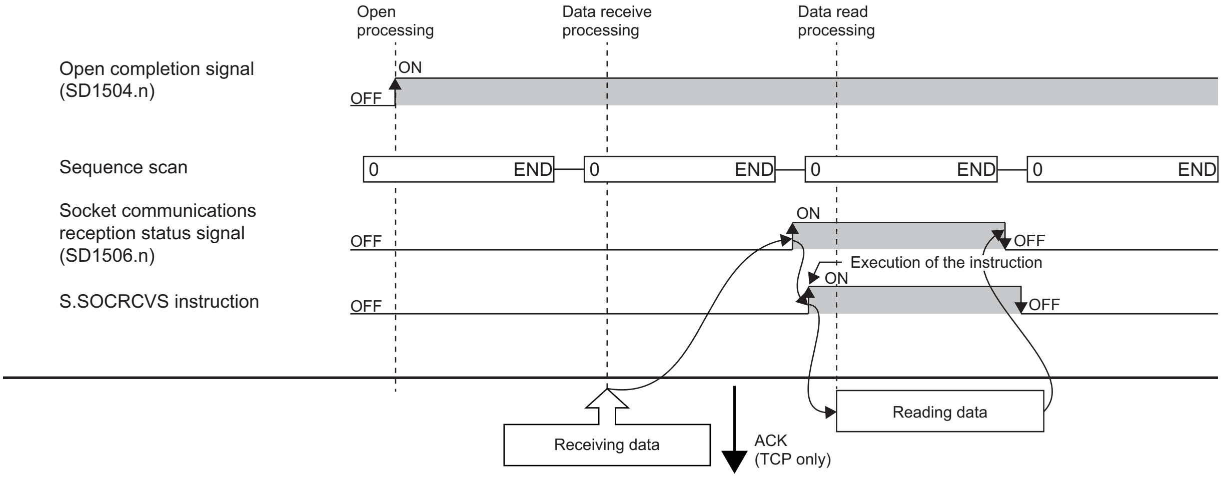
- The default receive data size is 2046 bytes. To receive 2047 bytes of data or more, change the receive data size using the SP.SOCRMODE instruction.
- When an odd number of bytes of data is received, invalid data is stored in the higher byte of the device where the last receive data is stored.

Processing details

- The S.SOCRCVS instruction reads the receive data of the connection specified by (s) from the socket communications receive data area (where the data received from an external device in each connection is stored).



- The following figure shows the timing of receive processing using the S.SOCRCVS instruction.



Precautions

When reading receive data from the same connection, do not use this command together with the SP.SOCRCV instruction.

Operation error

Error code (SD0)	Description
2820H	The amount of data received exceeds the relevant setting area in the device/label memory in the receive data storage device.
3405H	The connection number specified by (s) is a value other than 1 to 16.

Point

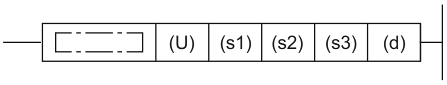
To avoid receiving too much amount of data, the SP.SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.

Sending data

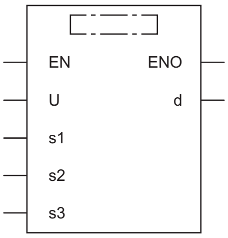
SP.SOCSND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

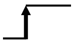
This instruction sends the data to the external device of the specified connection.

Ladder	ST
	ENO:=SP_SOCSND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCSND	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s3)	Start device for storing the send data	—	Word	ANY16[*1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(U)
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (s3)

Device	Item	Description	Setting range	Set by
+0	Send data length	Specify the send data length. (Number of bytes)	1 to 10238	User
+1 to +□	Send data	Specify the send data.[1]	—	User

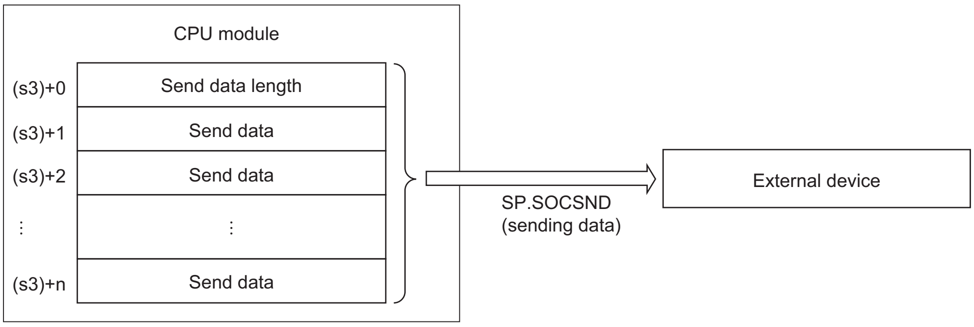
[1] The send data is sent in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last send data is stored in the lower byte of the last data storage area.

Point

When TCP is used, the send data length should be equal to or less than the maximum window size (TCP receive buffer) of the external device. Data which exceeds the maximum window size of the external device cannot be sent.

Processing details

- Sends the data in the device specified by (s3) to the external device of the connection specified by (s1).



- The execution status and the completion status of the SP.SOCSND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the SP.SOCSND instruction completes, and turns off during the next END processing.

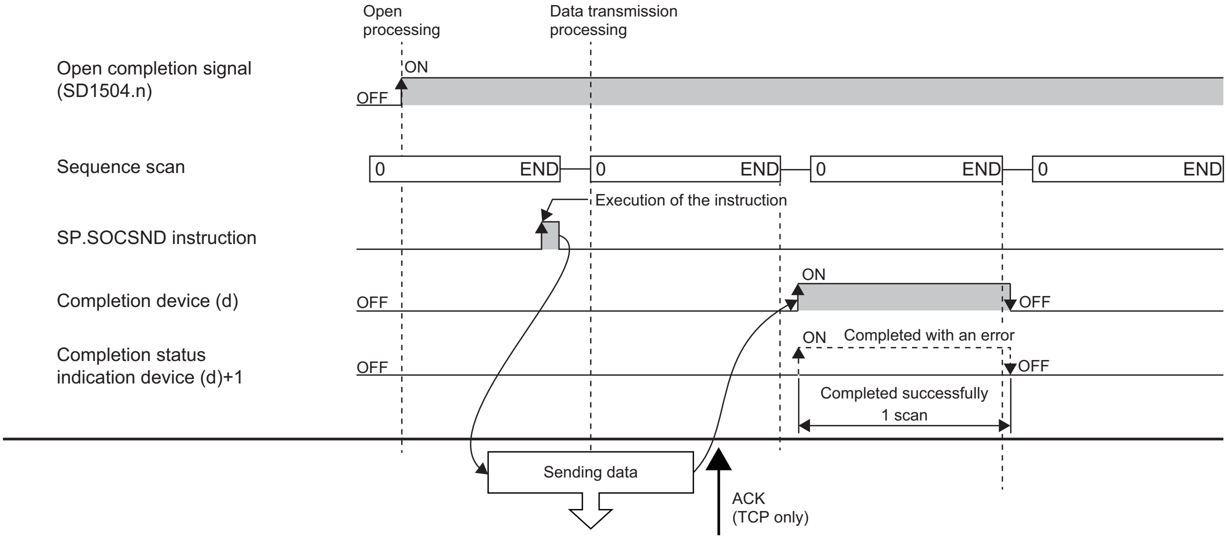
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.SOCSND instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCSND instruction completes, and turns off during the next END processing.

- The following figure shows the timing of receive processing using the SP.SOCSND instruction.



Even after the completion device turns on, data may be sent continuously. Check the completion of the send processing on the receiving side.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

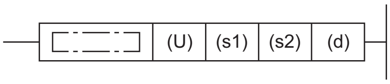
📖 MELSEC iQ-R Ethernet User's Manual (Application)

Reading connection information

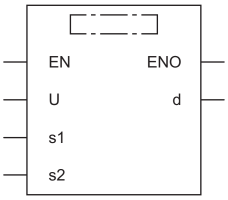
SP.SOCCINF

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction reads the connection information of the connection specified by (s1) and stores it in the device specified by (d) and later.

Ladder	ST
	ENO:=SP_SOCCINF(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCINF	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d)	Head device for storing connection information	—	Word	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

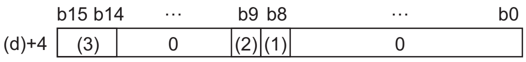
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (d)

Device	Item	Description	Setting range	Set by
+0 +1	IP address of external device	Store the IP address of an external device.	00000001H to FFFFFFFFH 00000000H: No communication destination (FFFFFFFFH: simultaneous broadcast)	System
+2	Destination port number	Store the destination port number of an external device.	0001H to FFFFH (FFFFH: Simultaneous broadcast)	System
+3	Own station port number.	Store the own station port number.	0001H to 1387H 1392H to FFFEH	System
+4	Connection use application	Store the usage of a connection.  (1) Communication method (protocol) (b8) • 0: TCP/IP • 1: UDP/IP (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open	—	System

Processing details

Reads the connection information of the connection specified by (s1).

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

When completed with an error, an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

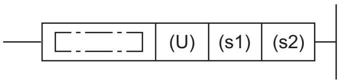
📖 MELSEC iQ-R Ethernet User's Manual (Application)

Changing the communication target (UDP/IP)

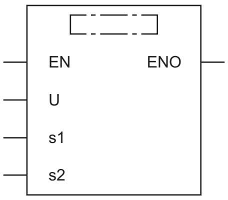
SP.SOCCSET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction changes the communication target IP address and port number of the specified connection.

Ladder	ST
	ENO:=SP_SOCCSET(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCSET	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2 +3	IP address of external device	Store the IP address of an external device.	00000001H to FFFFFFFFH (FFFFFFFFH: simultaneous broadcast)	User
+4	Destination port number	Store the destination port number of an external device.	0001H to FFFFH (FFFFH: simultaneous broadcast)	User

Processing details

In UDP/IP communications, this instruction changes the communication target IP address and port number of the connection specified by (s1).

Point

- Using the SP.SOCCSET instruction allows the user to change the communication destination without closing the connection.
- If the SP.SOCCSET instruction is executed while there is data in the receive data area, the instruction is validated after the SP.SOCRCV or S.SOCRCVS dedicated instruction is executed. If the SP.SOCCSET instruction is executed while there is no data in the receive data area, the instruction is validated soon after it is executed.

Precautions

Do not use the SP.SOCCSET instruction to change the communication destination during execution of the SP.SOCSND instruction.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

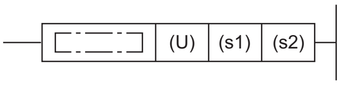
When completed with an error, an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
📖 MELSEC iQ-R Ethernet User's Manual (Application)

Changing the receive mode

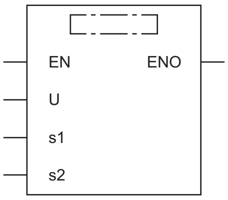
SP.SOCRMODE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

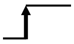
This instruction changes the TCP receive mode and receive data size for the specified connection (invalid for UDP communications connections).

Ladder	ST
	ENO:=SP_SOCRMODE(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCRMODE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 4)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	TCP receive mode[1]	Store the TCP receive mode. • 0: TCP standard receive mode • 1: TCP fixed-length receive mode	0, 1	User
+3	Receive data size	Store the size of socket communications receive data (Number of bytes)	1 to 10238	User

[1] This item is invalid for UDP communications connections.

Processing details

- This instruction changes the TCP receive mode and receive data size for the connection (other than a UDP communications connection) specified by (s1).
- For TCP connections, the function enables the mode specified by (s2)+2.

■TCP standard receive mode

Upon receipt of data, the instruction stores the data in the socket communications receive data area and turns on SD1506 (socket communications receive status signal).

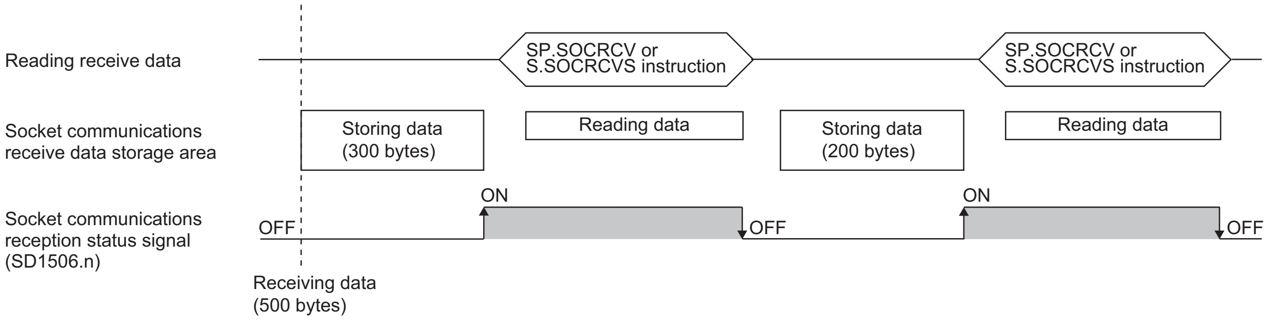
If the received data exceeds the specified receive data size, the excess data becomes the next receive data.

If data is received later before data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, it is stored in the receive data area in the OS.

If the receive data area in the OS contains data when data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, the instruction stores the data in the socket communications receive data area and turns on SD1506 (socket communications receive status signal).

Ex.

When 500 bytes of data is received while the receive data size is set to 300 bytes



■TCP fixed-length receive mode

Upon receipt of data, the instruction stores the data in the socket communications receive data area. If the specified receive data size is not reached, SD1506 (socket communications receive status signal) does not turn on.

Data reception is repeated until the received data reaches the receive data size. When it reaches the receive data size, SD1506 (socket communications receive status signal) turns on.

If the received data exceeds the specified receive data size, the excess data becomes the next receive data.

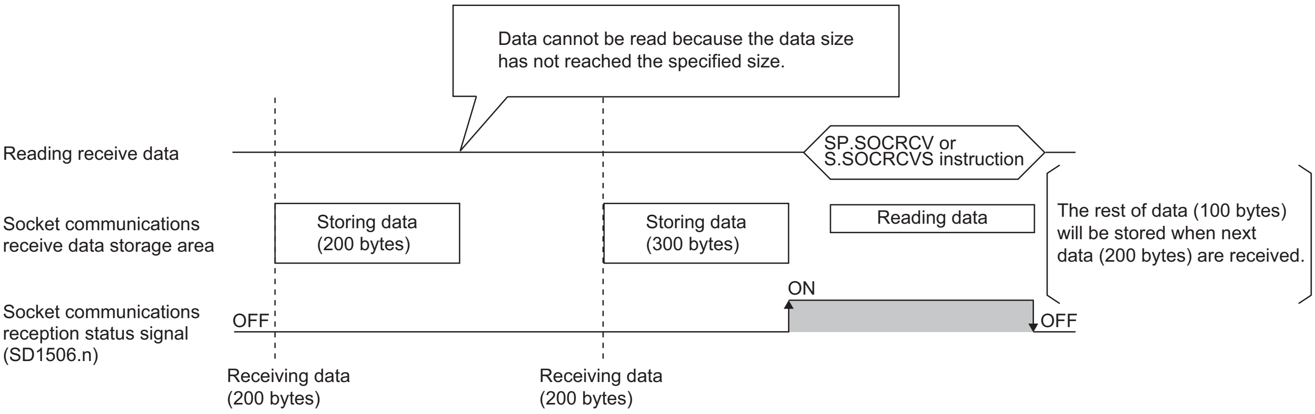
If data is received later before data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, it is stored in the receive data area in the OS.

If the receive data area in the OS contains data when data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, the instruction stores the data in the socket communications receive data area, but does not turn on SD1506 (socket communications receive status signal) if the data has not reached the specified receive data size.

Thereafter, data reception is repeated until the received data reaches the receive data size. When it reaches the receive data size, SD1506 (socket communications receive status signal) turns on.

Ex.

When 200 bytes of data is received continuously while the receive data size is set to 300 bytes



Point

- Effective use of devices

The receive data storage device used by the SP.SOCRCV or S.SOCRCVS instruction needs a 1024-word area by default. Specifying the receive data size in 1024 words or less enables effective use of the device.

- Preventing receive data from being divided

Depending on the line type, data to be received from the external device may be divided before arrival. In this case, specifying the receive data size in TCP fixed-length receive mode can prevent receive data from being divided.

- Preventing receive data from being connected

Due to a delay in receive processing of the sequence program, data which has been divided and sent may be connected before receiving depending on the external device.

Specifying the receive data size in TCP fixed-length receive mode enables data to be correctly divided and received.

- The size of the receive data to be read once by the SP.SOCRCV or S.SOCRCVS instruction is specified in (s2)+3. In the case of UDP, if the received data exceeds the specified receive data size, the excess data becomes the next receive data.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

When completed with an error, an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

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

Point

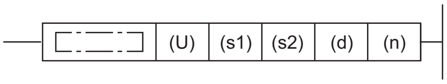
When the receive status signal does not turn on in TCP fixed-length receive mode, the data received as of the current time can be read with the SP.SOCRDATA instruction to check whether the data sent from the external device is missing.

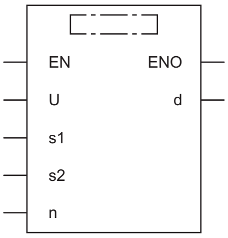
Reading socket communications receive data

S(P).SOCRDATA

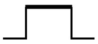
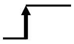
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)** ✗

These instructions read data by the number of words specified by (n) from the socket communications receive data area of the connection specified by (s1), and store them in the device specified by (d) and later.

Ladder	ST
	ENO:=S_SOCRDATA(EN,U,s1,s2,n,d); ENO:=SP_SOCRDATA(EN,U,s1,s2,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
S.SOCRDATA	
SP.SOCRDATA	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d)	Head device for storing the data that has been read	—	Word	ANY16[*1]
(n)	Number of read data	1 to 5120	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Processing details

These instructions read data by the number of words specified by (n) from the socket communications receive data area of the connection specified by (s1), and store them in the device specified by (d) and later. If the read data (n) is 0, no processing is performed.

Point

- The receive data length can be read by setting the number of read data to 1 word. As a result, the device for storing receive data when the SP.SOCRCV or S.COSCRCVS instruction is executed can be changed.
- After issuing the S(P).SOCRDATA instruction to check the data to be received this time and issuing the SP.SOCRMODE instruction to specify the size of the data to be received next time, the SP.SOCRCV or S.SOCRCVS instruction can be used to read the data of this time. As a result, based on the data received this time, the size of data to be received next can be specified.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.
	The value of the device specified by (n) exceeds 5120.

When completed with an error, an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

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

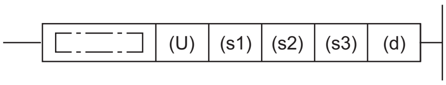
8.3 Predefined Protocol Support Function Instruction

Executing the registered protocols

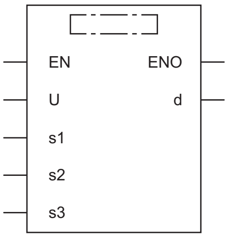
SP.ECPRTCL

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

This instruction executes the protocol that has been set by the predefined protocol support function.

Ladder	ST
	ENO:=SP_ECPRTCL(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.ECPRTCL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Number of protocols to be executed continuously	1 to 8	16-bit unsigned binary	ANY16
(s3)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s2)	○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s3)	○	—	○*1	—	—	—	—	○*1	—	—	—	—
(d)	○*1	—	○	—	—	—	—	○*1	—	—	—	—

*1 A local device and a file register which is set for each program cannot be used.

■Control data

Operand: (s3)

Device	Item	Description	Setting range	Set by
+0	Resulting number of executed protocols	The number of protocols executed by the SP.ECPRTCL instruction is stored. Any protocol where an error occurred is also included in the execution number. If the setting of setting data or control data contains an error, "0" is stored.	0, 1 to 8	System
+1	Completion status	The completion status is stored upon completion of the instruction. When two or more protocols are executed, the execution result of the protocol executed last is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Execution protocol number 1	Specify the number of the protocol to be executed first.	1 to 128	User
+3	Execution protocol number 2	Specify the number of the protocol to be executed second.	0, 1 to 128	User
+4	Execution protocol number 3	Specify the number of the protocol to be executed third.	0, 1 to 128	User
+5	Execution protocol number 4	Specify the number of the protocol to be executed fourth.	0, 1 to 128	User
+6	Execution protocol number 5	Specify the number of the protocol to be executed fifth.	0, 1 to 128	User
+7	Execution protocol number 6	Specify the number of the protocol to be executed sixth.	0, 1 to 128	User
+8	Execution protocol number 7	Specify the number of the protocol to be executed seventh.	0, 1 to 128	User
+9	Execution protocol number 8	Specify the number of the protocol to be executed eighth.	0, 1 to 128	User
+10	Collation match Receive packet number 1	If receiving is included in the communication type of the protocol that has been executed first, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the first protocol, "0" is stored.	0, 1 to 16	System
+11	Collation match Receive packet number 2	If receiving is included in the communication type of the protocol that has been executed second, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the second protocol, "0" is stored. If the number of protocols executed is less than 2, "0" is stored.	0, 1 to 16	System
+12	Collation match Receive packet number 3	If receiving is included in the communication type of the protocol that has been executed third, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the third protocol, "0" is stored. If the number of protocols executed is less than 3, "0" is stored.	0, 1 to 16	System
+13	Collation match Receive packet number 4	If receiving is included in the communication type of the protocol that has been executed fourth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fourth protocol, "0" is stored. If the number of protocols executed is less than 4, "0" is stored.	0, 1 to 16	System

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+14	Collation match Receive packet number 5	If receiving is included in the communication type of the protocol that has been executed fifth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the fifth protocol, "0" is stored. If the number of protocols executed is less than 5, "0" is stored.	0, 1 to 16	System
+15	Collation match Receive packet number 6	If receiving is included in the communication type of the protocol that has been executed sixth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the sixth protocol, "0" is stored. If the number of protocols executed is less than 6, "0" is stored.	0, 1 to 16	System
+16	Collation match Receive packet number 7	If receiving is included in the communication type of the protocol that has been executed seventh, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the seventh protocol, "0" is stored. If the number of protocols executed is less than 7, "0" is stored.	0, 1 to 16	System
+17	Collation match Receive packet number 8	If receiving is included in the communication type of the protocol that has been executed eighth, the receive packet number successful in collation match is stored. If the communication type is "receive only", "0" is stored. If an error occurs during execution of the eighth protocol, "0" is stored. If the number of protocols executed is less than 8, "0" is stored.	0, 1 to 16	System

Processing details

- This instruction executes the protocol registered using the engineering tool. Using the connection specified by (s1), the instruction executes the protocol in accordance with the control data stored in the device specified by (s3) and later.
- The instruction continuously executes as many protocols as specified by (s2) (a maximum of 8 protocols) at one time.
- The number of executed protocols is stored in the device specified by (s3)+0.
- The protocol execution status can be checked with the predefined protocol support function execution status check area (Un\G350 to Un\G669). (📖 MELSEC iQ-R Ethernet User's Manual (Application))
- The execution status and the completion status of the SP.ECPRTCL instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

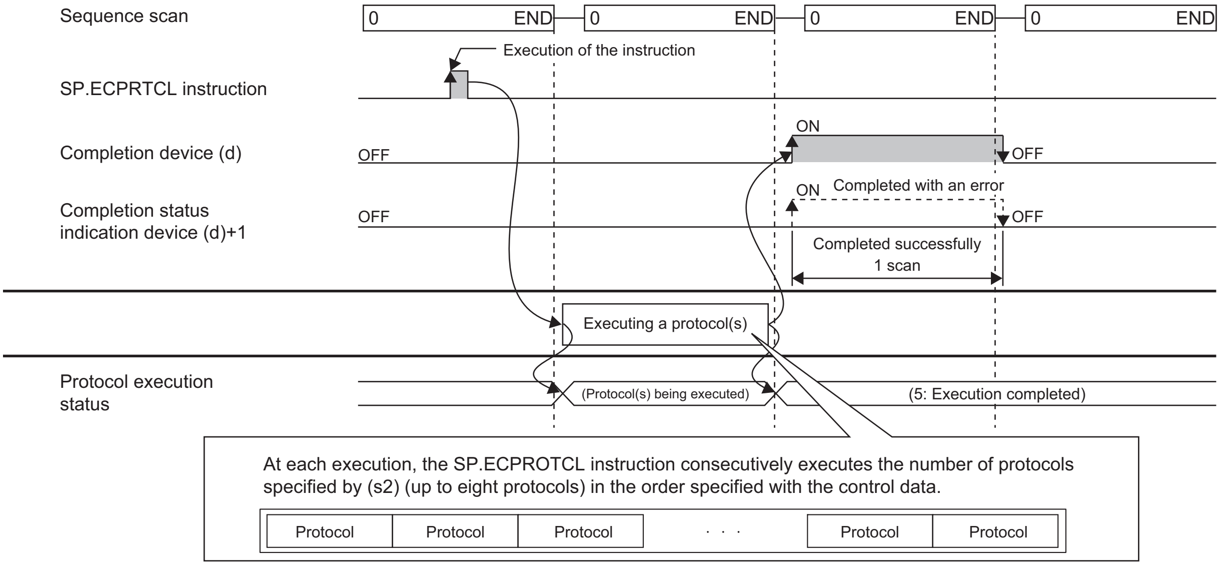
- Completion device (d)

This device turns on during END processing of the scan where the SP.ECPRTCL instruction completes, and turns off during the next END processing.

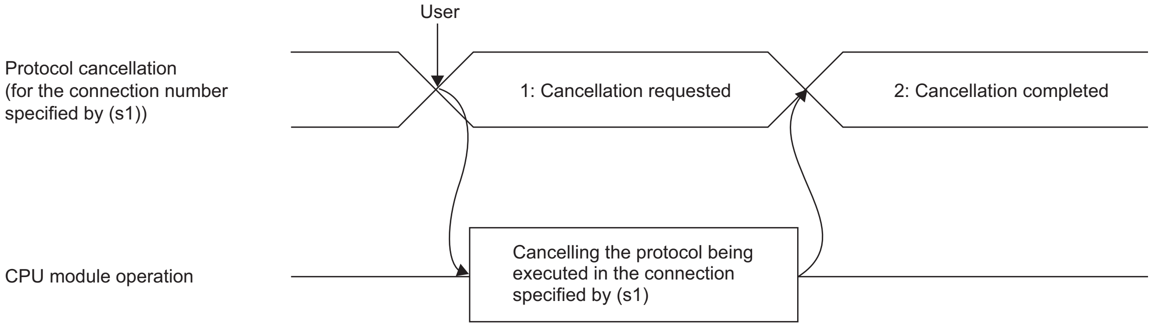
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.ECPRTCL instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the SP.ECPRTCL instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s3)+1.

- The following figure shows the SP.ECPRTCL instruction execution timing.

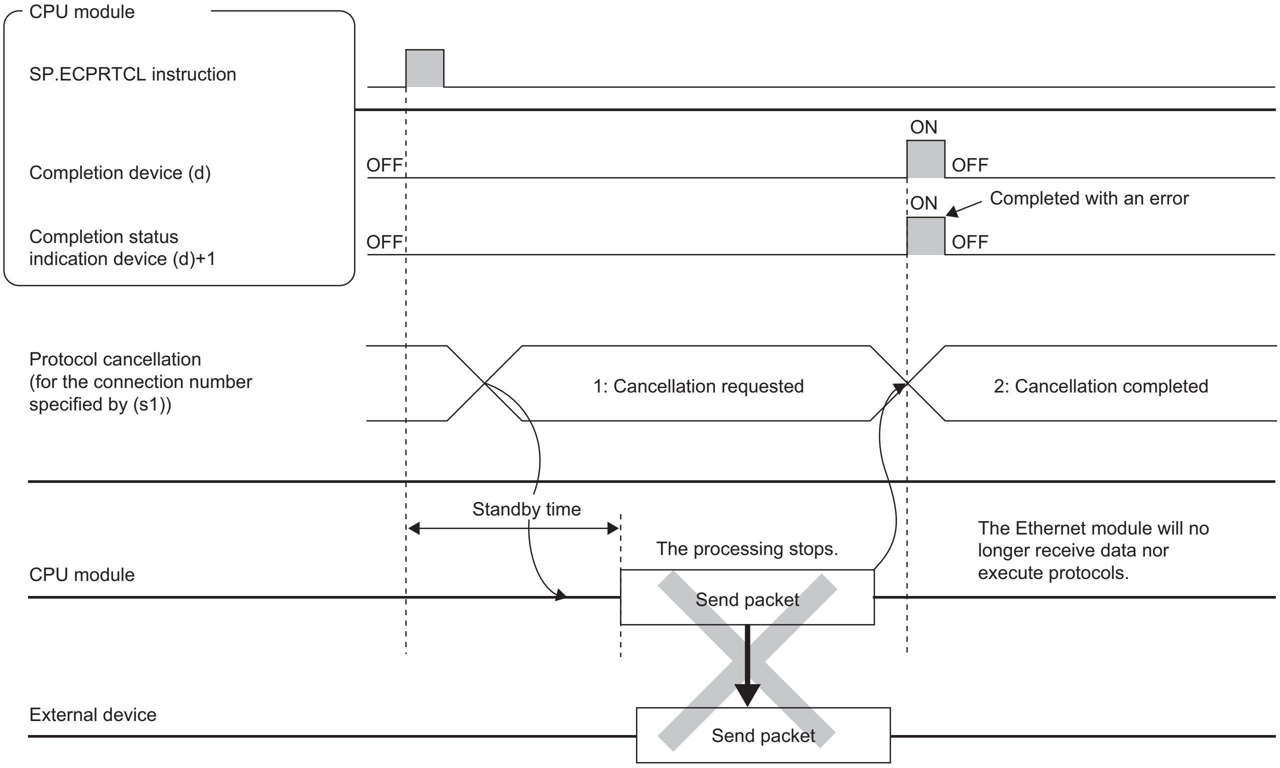


- Protocol execution can be canceled by setting a protocol cancel request. The protocol cancel request is specified in the predefined protocol support function execution status check area (Un\G350 to Un\G669). (📖 MELSEC iQ-R Ethernet User's Manual (Application))



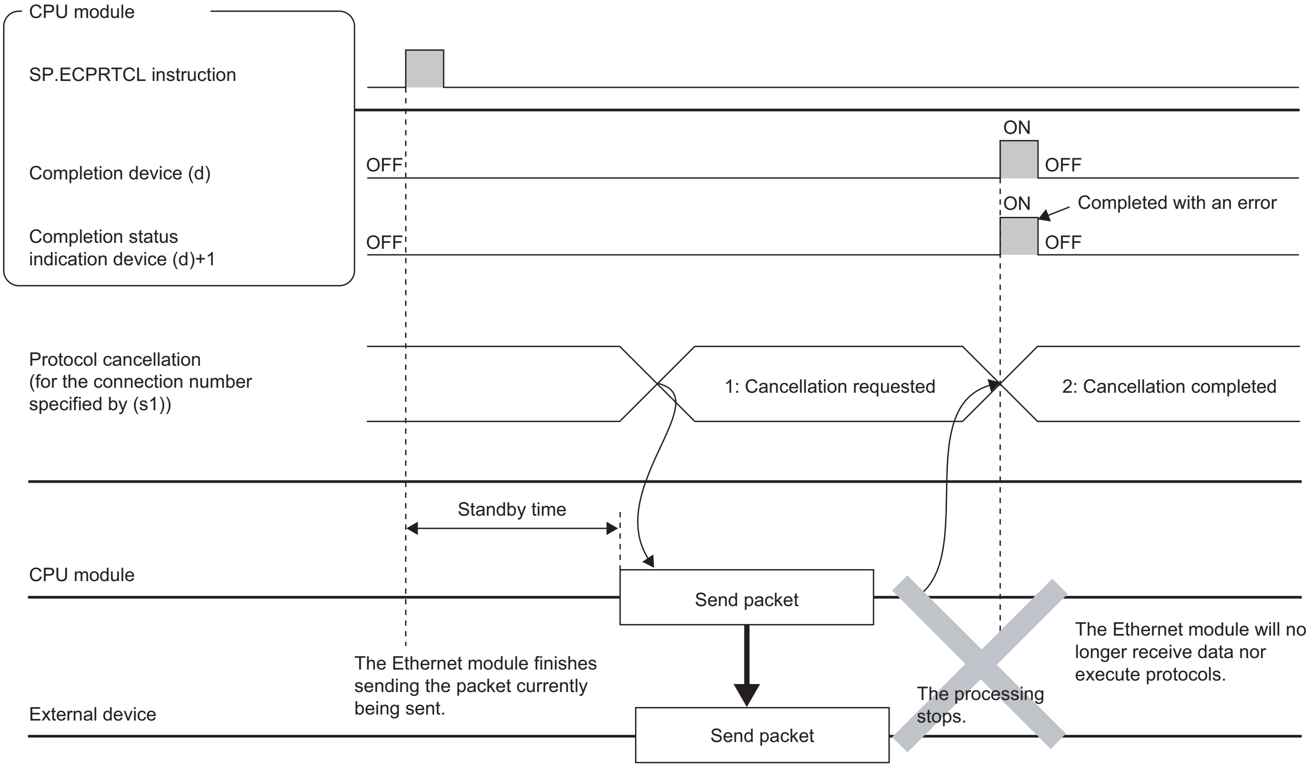
- The following figure shows the protocol cancel operations from time to time.
  - If a cancel request is issued before transmission

The following figure shows the operation when the protocol execution status is "1: Waiting for transmission".



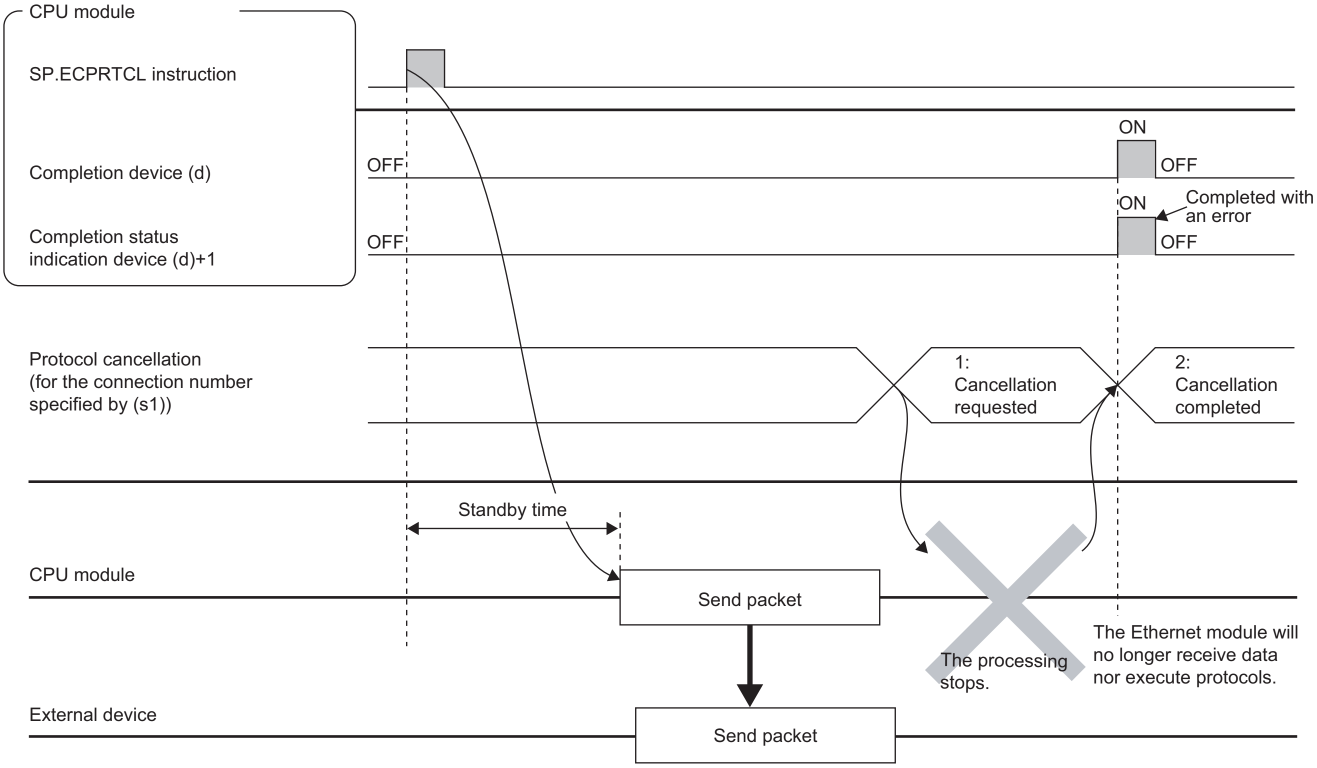
- If a cancel request is issued before completion of transmission

The following figure shows the operation when transmission has not been completed while the protocol execution status is "2: Sending".



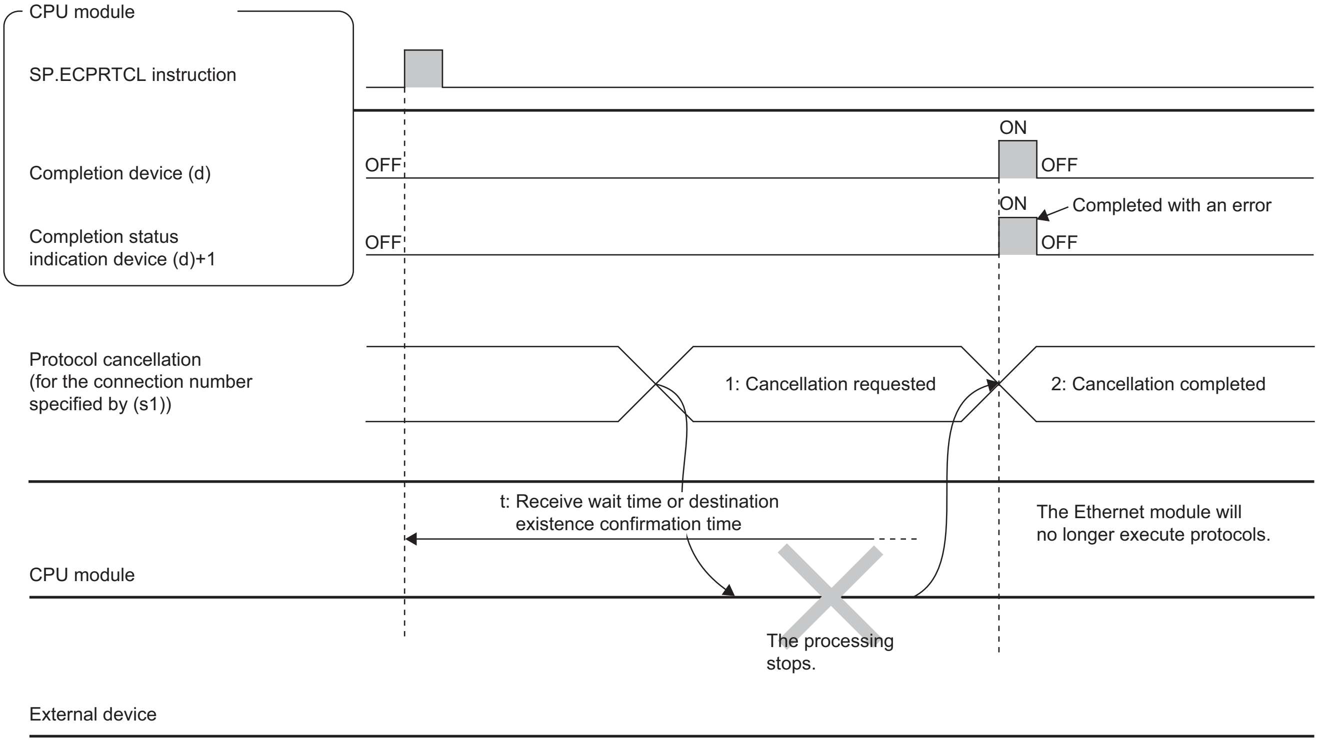
- If a cancel request is issued upon completion of transmission

The following figure shows the operation when transmission has been completed while the protocol execution status is "2: Sending".



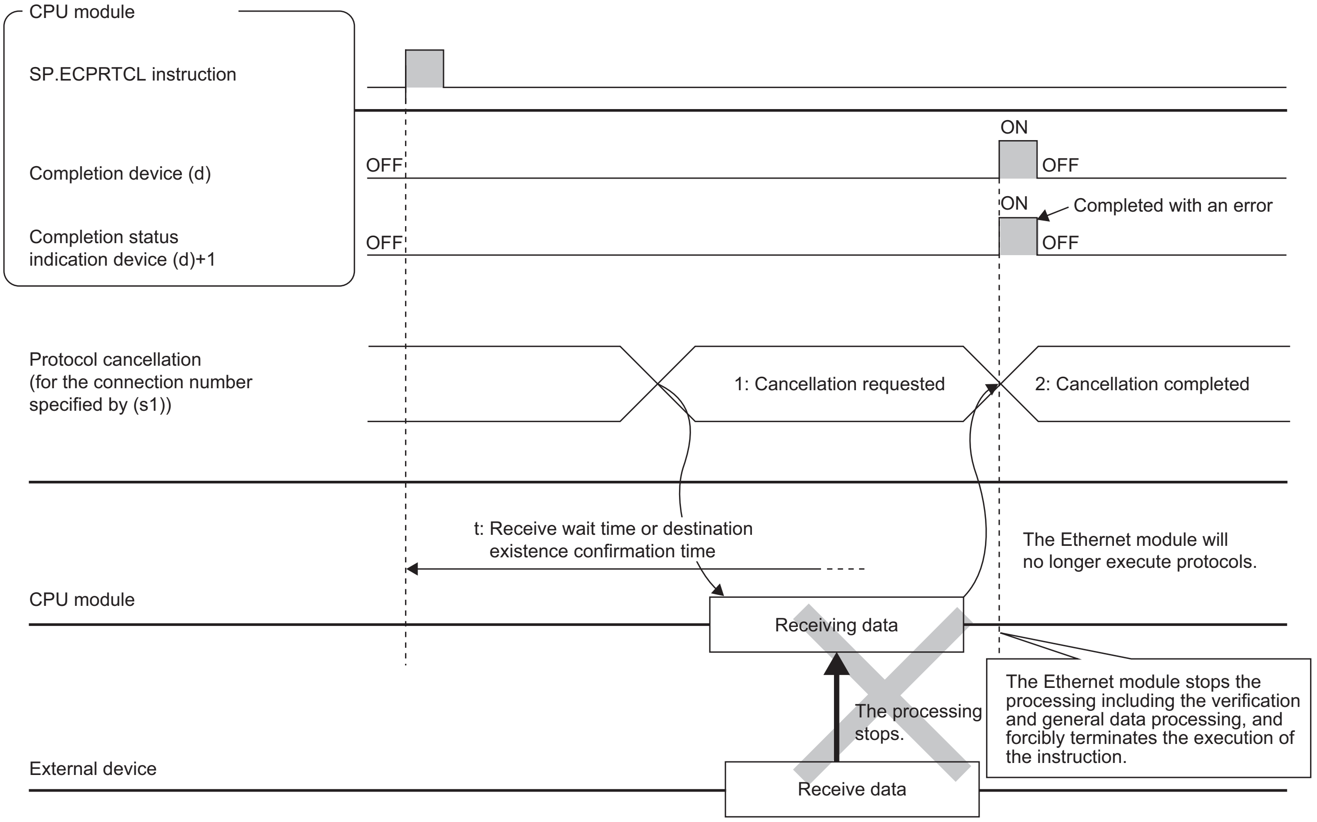
- If a cancel request is issued while waiting for reception

The following figure shows the operation when the protocol execution status is "3: Waiting for data reception".



- If a cancel request is issued during receiving

The following figure shows the operation when the protocol execution status is "4: Receiving".



Operation error

Error code (SD0)	Description
3405H	(s1) is not a value in the range from 1 to 16.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s3)+1.

For the error code stored in the completion status (s3)+1, refer to the following.

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

Precautions

- If an error occurs in the mth protocol while multiple protocols are being executed, the instruction does not execute the "m+1"th protocol and after and is completed with an error.
- When a protocol including no-conversion variables is executed, the total data length of the variables used in one packet may exceed 1920 bytes. In this case, the instruction may obtain CPU device values over several scans. Therefore, do not change the CPU device values specified in non-conversion variables from the start of the instruction to the end of execution.
- Protocol cancellation
  - The SP.ECPRTCL instruction is completed with an error and stores the protocol cancel request error (C404H) in the device (completion status) specified by (s3)+1.
  - If a cancel request is issued while no protocol is being executed, the CPU module completes the cancel request without performing any processing.
  - While no communication protocol is used, any cancel request is ignored if issued.
  - When multiple protocols are executed continuously, a cancel request may be issued during execution of the nth protocol. In this case, the CPU module forcibly terminates the nth protocol and does not execute the subsequent protocols. Protocol number n being executed is stored in the device specified by ((s3)+0), the receive packet number successful in comparison match is stored in the device specified by 1 to (n-1), and the protocol cancel request error (C404H) is stored in the device specified by ((s3)+1).
  - The CPU module periodically checks for a cancel request. For this reason, it may take time until cancel processing is performed after a cancel request is issued.
- The SP.ECPRTCL instruction itself does not open/close a connection and therefore the SP.SOCOPEN/SOCCLOSE instructions need to be used to open/close the connection.

☞ Page 1085 SP.SOCOPEN,Page 1088 SP.SOCCLOSE

- If same instructions are executed for the same connection, the subsequent instruction is ignored and is not executed until the preceding instruction is completed.
- If the receive waiting time is set to "0: Infinite wait", the SP.ECPRTCL instruction is not completed until the data specified in the protocol setting is received.

8.4 SLMP Frame Send Instruction

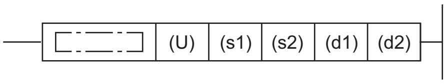
Sending an SLMP frame

SP.SLMPSND

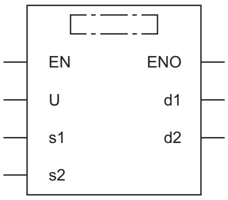
RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

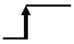
This instruction sends SLMP messages to the SLMP-compatible device.

Ladder	ST
	ENO:=SP_SLMPSND(EN, U, s1, s2, d1, d2)

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SLMPSND	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	☞ Page 1118 Control data	Word	ANY16_ARRAY (Number of elements: 19)
(s2)	Start device where a request frame is stored	☞ Page 1119 Request frame	Word	ANY16_ARRAY[*1]
(d1)	Start device for storing a response frame	☞ Page 1120 Response frame	Word	ANY16_ARRAY[*1]
(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] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

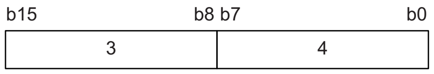
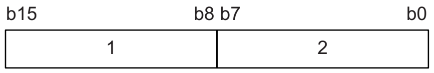
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(U)
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○[1]	—	—	—	—	○	—	—	—	—

[1] T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/error completion type	b15 b7 b0 1 0 (2) 0 (1) (1) Execution type (b0) • 0: Without arrival check (The instruction is regarded as completed when a request message is sent from own station.)[1] • 1: With arrival check (The instruction is regarded as completed when a response message is received from the external device.) (2) Error completion type (b7) Specify whether to set data at completion with an error or not. • 0: Do not set data at completion with an error in (s1)+13 and later. (Clear (s1)+13 and later.) • 1: Set data at completion with an error in (s1)+13 and later.	—	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
+2	Own station channel	Specify the channel to be used by own station. Since whether or not a serial number[2] is given to the request message depends on the channel, specify the channel as follows according to the application. • 1: No serial number is given • 2 to 9: Serial number is given	1 to 9	User
+3	IP address of external device (third and fourth octets)	Specify the IP address (third and fourth octets) of external device. b15 b8 b7 b0 3 4 3, 4: Indicates the octets of the IP address.	00000001H~FFFFFFFEH (both (s1)+3 and (s1)+4 together) (1 to 4294967294)	User
+4	IP address of external device (first and second octets)	Specify the IP address (first and second octets) of external device. b15 b8 b7 b0 1 2 1, 2: Indicates the octets of the IP address.		User
+5	Destination port number	Specify the destination port number.	1 to 65534 (1 to FFFEH)	User
+6	Request destination network number	0000H (fixed)	0000H	User
+7	Request destination station number	00FFH (fixed)	00FFH	User
+8	Request destination module I/O number	Specify a module I/O number of the request destination. • 03FFH: Own station/control CPU • 03E0H: Multiple CPU No. 1 • 03E1H: Multiple CPU No. 2 • 03E2H: Multiple CPU No. 3 • 03E3H: Multiple CPU No. 4 • 03D0H: Control system CPU • 03D1H: Standby system CPU • 03D2H: System A CPU • 03D3H: System B CPU	03FFH, 03E0H to 03E3H, 03D0H to 03D3H	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+9	Request destination multidrop station number	0000H (fixed)	0000H	User
+10	Number of resends	The device becomes effective when the execution type specified by (s1)+0 is "1: With arrival check". ■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+11. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+11	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent the number of times specified in (s1)+10. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
+12	Clock setting flag	The validity status (valid or invalid) of the data in (s)+13 and later is stored. Note that the data in (s1)+13 and later is cleared when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+13	Clock data	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+14		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)		
+15		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)		
+16		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))		
+17	IP address of error detected device (third and fourth octets)	The IP address (third and fourth octets) of the station where an error was detected is stored.  3, 4: Indicates the octets of the IP address.	—	System
+18	IP address of error detected device (first and second octets)	The IP address (first and second octets) of the station where an error was detected is stored.  1, 2: Indicates the octets of the IP address.	—	System

*1 If (s1)+0 is set to "0: Without arrival check", receive data is not set. Set 0 in (s1)+0 in the following cases:
· When a command that does not return a response message is used
· When a response message is not referred to

*2 Give the serial numbers when sending several request messages to the same SLMP-compatible device. Serial numbers to be given are automatically numbered by the system. For the serial number, refer to the following.
📖 SLMP Reference Manual

■Request frame

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the data length from the monitoring timer to the request data. (In units of bytes)	1 to 2000	User
+1	Monitoring timer	This timer sets the waiting time for the external device that received a request message to wait for the response after it issued a processing request to the access destination. (Unit: Increments of 250ms) • 0: Infinite wait • 1 to 65535: 1 to 65535 × 250ms	0 to 65535	User
+2 to +□	Request data	The request data of the SLMP message is stored.	—	User

■Response frame

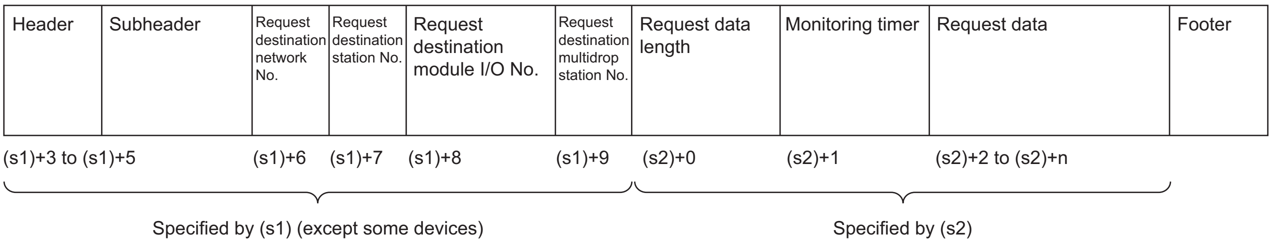
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The data length from the end code to the response data is stored. (In units of bytes)	2 to 2000	System
+1	End code	The result of command processing is stored. In normal end, 0 is stored. In abnormal end, an error code set by the external device is stored.	—	System
+2 to +□	Response data	Execution results for the request data are set. (Some commands do not return response data.)	—	System

Processing details

- This instruction sends the request frame in the device specified by (s2) and later to the external device specified by the external device IP address in the control data. When a response message is received from the external device, it is stored in the device specified by (d1).

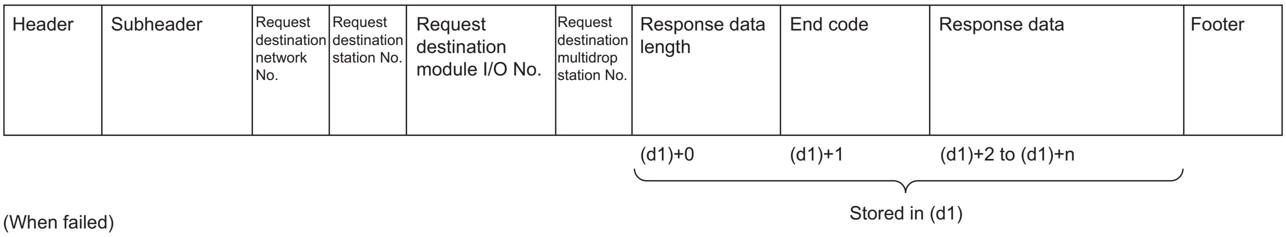
The following figures show the request data and the response data in normal/abnormal end.

●Request data

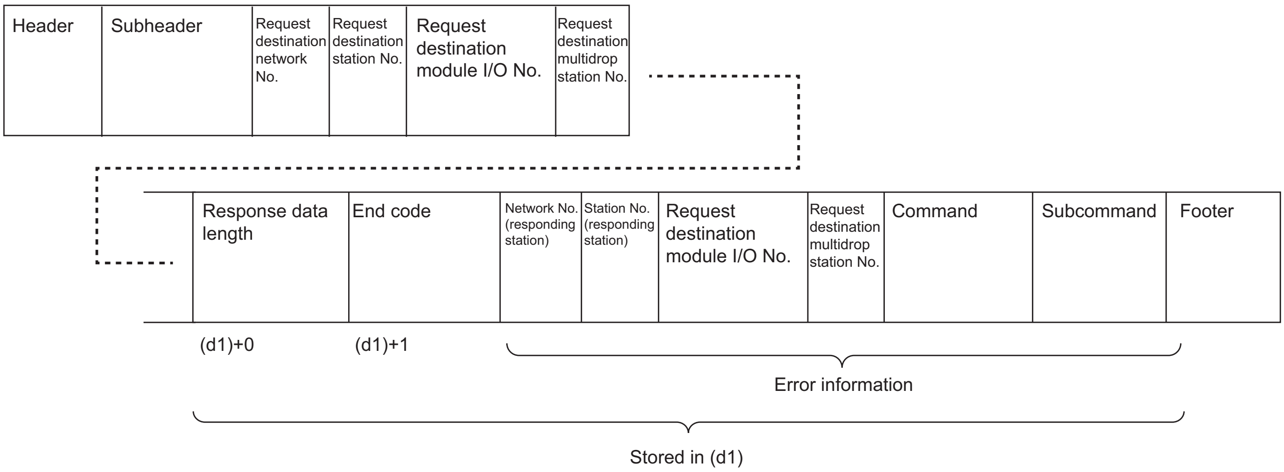


●Response data

(When completed)



(When failed)



- The SP.SLMPSND instruction communicates using UDP. Set the external device to use UDP.
- The SP.SLMPSND instruction communicates in binary code. Match the setting of the external device also with the binary code.
- The execution status and the completion status of the SP.SLMPSND instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)+0

This device turns on during END processing of the scan where the SP.SLMPSND instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the SP.SLMPSND instruction.

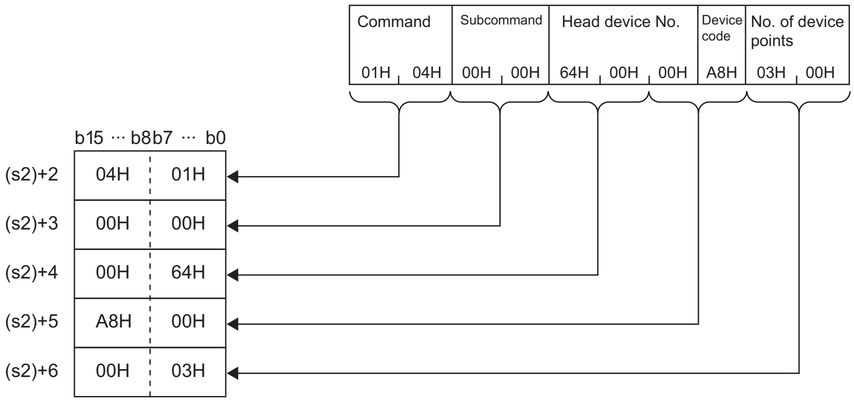
When completed successfully: The device remains off.

When completed with an error: The device turns on during the END processing of the scan where the SP.SLMPSND instruction completes, and turns off during the next END processing.

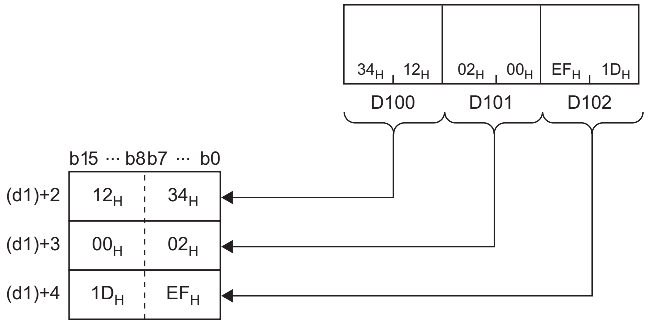
Ex.

Sending "Read (command: 0401H)" (reading in units of words) which reads the value in D100 to D102

●Request data

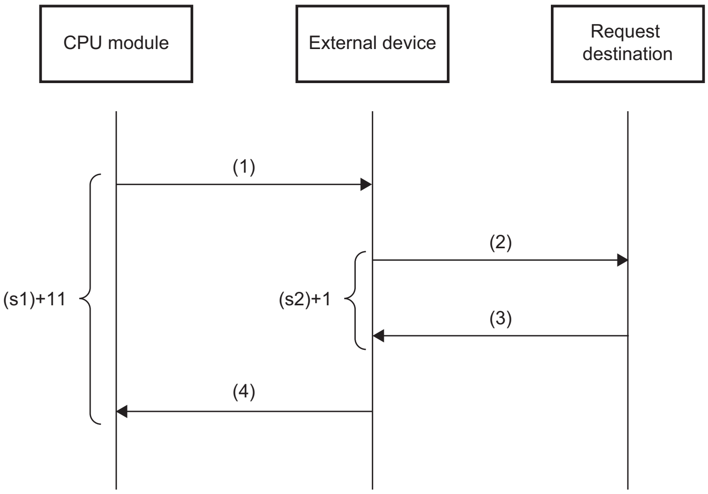


●Response data



Precautions

- When executing multiple SP.SLMPSND instructions concurrently, be careful not to overlap the channels of the SP.SLMPSND instructions. Multiple SP.SLMPSND instructions specifying the same channels cannot be used concurrently. When the execution conditions of the SP/SLMPSND instructions in the same channel are satisfied in the same sequence scan, only the SP/SLMPSND instruction that has been executed first is enabled and the subsequent SP/SLMPSND instructions are not executed. In addition, any subsequent SP.SLMPSND instruction of the same channel setting as the SP.SLMPSND instruction being executed is not executed. If the CPU module does not execute the processing of the SP.SLMPSND instruction, SM699 turns on.
- Specify the arrival monitoring time ((S1)+11) of the control data and monitoring timer ((S2)+1) of the request frame so that the arrival monitoring time≥monitoring timer.



(1) Request message
(2) Processing request from external device to request destination
(3) Processing response from request destination to external device
(4) Response message

Point

The SP.SLMPSND instruction is successfully completed even if the target device returns an abnormal response. When the SP.SLMPSND instruction is completed successfully, the response is whether normal or abnormal can be identified by the end code of the response frame. When an abnormal response is returned, check the manual of the SLMP-compatible device being used and take corrective action.

Operation error

Error code (SD0)	Description
3405H	The value set to (s1)+2 as own station channel is out of the range, 1 to 9.
	The value set to (s2)+0 as the request data length is 0 or exceeds 2000.

Upon completion with an error, the completion status indication device (d2)+1 is turned on and an error code is stored in the completion status (s1)+1. For the error code stored in the completion status (s1)+1, refer to the following.
🕮 MELSEC iQ-R Ethernet User's Manual (Application)

8.5 File Transfer Function Instructions

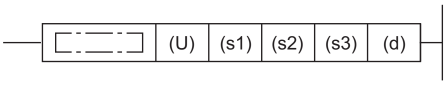
Sending FTP client files

SP.FTPPUT

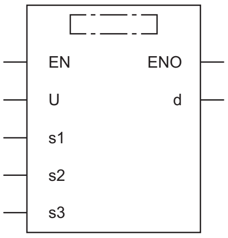
RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "22" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)

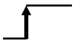
This instruction sends files in the CPU module, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).

Ladder	ST
	ENO:=SP_FTPPUT(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FTPPUT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 4)
(s2)	Name of files stored in the CPU module (transfer source)[1]	—	Unicode string[2]	ANYSTRING_DOUBLE
(s3)	Folder path of the FTP server (transfer destination)[1]	—	Unicode string[2]	ANYSTRING_DOUBLE
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Unicode string or the start device where the Unicode string is stored

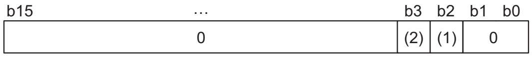
[2] Even though the data type is Unicode string, only one-byte alphanumeric characters, symbols, and kana characters; and two-byte characters (Shift JIS codes) can be used. Unsupported characters are treated as "_".

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	 (1) Transfer completion file delete setting (b2) Specify whether to delete transfer completion files or not. • 0: Do not delete • 1: Delete (2) Temporary file create setting (b3) Specify whether to create a temporary file (FTPCLI_I.TMP) during the file transfer processing or not. Setting this bit to 0 prevents files from becoming undefined in the transfer destination when a cable is disconnected or power is shut off during the file transfer processing. • 0: Create • 1: Do not create	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Total number of files to be transferred	The total number of files to be transferred by the SP.FTPPUT instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

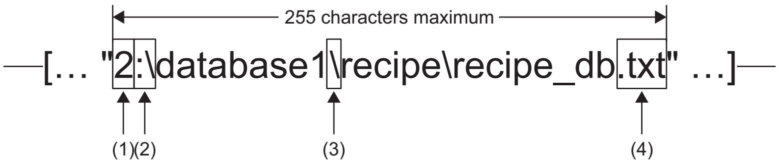
Processing details

- This instruction sends files in the CPU module, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3). The CPU module opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of the instruction, and closes a connection after sending files. For details on the parameter setting, refer to the following.

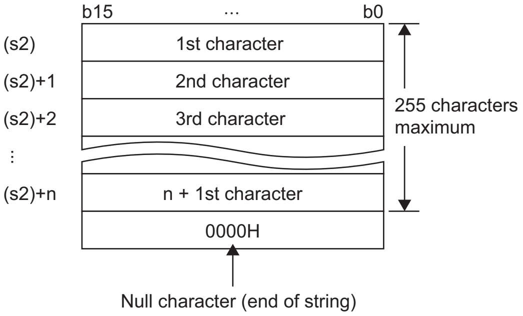
📖 MELSEC iQ-R Ethernet User's Manual (Application)

- The total number of files to be transferred by the SP.FTPPUT instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3.

- Specify the transfer source drive number (2 to 4) of the CPU module, the folder path where the files are stored, and the file name (including an extension) in (s2) in Unicode string. The maximum number of characters used in a file path is 255. The maximum number of characters used in a path, excluding the file name, is 246 (not including a delimiter). Use one-byte '\' or '/' as a delimiter to specify the boundaries between the elements in a file path.



(1)Drive numbers that can be specified are 2 to 4.
(2)Use one-byte ':\' or ':/' as a delimiter of the drive number.
(3)Use one-byte '\' or '/' as a delimiter of the folder path and file.
(4)The specified file name should include an extension.



- Wild card characters (*, ?) can be used in the file name or the extension specified in (s2).

Symbol	Description
*	An asterisk '*' is replaced with any character or string (including none) in a file name.
?	A question mark '?' is replaced with a character (excluding none) in a file name. ('?' can be used multiple times.)

Wild card characters do not recognize periods.

Using wild card characters in the following ways results in an error.

- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

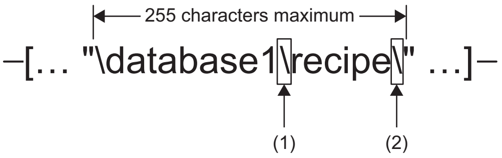
When any wild card character is used, the number of files that can be transferred is determined by the total size of the file names of the specified files. The specified files can be transferred when the number of these files and the total size of the file names satisfy the following condition. If a file transfer function instruction is executed without satisfying the following condition, the instruction completes with an error.

$(Fi + NM) + 1 < 65536$ [bytes]

N: Total number of files that match the wild card specification
Fi: Total size of the file names that match the wild card specification
M: Specific information size (Fixed value: 6 bytes)

- If an error occurs in any one of the files to be transferred during execution of the SP.FTPPUT instruction, the transfer processing will be stopped upon detection of the error, and the rest of target files will not be transferred.
- Untransferable files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file path which includes a file name and an extension exceeds 255, files will not be transferred even though the wild card specification conditions are satisfied.

- Specify the folder path of the transfer destination FTP server in (s3) in Unicode string. The specified folder path shall be a relative path from home directory of the FTP server. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path.[1] The maximum number of characters used in the folder path is 255. Note that the total number of characters in a folder path (including the delimiter at the end) and the file name specified in (s2) must be within the maximum path length supported by the transfer destination FTP server. The delimiter at the end of a string can be omitted. When omitted, '\' is assumed to be set at the end. If a nonexistent folder path is specified, a folder is automatically created by the system at execution of the instruction, and then the processing is performed.



(1)Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path.[1]
(2)The delimiter at the end of string can be omitted.

*1 Note that '\' cannot be used as a delimiter for some FTP servers.

- If a NULL character is specified by (s3) or only "0000H" is specified for the device, the CPU module directly accesses under the home directory of the FTP server. For details, follow the FTP server specifications.
- If a file with the same name exists in the transfer destination, the file will be overwritten.
- The maximum size of a file that can be send is 4G bytes.
- The execution status and the completion status of the SP.FTPPUT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

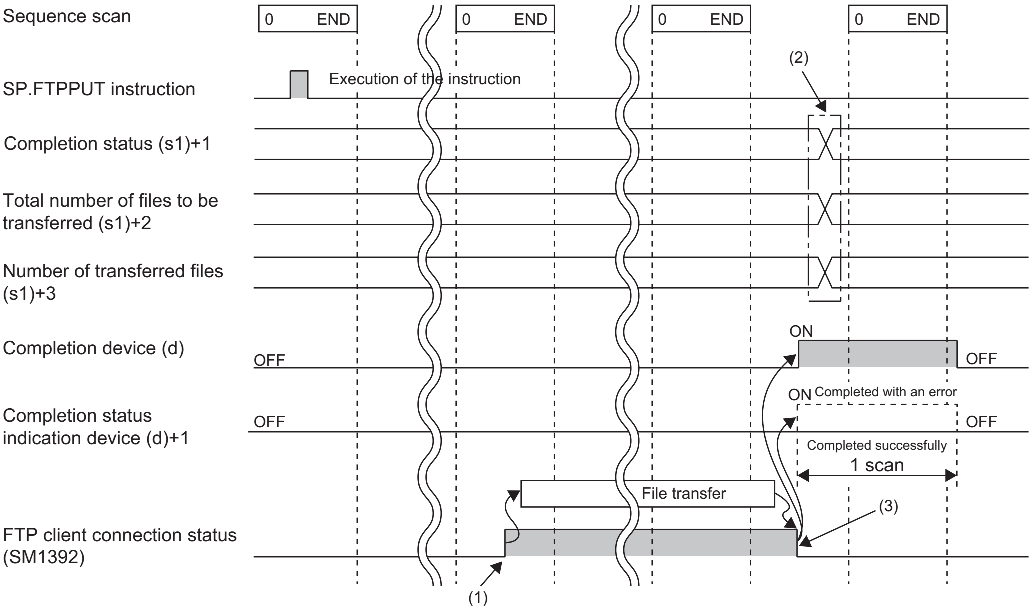
- Completion device (d)

This device turns on during END processing of the scan where the SP.FTPPUT instruction completes, and turns off during the next END processing.

- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.FTPPUT instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the SP.FTPPUT instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s1)+1.

- The following figure shows the execution timing of the SP.FTPPUT instruction.



(1)SM1392 turns on during the END processing after the CPU module is connected to the FTP server.
(2)Values are stored upon completion of the instruction.
(3)When all files have been transferred, SM1392 turns off.

- SM1392 (FTP client connection status) is on while the CPU module is connected to the FTP server, and SM1392 turns off when disconnected.
- SM753 (File access in progress) turns on while the SP.FTPPUT instruction is being executed. While SM753 is on, the SP.FTPPUT instruction cannot be executed. If executed, no processing is performed.
- If the SP.FTPPUT instruction is executed while the SP.FTPPUT or SP.FTPGET instruction is being executed, the instruction is ignored and not executed until the currently executing instruction completes. When the instruction is ignored, SM699 (Dedicated instruction skip flag) turns on.

- In the following cases, the instruction completes with an error: when there is no free space in the transfer destination; or when b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), but there is not enough free space for storing transfer-target files and a temporary file (same size as transfer-target files) in the transfer destination.
- Even though the operating status of the CPU module is switched from RUN to STOP during the file transfer processing, the CPU module continues the processing until completed.

Precautions

- If a cable is disconnected, power is shut off, or the CPU module is reset during the file transfer processing, delete unnecessary files (such as a temporary file and undefined files) on the FTP server as needed. Then, transfer files again.
- When b2 (Transfer completion file delete setting) of (s1)+0 is set to 1 (Delete), note the following.

Item	Description
When files in the CPU module are transferred	Files required for the CPU module to operate are also deleted. If deleted, operations of the CPU module cannot be guaranteed.
When wild card characters are used to specify a file name	Required files may be deleted unintentionally.

- When transfer source file access is restricted by the file password function, the SP.FTPPUT instruction completes with an error. Note, however, that the instruction can be executed if b2 (Transfer completion file delete setting) of (s1)+0 is set to 0 (Do not delete) and the file password setting type is "Write Protection".
- When b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), a temporary file of 12 characters (FTPCLI_I.TMP) will be created in the transfer destination. Therefore, set the folder path so that the total number of characters in the folder path and the temporary file does not exceed the maximum path length supported by the FTP server.
- Do not use any unsupported characters. If an unsupported character is included in the file name or the folder name specified by (s2) and (s3), the character is converted to "_" and processed. When wild card characters are used and an unsupported character is included in the name of files stored in the transfer source CPU module, the character is converted to "_" and processed. For this reason, the corresponding file is transferred in the same way as files having the same file name or folder name after character conversion are transferred.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in each setting area in the device/label memory in device specified by (s2), (s3) and later.
3405H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2), excluding the file name, exceeds 246 (not including a delimiter).
	The total number of characters in the strings specified by (s2) (only the file name part, excluding the drive number and folder path) and (s3) exceeds 255.
	The drive number specified by (s2) is out of range.
3426H	A file name is not specified by (s2).
	The file name that cannot be transferred is specified by (s2).
	The delimiter used to separate the drive number in (s2) is neither ':\' nor ':/'.
	Two or more asterisks '*' are used in the file name (before the period) or the extension specified by (s2).
	An asterisk '*' and a question mark '?' are used in the file name (before the period) or the extension specified by (s2).
	Wild card characters, '*' and '?', are used in the string specified by (s3).
3430H	The SP.FTPPUT instruction was executed without setting FTP client parameters.

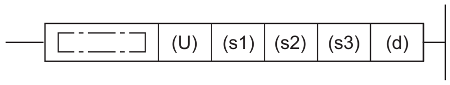
Retrieving FTP client files

SP.FTPGET

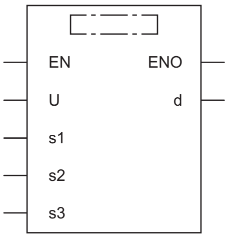
RnCPU RnENCPU ✗RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "22" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)

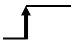
This instruction retrieves files on the FTP server, which are specified by (s2), to the folder path of the CPU module, which is specified by (s3).

Ladder	ST
	ENO:=SP_FTPGET(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FTPGET	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 4)
(s2)	Name of files stored in the FTP server (transfer source)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(s3)	Folder path of the CPU module (transfer destination)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Unicode string or the start device where the Unicode string is stored

*2 Even though the data type is Unicode string, only one-byte alphanumeric characters, symbols, and kana characters; and two-byte characters (Shift JIS codes) can be used. Unsupported characters are treated as "_".

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)

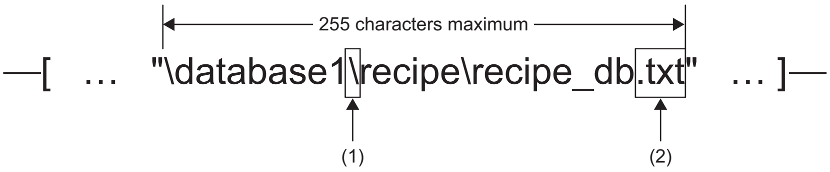
Device	Item	Description	Setting range	Set by
+0	Application setting area	b15 … b3 b2 b1 b0 0 | (2) | (1) | 0 (1) Transfer completion file delete setting (b2) Specify whether to delete transfer completion files or not. • 0: Do not delete • 1: Delete (2) Temporary file create setting (b3) Specify whether to create a temporary file (FTPCLI_I.TMP) during the file transfer processing or not. Setting this bit to 0 prevents files from becoming undefined in the transfer destination when a cable is disconnected or power is shut off during the file transfer processing. • 0: Create • 1: Do not create	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Total number of files to be transferred	The total number of files to be transferred by the SP.FTPGET instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

Processing details

- This instruction retrieves files on the FTP server, which are specified by (s2), to the folder path of the CPU module, which is specified by (s3). The CPU module opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of the instruction, and closes a connection after retrieving files. For details on the parameter setting, refer to the following.

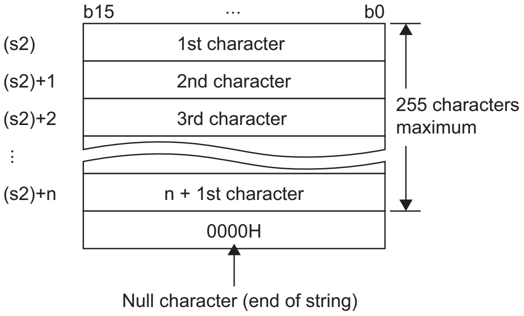
MELSEC iQ-R Ethernet User's Manual (Application)

- The total number of files to be transferred by the SP.FTPGET instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3.
- Specify the folder path where the transfer source files are stored on the FTP server, and the file name (including an extension) in (s2) in Unicode string. The maximum number of characters used in a file path is 255. The specified file path shall be a relative path from home directory of the FTP server. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path or the file name.[*1]



(1)Use one-byte '\' or '/' as a delimiter of the folder path or file.[*1]
(2)The specified file name should include an extension.

*1 Note that '\' cannot be used as a delimiter for some FTP servers.



- Wild card characters (*, ?) can be used in the file name or the extension specified in (s2).

Symbol	Description
*	An asterisk '*' is replaced with any character or string (including none) in a file name.
?	A question mark '?' is replaced with a character (excluding none) in a file name. ('?' can be used multiple times.)

Wild card characters do not recognize periods.

Using wild card characters in the following ways results in an error.

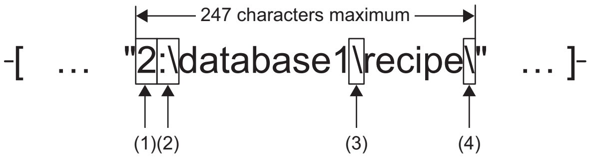
- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

When any wild card character is used, the number of files that can be transferred is determined by the total size of the file names of the specified files. The specified files can be transferred when the number of these files and the total size of the file names satisfy the following condition. If a file transfer function instruction is executed without satisfying the following condition, the instruction completes with an error.

$(Fi + NM) + 1 < 65536$ [bytes]

N: Total number of files that match the wild card specification
Fi: Total size of the file names that match the wild card specification
M: Specific information size (Fixed value: 6 bytes)

- If an error occurs in any one of the files to be retrieved during execution of the SP.FTPGET instruction, the transfer processing will be stopped upon detection of the error, and the rest of target files will not be transferred.
- Untransferable files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file path which includes a file name and an extension exceeds 255, files will not be transferred even though the wild card specification conditions are satisfied.
- If only a file name is specified by (s2), the CPU module directly accesses under the home directory of the FTP server. When specifying only a file name, start with a delimiter. For details, follow the FTP server specifications.
- Specify the folder path of the transfer destination CPU module in (s3) in Unicode string. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path. The maximum number of characters in the folder path is 247 (246 when a delimiter at the end of the string is omitted). Note that the total number of characters in a folder path (including the delimiter at the end) and the file name specified in (s2) must be within the maximum path length (255 characters) supported by the CPU module. The delimiter at the end of a string can be omitted. When omitted, '\' is assumed to be set at the end. If a nonexistent folder path is specified, a folder is automatically created by the system at execution of the instruction, and then the processing is performed.



(1)Drive numbers that can be specified are 2 to 4.
(2)Use one-byte ':\' or ':/' as a delimiter of the drive number.
(3)Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path.
(4)The delimiter at the end of string can be omitted.

- If a file with the same name exists in the transfer destination, the file will be overwritten.
- The maximum size of a file that can be retrieved is 4G bytes.
- The execution status and the completion status of the SP.FTPGET instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the SP.FTPGET instruction completes, and turns off during the next END processing.

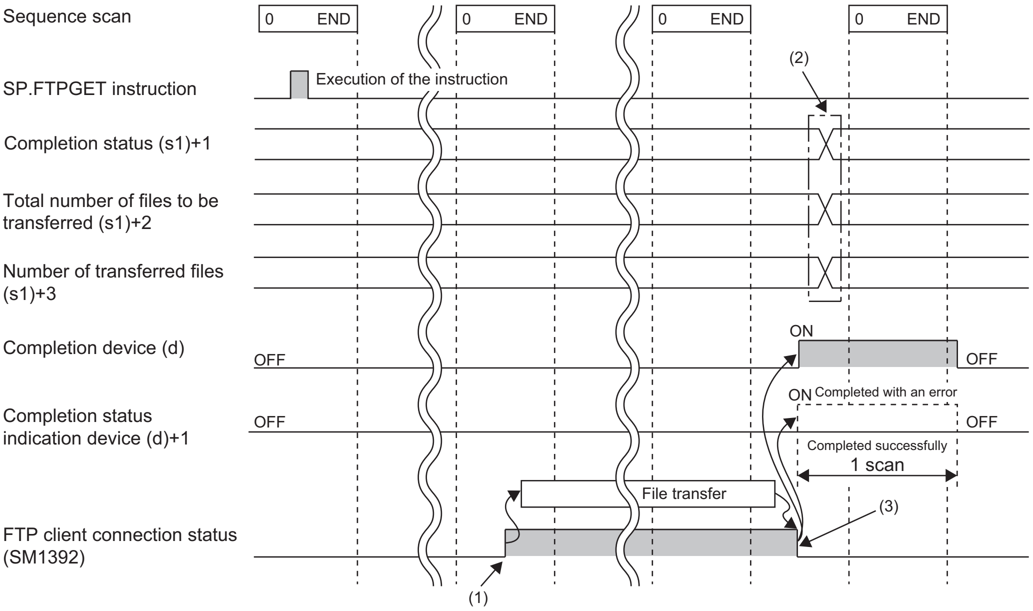
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.FTPGET instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.FTPGET instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s1)+1.

- The following figure shows the execution timing of the SP.FTPGET instruction.



(1)SM1392 turns on during the END processing after the CPU module is connected to the FTP server.

(2)Values are stored upon completion of the instruction.

(3)When all files have been transferred, SM1392 turns off.

- SM1392 (FTP client connection status) is on while the CPU module is connected to the FTP server, and SM1392 turns off when disconnected.
- SM753 (File access in progress) turns on while the SP.FTPGET instruction is being executed. While SM753 is on, the SP.FTPGET instruction cannot be executed. If executed, no processing is performed.
- If the SP.FTPGET instruction is executed while the SP.FTPPUT or SP.FTPGET instruction is being executed, the instruction is ignored and not executed until the currently executing instruction completes. When the instruction is ignored, SM699 (Dedicated instruction skip flag) turns on.
- In the following cases, the instruction completes with an error: when there is no free space in the transfer destination; or when b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), but there is not enough free space for storing transfer-target files and a temporary file (same size as transfer-target files) in the transfer destination.
- Even though the operating status of the CPU module is switched from RUN to STOP during the file transfer processing, the CPU module continues the processing until completed.

Precautions

- If a cable is disconnected, power is shut off, or the CPU module is reset during the file transfer processing, transfer files again. In those cases, unnecessary files will be deleted in the following ways.

Unnecessary file	Delete operation
Temporary file (FTPCLI_I.TMP)	A temporary file left in the FTP client will be automatically deleted at the start of next file transfer processing to the same folder.
Undefined transfer target file	An undefined transfer target file left in the FTP client will be overwritten during next file transfer processing to the same file.

- When a file with the same name has already existed in the transfer destination and its access is restricted by the file password function, the SP.FTPGET instruction completes with an error.
- When b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), set the maximum number of characters in the folder path of the CPU module to 243 (242 when a delimiter at the end of the string is omitted). Since a temporary file of 12 characters (FTPCLI_I.TMP) is created in the transfer destination, set the folder path so that the total number of characters in the folder path and the temporary file does not exceed the maximum path length (255 characters) supported by the CPU module.
- Do not use any unsupported characters. If an unsupported character is included in the file name or the folder name specified by (s2) and (s3), the character is converted to "_" and processed. For this reason, the corresponding file is transferred in the same way as files having the same file name or folder name after character conversion are transferred. When wild card characters are used and an unsupported character is included in the name of files stored in the transfer source FTP server, files cannot be retrieved properly. (How to treat unsupported characters depends on the specifications of the FTP server.)

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in each setting area in the device/label memory in device specified by (s2), (s3) and later.
3405H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2), excluding the file name, exceeds 246 (not including a delimiter).
	The total number of characters in the strings specified by (s3) and (s2) (only the file name part) exceeds 255.
	The drive number specified by (s3) is out of range.
3426H	A file name is not specified by (s2).
	The file name that cannot be transferred is specified by (s2).
	The delimiter used to separate the drive number in (s3) is neither ':\' nor ':/'.
	Two or more asterisks '*' are used in the file name (before the period) or the extension specified by (s2).
	An asterisk '*' and a question mark '?' are used in the file name (before the period) or the extension specified by (s2).
	Wild card characters, '*' and '?', are used in the string specified by (s3).
3430H	The SP.FTPGET instruction was executed without setting FTP client parameters.

9 PID OPERATION INSTRUCTION

PID operation instruction and PID control instruction

There are two types of instructions for PID control.

Type	Application	Reference (Overview)	Reference (Details)
PID operation instruction	This instruction is used to perform PID control using auto tuning.	Page 1134 Overview	Page 1145 PID Operation Instruction
PID control instruction	This instruction is used to perform PID control equal to that of the MELSEC-Q series and MELSEC-L series.	Page 1149 Overview	Page 1158 PID Control Instructions (Inexact Differential),Page 1170 PID Control Instructions (Exact Differential)

The following table lists the specifications comparison between PID operation instruction and PID control instruction.

Item	PID operation instruction	PID control instruction
PID operation method	Inexact differential	Inexact differential/exact differential
Sampling period/sampling time	1 to 32767ms	10 to 60000ms
Number of control loops	1 loop per instruction	32 loops maximum
Auto tuning	Enabled (limit cycle method and step response method)	Disabled

9.1 Overview

This section describes an overview of PID control using the PID operation instruction.

PID operation instruction

The PID operation instruction calculates the manipulated value (MV) from the process value (PV) by combining the P action (proposal action), I action (integral action), and D action (derivative action) to get closer to the set value (SV).

■Alarm output function

This function turns on the alarm output for the variations of input (process value) and output (manipulated value).

■Output upper/lower limit value setting

This function suppresses the increase of integral terms in PID control by setting the output upper/lower limit values.

■Auto tuning function

This function automatically sets the proportional gain (K_P), integral time (T_I), and derivative time (T_D). The auto tuning is performed in two methods: limit cycle method and step response method.

■Operation method of the PID operation instruction

The instruction performs PID operation in the velocity type or the process value derivative type.

Basic operational expressions of PID operation instruction [Reference]

The instruction performs PID operation in the velocity type or the process value derivative type.

An operational expression of direct action or reverse action is executed depending on the value of bit 0 in the device specified by (s3)+1 (Action setting (ACT)).

The operation is performed using the control data stored in the device areas, (s3) and later.

- Operational expressions

Action (Bit 0 of (s3)+1)	Operational expression
Direct action (Off)	$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} EV_n + D_n\}$ $EV_n = PV_{nf} - SV$ $D_n = \frac{T_D}{T_S + K_D \cdot T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$
Reverse action (On)	$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} EV_n + D_n\}$ $EV_n = SV - PV_{nf}$ $D_n = \frac{T_D}{T_S + K_D \cdot T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{nf}	Process value of the sampling period this time (after filtering)
PV_{nf-1}	Process value of the sampling period last time (after filtering)
PV_{nf-2}	Process value of the sampling period two times before (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
D_{n-1}	Derivative term of the sampling period last time
K_P	Proportional gain
T_S	Sampling period
T_I	Integral constant
T_D	Derivative constant
K_D	Derivative gain

PV_nf (process value of the sampling period this time (after filtering)) is calculated by using the following operational expression. If the input filter coefficient is not set, the value will be the same as the input process value (PV).

$PV_{nf} = PV_n + L(PV_{nf-1} - PV_n)$

Where, PV_nf: Process value for the sampling period this time, L: Filter coefficient, PV_{nf-1}: Process value for the sampling period last time (after filtering)

Control data

The details on the control data used by the PID operation instruction are described.

Sampling time: (s3)

Setting range: 1 to 32767 [ms]

Set a cycle (ms) to perform PID operation.

- Auto tuning (limit cycle method)

Set a cycle so that the following condition is satisfied: Operation cycle of the programmable controller < Sampling time

- Auto tuning (step response method)

Set a cycle to 1000ms or longer.

■Maximum error

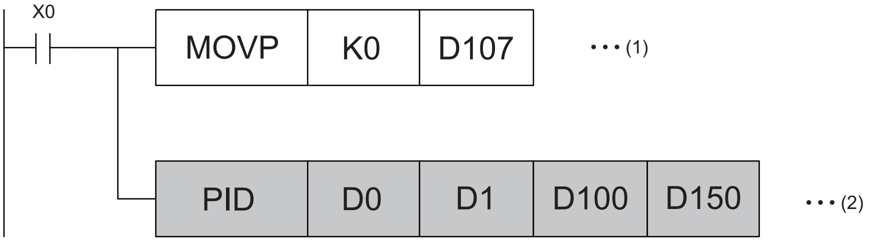
The maximum error of the sampling time (T_S) is -(1 operation cycle + 1ms) to +(1 operation cycle).

- When the sampling time (T_S) value is small

The variation in the maximum error may become a problem. Set the constant scan and execute the instruction, or program it in the timer interrupt routine.

- When the sampling time value is shorter than one operation cycle of the programmable controller

A PID operation error (11A6H) occurs, but PID operation is executed assuming that the sampling time (T_S) equals to the operation cycle. In this case, program the instruction in the timer interrupt routine, clear the value in (s3)+7, and then execute the instruction.



(1) Reset the value in (s3)+7. (The internal processing register is cleared by the pulse conversion command at the first execution of the interrupt routine.)
(2) Execute the PID operation.

Action setting: (s3)+1

■Direction (direct action/reverse action): Bit 0 of (s3)+1

Setting range: Off = Direct action, On = Reverse action

Select the direction of PID control, direction action or reverse action.

- Auto tuning (limit cycle method)

The PID control direction must be set.

- Auto tuning (step response method)

At completion of auto tuning executed in whichever mode, direct action or reverse action, setting is made automatically.

[Direct action (bit 0 of (s3)+1 = Off)]

[Reverse action (bit 0 of (s3)+1 = On)]

■Alarm setting (input variation, output variation): Bit 1 and bit 2 of (s3)+1

Setting range: Off = Alarm disabled, On = Alarm enabled

The input and output variation amounts can be checked. The check result can be checked in (s3)+24. (☞ Page 1141 Alarm output flag: (s3)+24)

- Input variation (bit 1 of (s3)+1)

To use the input variation alarm, the following bit needs to be on and the values need to be set to the following devices.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 1	Input variation alarm	On: Enabled Off: Disabled
Input variation alarm setting value	(s3)+20		Input variation (increase) alarm setting value	0 to 32767
	(s3)+21		Input variation (decrease) alarm setting value	

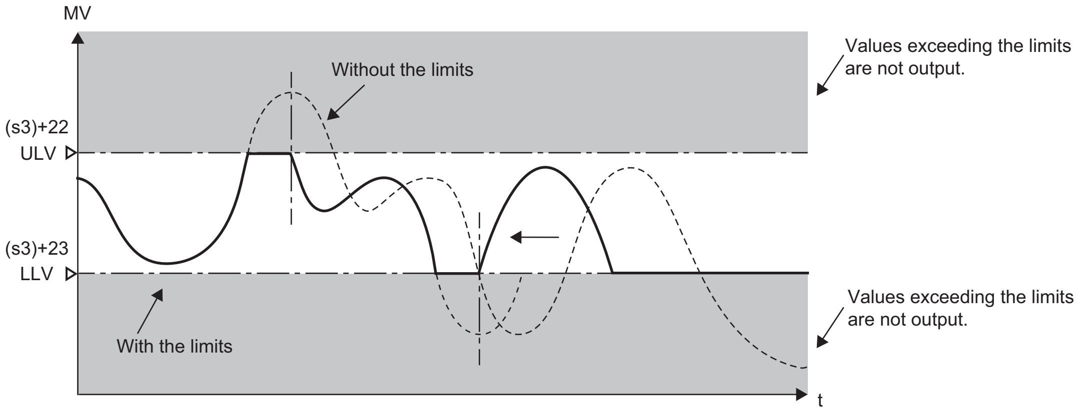
- Output variation (bit 2 of (s3)+1)

To use the output variation alarm, the following bits need to be on and the values need to be set to the following devices.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 2	Output variation alarm	On: Enabled Off: Disabled
		Bit 5	Output upper/lower limit value setting	Off (always)
Output variation alarm setting value	(s3)+22		Output variation (increase) alarm setting value	0 to 32767
	(s3)+23		Output variation (decrease) alarm setting value	

■Output upper/lower limit value setting: Bit 5 of (s3)+1

The manipulated value (MV) will be as follows according to this setting.



MV: Manipulated value
ULV: Output upper limit value
LLV: Output lower limit value
t: Time

This setting suppresses the increase of integral terms in PID control. To use this function, turn off the bit 2 of (s3)+1.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 2	Output variation alarm	Off (always)
		Bit 5	Output upper/lower limit value setting	On: Enabled Off: Disabled

Input filter: (s3)+2

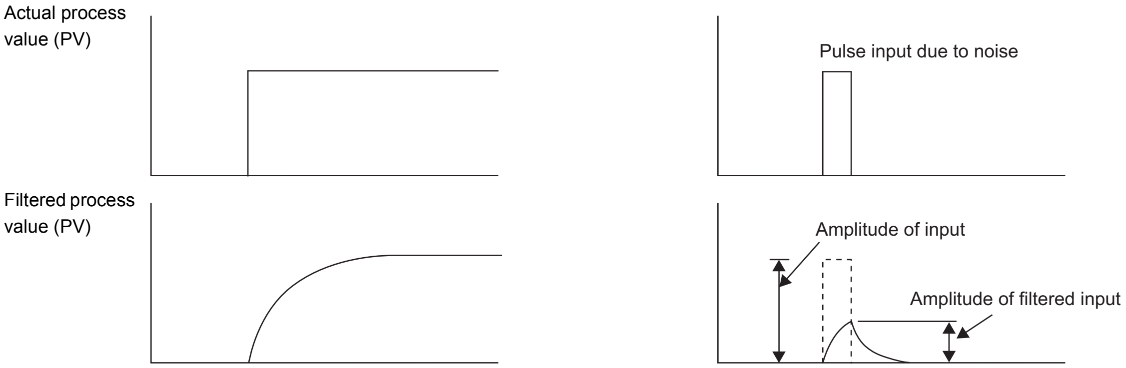
Setting range: 0 to 99 [%]

PID control: Proportional action, integral action, derivative action

The input filter (α) is a software filter used to reduce the variations caused by noise in the process value (PV). The influence of noise can be suppressed by setting the input filter (α) properly according to the characteristics and noise level of the control target.

- If the filter coefficient is too small, the effect will be reduced.
- If the filter coefficient is too large, the input response will deteriorate.

The input filter (α) acts on the set value (SV) and thus affects the proportional action, integral action, and derivative action.



Proportional gain: (s3)+3

Setting range: 1 to 32767 [%]

PID control: Proportional action

The manipulated value (MV) increases in proportion to the deviation (difference between the set value (SV) and the process value (PV)) in proportional operation. This ratio is called the proportional gain (K_P) and represented by the following relational expression.

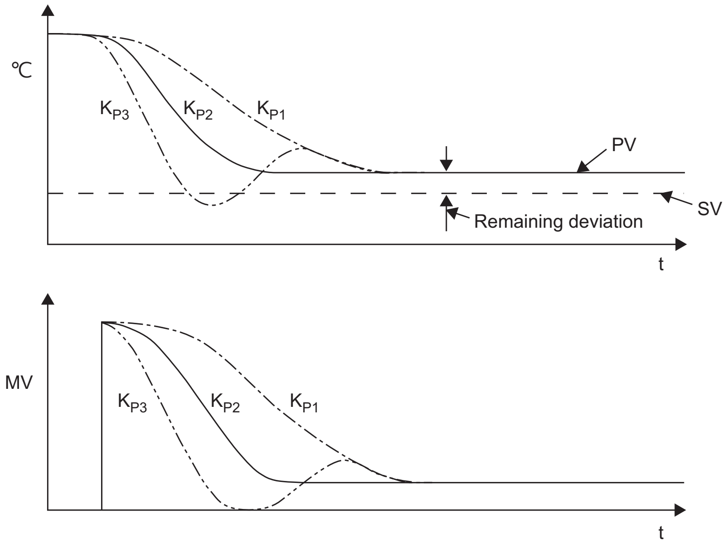
Manipulated value (MV) = Proportional gain (K_P) × Deviation (EV)

The reciprocal of the proportional gain (K_P) is called the proportional band.

As the proportional gain (K_P) increases, the motion to get the process value (PV) closer to the set value (SV) becomes strong.

Ex.

Proportional action (P action) in the case of cooling (direct action)



Proportional gain (K_P): $K_{P3} > K_{P2} > K_{P1}$

℃: Temperature

SV: Set value

PV: Process value

MV: Manipulated value

t: Time

Integral time: (s3)+4

Setting range: 0 to 32767 [× 100ms] (0 = ∞) (No integration)

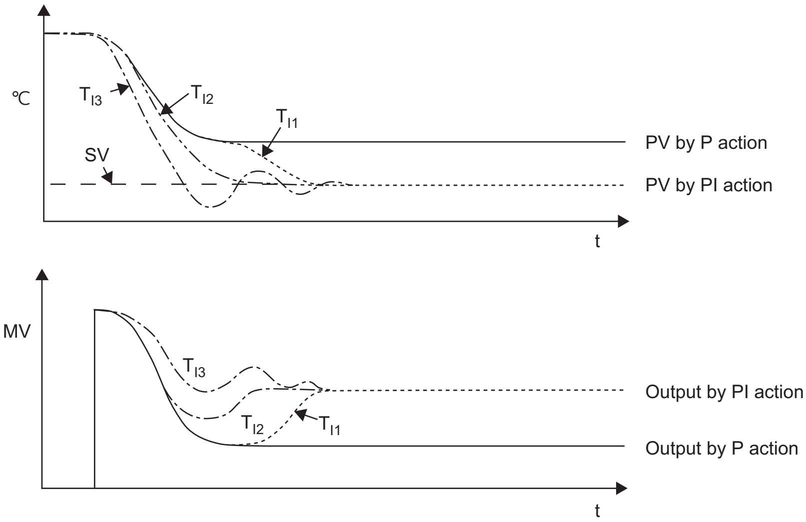
PID control: Integral action

The integral time (T_I) is the time from when an deviation occurs in integral action to when the output of the integral action becomes the output of proportional action.

Reducing the integral time (T_I) accelerates the integral operation.

Ex.

PI action in the case of cooling (direct action)



Integral time (T_I): $0 < T_{I3} < T_{I2} < T_{I1}$

℃: Temperature

SV: Set value

MV: Manipulated value

t: Time

Derivative gain: (s3)+5

Setting range: 0 to 200 [%]

PID control: Derivative action

The output of the derivative action is filtered. The derivative gain (K_D) affects only the derivative action.

- If the derivative gain (K_D) is decreased, the output responds instantaneously to a change in the process value (PV) caused by a disturbance.
- If the derivative gain (K_D) is increased, the output takes time to respond to a change in the process value (PV) caused by a disturbance.

Point

First, set the derivative gain (K_D) to 0 and adjust it using the input filter (α). If the change in the output responds too sensitive to the disturbance, increase the value.

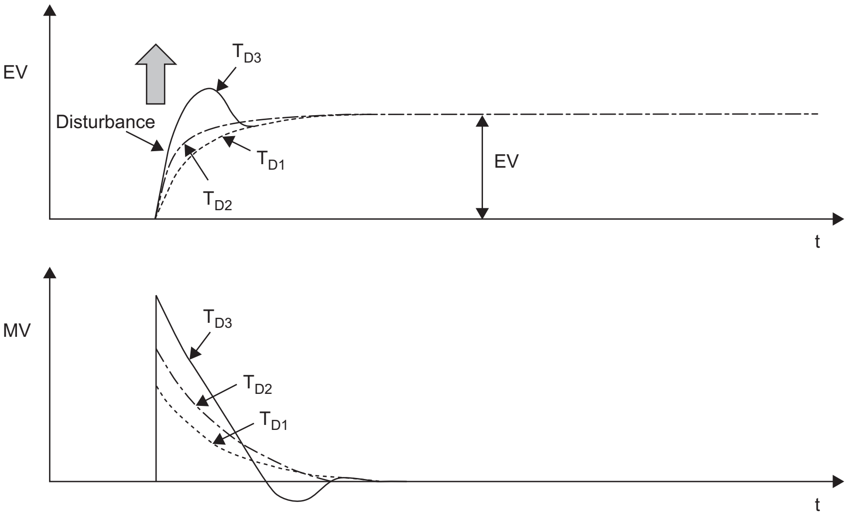
Derivative time (T_D): (s3)+6

Setting range: 0 to 32767 [× 10ms] (0 = no derivation)

PID control: Derivative action

The derivative time is used to be sensitive to the change in the process value (PV) caused by a disturbance and minimize the changes.

- Increasing the derivative time (T_D) prevents more positively the control target from fluctuating due to a disturbance.



Derivative time (T_D): $T_{D3} > T_{D2} > T_{D1}$

EV: Deviation

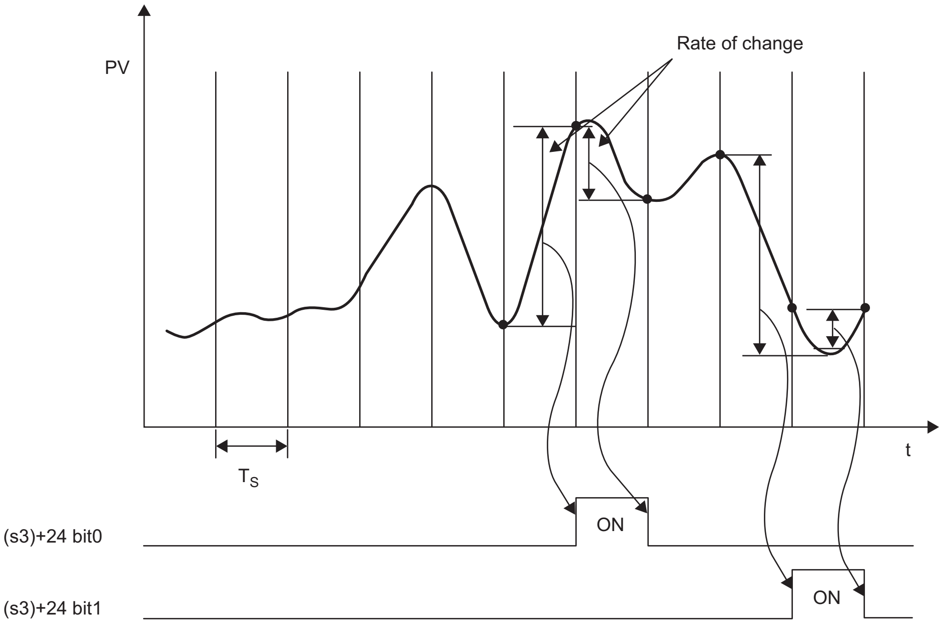
MV: Manipulated value

t: Time

Alarm output flag: (s3)+24

When the set input/output variation is exceeded, bits of (s3)+24 turn on as an alarm flag immediately after execution of the PID operation instruction.

- When the input variation alarm (bit 1 of (s3)+1) is on



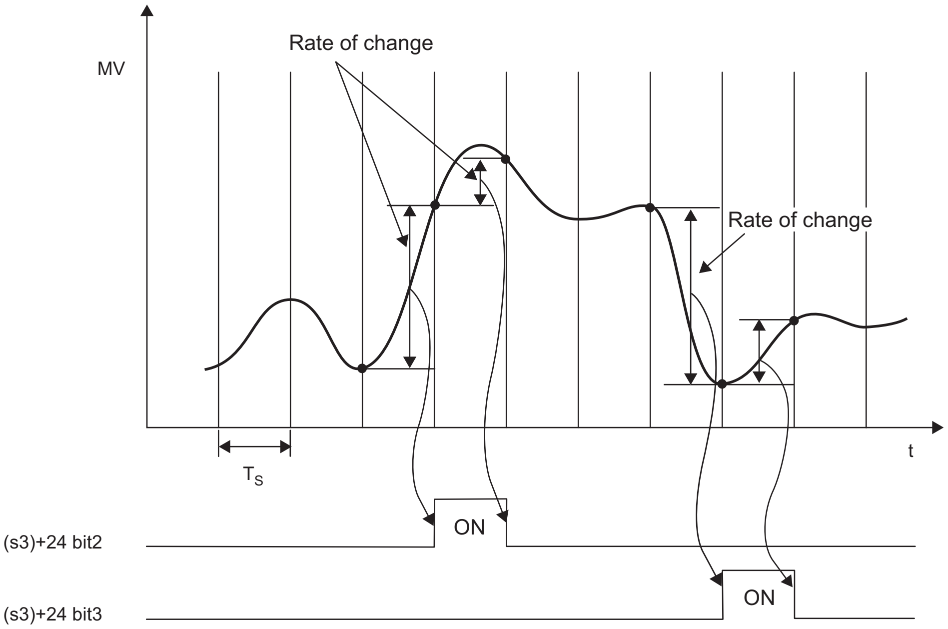
PV: Process value

T_S: Sampling time

t: Time

Bit 0 and bit 1 of (s3)+24: Alarm flag

- When the output variation alarm (bit 2 of (s3)+1) is on



MV: Manipulated value
T_S: Sampling time
t: Time
Bit 2 and bit 3 of (s3)+24: Alarm flag

Auto tuning

To obtain satisfactory results in PID control, it is required to determine the optimum values of each constant (control data) appropriate for the control target. The auto tuning function automatically sets the important constants for PID control, proportional gain, integral time, and derivative time.

The auto tuning function is performed in two methods: limit cycle method and step response method.

Point

Start auto tuning after the system becomes stable. If not, auto tuning may not be performed correctly.

Limit cycle method

For an overview of the limit cycle method, refer to the following.

☞ Page 2056 Overview of limit cycle method

■Control data set by auto tuning (limit cycle method)

- Proportional gain (K_P): (s3)+3
- Integral time (T_I): (s3)+4
- Derivative time (T_D): (s3)+6

■Procedure

1. Set the action of PID operation.

Set the action (direct action or reverse action) with the bit 0 of (s3)+1 (Action setting (ACT)).

2. Select the limit cycle method.

Turn on the bit 6 of (s3)+1 (Action setting (ACT)).

(When the bit is off, the step response method is selected.)

3. Turn on the auto tuning execution flag.

Turn on the bit 4 of (s3)+1 (Action setting (ACT)).

4. Set an input filter constant.

Set a value in (s3)+2 (Input filter (α)).

5. Set a sampling time.

Set a value in (s3)+0 (Sampling time (T_S)).

6. Set the output upper limit value (ULV).

Set the upper limit of the manipulated value (MV) in (s3)+26 (Output upper limit value (ULV)).

7. Set the output lower limit value (LLV).

Set the lower limit of the manipulated value (MV) in (s3)+27 (Output lower limit value (LLV)).

8. Set the PV value threshold (hysteresis) width (SHPV).

Set a value in (s3)+25 (PV value threshold (hysteresis) width (SHPV)).

9. Set a set value (SV).

Set a set value (SV) in (s1) of the PID operation instruction.

10. Start auto tuning.

Auto tuning starts based on the process value (PV) when the start contact device of the PID operation instruction turns on.

The auto tuning related flags (bit 4 and bit 6) of (s3)+1 (Action setting (ACT)) turn off upon completion of auto tuning.

Step response method

For an overview of the step response method, refer to the following.

☞ Page 2057 Overview of step response method

■Control data set by auto tuning (step response method)

- Direction (direct action/reverse action): Bit 0 of (s3)+1
- Proportional gain (K_P): (s3)+3
- Integral time (T_I): (s3)+4
- Derivative time (T_D): (s3)+6

■Procedure

1. Set the output value for auto tuning.

Transfer the output value for auto tuning to the manipulated value (MV).

Set the output value for auto tuning to the "maximum allowable output value of the external device × 0.5 to 1".

2. Set the data that is not set by auto tuning.

Set the following items, which are not set by auto tuning, depending on the system.

Setting item		Remarks
(s1)	Set value (SV)	Set a value so that the difference from the process value (PV) becomes 150 or more.[1]
(s3)	Sampling time (T_S)	Set a cycle to 1000ms or longer.[2]
(s3)+2	Input filter (α)	—
(s3)+5	Derivative gain (K_D)	When the input filter is used, set the derivative gain to 0.
Others		Set other control data as needed.

[1] Difference between the set value (SV) and the process value (PV)
To perform auto tuning correctly, the difference between the process value (PV) and the set value (SV) must be 150 or more at the time of auto tuning start. If the difference is less than 150, set a set value (SV) for auto tuning.
After completion of auto tuning, set the SV back to the original value.

[2] Sampling time (T_S) setting
To perform auto tuning, set the sampling time (T_S) to 1000 ms or longer.
The sampling time should also be sufficiently longer than the output change period.

3. Start auto tuning.

Turn on the bit 4 of (s3)+1 (Action setting (ACT)) to start auto tuning.

When the amount of variation from the process value (PV) at start of auto tuning to the set value (SV) becomes 1/3 or more, auto tuning is completed and the bit 4 of (s3)+1 (Action setting (ACT)) turns off automatically.

Precautions

- Measures to be taken when the process value (PV) does not change

If the process value (PV) does not change properly due to an analog input disconnection or any other problems, auto tuning does not end. Detect and avoid such problems by creating a program that monitors the input value and the elapsed time from the start of auto tuning.

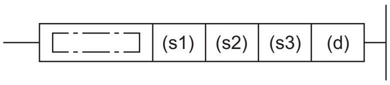
9.2 PID Operation Instruction

PID

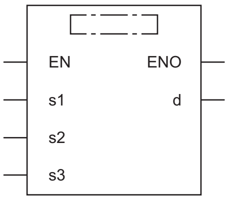
RnCPU **RnENCPU** ✗ RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

This instruction performs PID operation using the values set in (s1) to (s3), and stores the operation result in (d) at each cycle of sampling time.

Ladder	ST
	ENO:=PID(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PID	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the set value (SV) is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Device where the process value (PV) is stored	-32768 to 32767	16-bit signed binary	ANY16
(s3)	Start device where the control data are stored	—	16-bit signed binary	ANY16
(d)	Device for storing the manipulated value (MV)	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	—	—	○	○	—	—	—	○	—	—	—	—
(s2)	—	—	○	○	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—

Processing details

When the set value (s1), process value (s2), and control data (s3) to (s3)+6 are set and a program is executed, the operation result, manipulated value (MV), is stored to (d) at every cycle of sampling time. For details, refer to the following.

☞ Page 1134 Overview

■Setting items (arguments)

Setting item		Description	Number of occupied points
(s1)	Set value (SV)	Set the set value (SV). ■Auto tuning (limit cycle method) If the set value for auto tuning differs from that for normal PID control, set a value obtained by adding a bias value, and then set the SV back to the original value when the auto tuning flag turns off.	1 point
(s2)	Process value (PV)	Set the input value for PID operation.	1 point
(s3)	Control data[1]	■Auto tuning (limit cycle method) The device areas of 29 points starting from the device specified in (s3) are used.	29 points
		■Auto tuning (step response method) ❶ The device areas of 25 points starting from the device specified in (s3) are used when all of the bits 1, 2, and 5 of (s3)+1 (Action setting (ACT)) are not 0. ❷ The device areas of 20 points starting from the device specified in (s3) are occupied when all of the bits 1, 2, and 5 of (s3)+1 (Action setting (ACT)) are 0.	❶ 25 points ❷ 20 points
(d)	Manipulated value (MV)	■Normal PID control The user sets the initial manipulated value before execution of the instruction. After execution, the operation result is stored. ■Auto tuning (limit cycle method) During auto tuning, the output upper limit value (ULV) or output lower limit value (LLV) is output automatically. Upon completion of auto tuning, the given MV is set. ■Auto tuning (step response method) The user sets the step manipulated value before execution of the instruction. During auto tuning, the MV cannot be changed by the PID instruction.	1 point

[1] When auto tuning is not used, the same number of points are occupied as when the step response method is used.

■Setting items (control data)

Operand: (s3)

Device	Item		Description	Remarks
+0	Sampling time (T_S)		1 to 32767 [ms]	The sampling time must be longer than the operation cycle of the programmable controller.
+1	Action setting (ACT)	Bit 0	0: Direct action 1: Reverse action	Action direction specification
		Bit 1	0: Input variation alarm disabled 1: Input variation alarm enabled	—
		Bit 2	0: Output variation alarm disabled 1: Output variation alarm enabled	Do not turn on bit 2 and bit 5 at the same time.
		Bit 3	Reserved	—
		Bit 4	0: Auto tuning disabled 1: Auto tuning enabled	—
		Bit 5	0: No output upper/lower limit value setting 1: Output upper/lower limit value setting enabled	Do not turn on bit 2 and bit 5 at the same time.
		Bit 6	0: Step response method 1: Limit cycle method	Auto tuning mode selection
		Bits 7 to 15	Reserved	—
+2	Input filter constant (α)		0 to 99 [%]	0 = No input filter
+3	Proportional gain (K_P)		1 to 32767 [%]	—
+4	Integral time (T_I)		1 to 32767 [$\times$ 100ms]	0 = ∞ (No integration)
+5	Derivative gain (K_D)		0 to 200 [%]	0 = No derivative gain
+6	Derivative time (T_D)		1 to 32767 [$\times$ 10ms]	0 = No derivation
+7 to +19	These areas are used for internal processing of PID operation, and therefore data cannot be changed.			
+20[1]	Input variation (increase) alarm setting value		0 to 32767	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1.
+21[1]	Input variation (decrease) alarm setting value		0 to 32767	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1.
+22[1]	Output variation (increase) alarm setting value		0 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 1, and the bit 5 is 0.
	Output upper limit setting value		-32768 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 0, and the bit 5 is 1.

Operand: (s3)				
Device	**Item**		**Description**	**Remarks**
+23[*1]	Output variation (decrease) alarm setting value		0 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 1, and the bit 5 is 0.
	Output lower limit setting value		-32768 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 0, and the bit 5 is 1.
+24[*1]	Alarm output	Bit 0	0: Input variation (increase) not exceeded 1: Input variation (increase) exceeded	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1, or the bit 2 is 1.
		Bit 1	0: Input variation (decrease) not exceeded 1: Input variation (decrease) exceeded	—
		Bit 2	0: Output variation (increase) not exceeded 1: Output variation (increase) exceeded	—
		Bit 3	0: Output variation (decrease) not exceeded 1: Output variation (decrease) exceeded	—
+25[*2]	PV value threshold (hysteresis) width (SHPV)		Set according to the fluctuation of the process value (PV).	Occupied when the bit 6 of (s3)+1 (Action direction (ACT)) is 1 (limit cycle method).
+26[*2]	Output upper limit value (ULV)		Upper limit (ULV) of the manipulated value (MV)	
+27[*2]	Output lower limit value (LLV)		Lower limit (LLV) of the manipulated value (MV)	
+28[*2]	Wait time setting parameter from the end of tunning cycle to the start of PID control (K_W)		-5 to 32717 [%]	

*1 Occupied when the bit 1 of (s3)+1 (Action direction (ACT)) is 1, the bit 2 is 1, or the bit 5 is 1.

*2 Occupied when the bit 6 of (s3)+1 (Action direction (ACT)) is 1 (limit cycle method).

Operation error

Error code (SD0)	Description
11A0H	A value set for the sampling time (T_S) is out of the range, $T_S \leq 0$.
11A1H	A value set for the input filter constant (α) is out of the range, $\alpha < 0$ or $100 \leq \alpha$.
11A2H	A value the proportional gain (K_P) is out of the range, $K_P < 0$.
11A3H	A value set for the integral time (T_I) is out of the range, $T_I < 0$.
11A4H	A value set for the derivative gain (K_D) is out of the range, $K_D < 0$ or $201 \leq K_D$.
11A5H	A value set for the derivative time (T_D) is out of the range, $T_D < 0$.
11A6H	A value set for the sampling time (T_S) is less than the operation cycle of the programmable controller.
11A7H	The process value variation (ΔPV) overflowed.
11A8H	Deviation (EV) overflowed.
11A9H	The calculated integral value overflowed.
11AAH	The derivative gain (K_D) value overflowed.
11ABH	The calculated derivative value overflowed.
11ACH	The PID operation result overflowed.
11ADH	A value less than the output lower limit value is set for the output upper limit value.
11AEH	A value less than 0 is set for the input variation alarm setting value or output variation alarm setting value.
11AFH	■Step response method Improper auto tuning result • The deviation at the start of auto tuning (step response method) is 150 or less. • The deviation at the end of auto tuning (step response method) is one-third or more of the deviation at the time of start.
11B0H	■Step response method Auto tuning operation direction mismatch • The action direction estimated according to the relation between the set value (SV) and the process value (PV) at the start of auto tuning (step response method) and the action direction of the manipulated value (MV) did not match.
11B1H	■Step response method Improper auto tuning operation • Auto tuning (step response method) failed to operate correctly because the process value (PV) did not change properly.
11B2H	■Limit cycle method A value equal to or less than the output lower limit value (LLV) is set for the output upper limit value (ULV) for auto tuning (limit cycle method).

Error code (SD0)	Description
11B3H	■Limit cycle method A value set for the PV value threshold (hysteresis) width (SHPV) for auto tuning (limit cycle method) is out of the range, SHPV < 0.
11B4H	■Limit cycle method The system area used for auto tuning (limit cycle method) has been overwritten.
11B5H	■Limit cycle method The measurement time for auto tuning (limit cycle method) has been exceeded, and τ and τon time cannot be obtained properly. (τon > τ, τon < 0, τ < 0)
11B6H	■Limit cycle method The proportional gain (K_P) calculated by auto tuning (limit cycle method) overflowed.
11B7H	■Limit cycle method The integral time (T_I) calculated by auto tuning (limit cycle method) is out of the range, 0 to 32767.
11B8H	■Limit cycle method The derivative time (T_D) calculated by auto tuning (limit cycle method) is out of the range, 0 to 32767.

10 PID CONTROL INSTRUCTIONS

There are two types of instructions for PID control.

- PID OPERATION INSTRUCTIONS
- PID CONTROL INSTRUCTIONS

For how to use or compare them, refer to the following.

☞ Page 1134 PID operation instruction and PID control instruction

10.1 Overview

This section describes the operation methods, procedures, and helpful functions of PID control by using PID control instructions.

Point

The PID control instructions include those for inexact differential and exact differential.

Inexact differential is PID control that applies a primary delay filter to the input of a differentiation term, and is useful for the following.

- For control susceptible to high-frequency noise
- When energy effective to actuate an operation end is not provided when a step change occurs in an exact differential system

Exact differential is PID control that uses the input of a differential term as it is.

Operation method

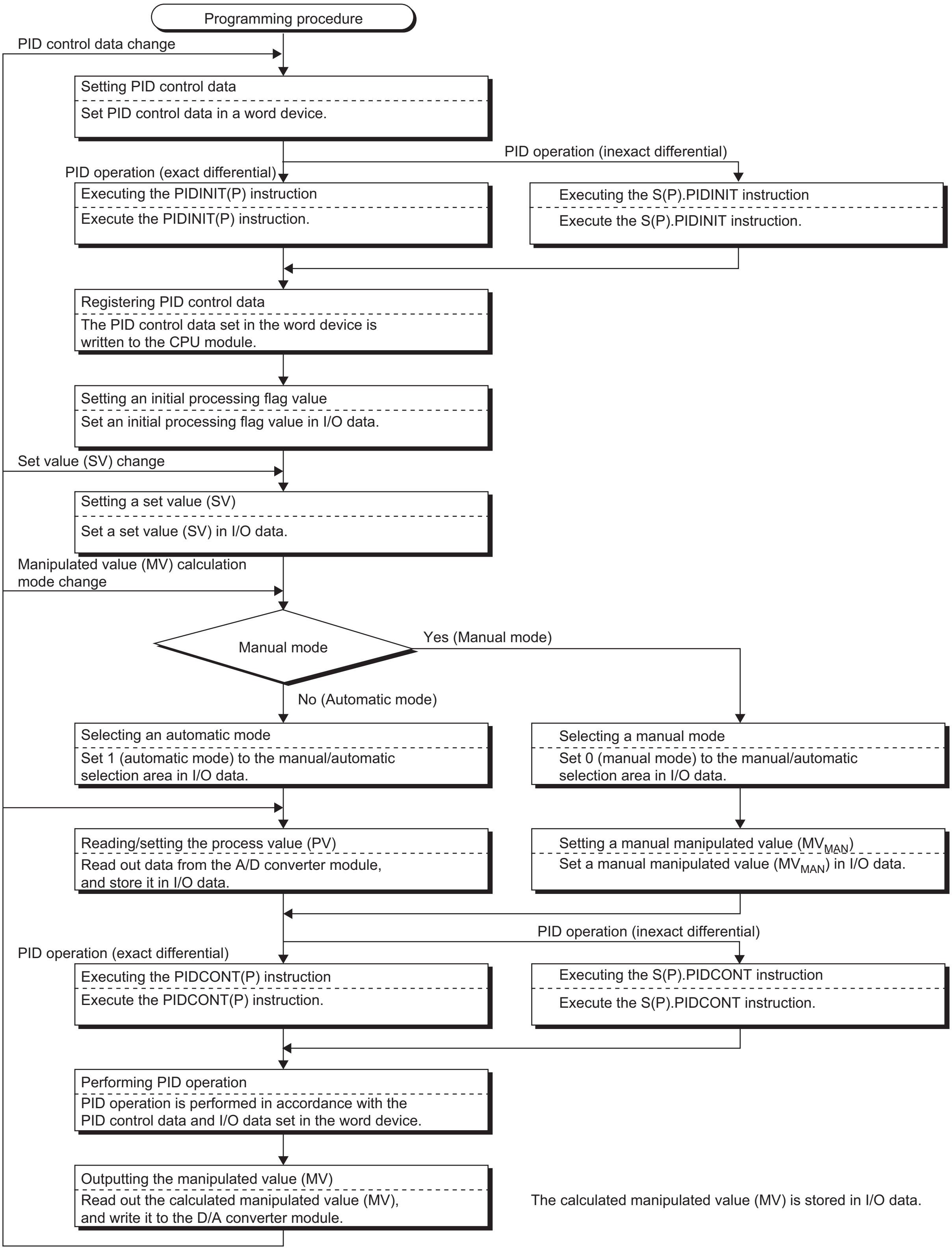
Two types of operation methods are available for PID control by using PID control instructions: velocity type and process value differentiation type.

The following table summarizes each operation methods.

Operation method	Description
Velocity type	This type of control calculates variations in the manipulated values (MV) during PID operation. The actual MV is the accumulated value of variations calculated each sampling period.
Process value differentiation type	This type of control performs PID operation by differentiating the process value (PV). Since the deviation is not subject to differentiation, an abrupt change in output due to derivative action can be reduced when the deviation is generated by changing the set value.

PID control procedure

Create a program for PID control following the procedure below. (Devices are used in the procedure.)



Point

- PID control data can be registered or changed every scan of the program. Whenever the data are registered or changed, execute the PIDINIT instruction. If not executed, the registered or changed data are not used when the PIDCONT instruction is executed.
- Note that the PIDINIT instruction does not need to be executed if the PID control data are changed for the specified loop by using the PIDPRMW instruction.

PID control data

PID control data is used to set the reference values for PID operation. The set values need to be registered to the CPU module by using the PIDINIT instruction before PID operation by the PIDCONT instruction starts. The PID control data can be set to a word device area with any numbers. Note that all the data for the number of loops used must be set in the area with consecutive device numbers.

For data assignment, refer to the following.

- Inexact differential: Page 1160 S(P).PIDINIT
- Exact differential: Page 1172 PIDINIT(P)

PID control data are classified into two types: data common to all loops and data for each loop.

Type	Item	Description	Setting range	Processing when set data is out of the valid range
Data common to all loops	Number of loops used	Number of loops where PID operation is performed	1 to 32	An error occurs, and PID operation is not performed for all loops.
	Number of loops in one scan	Number of loops to be used in one PID operation when multiple loops reach the sampling period at a same time	1 to 32	

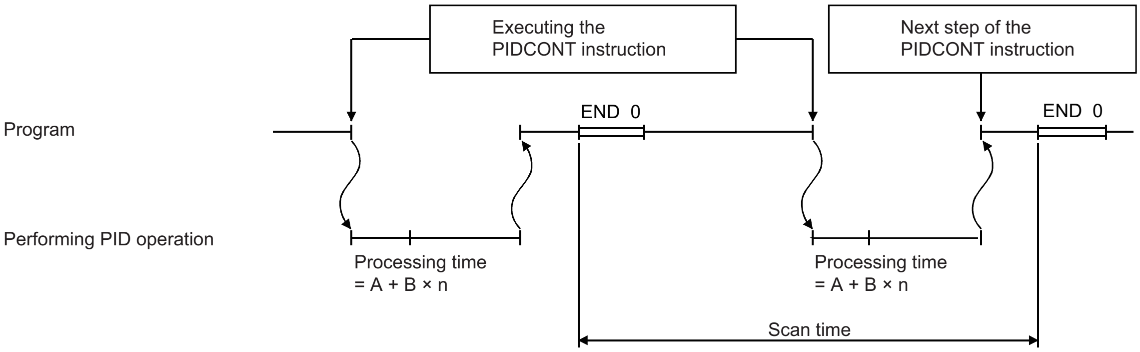
Type	Item	Description	Setting range	Processing when set data is out of the valid range
Data for each loop	Operational expression selection	Selects direct action or reverse action.	0: Direct action 1: Reverse action	An error occurs, and PID operation is not performed for the corresponding loop.
	Sampling period (T_S)	Sets a cycle of PID operation.	1 to 6000 (in increments of 10ms)	
	Proportional constant (K_P)	A constant of proportionality in PID operation	1 to 10000 (in increments of 0.01)	
	Integral constant (T_I)	A constant that expresses the magnitude of the integral action effect. Increasing the integral constant slows down the variation in the manipulated value.	1 to 32767 (in increments of 100ms)	
	Derivative constant (T_D)	A constant that expresses the magnitude of the derivative action effect. Increasing the derivative constant causes a significant change in the manipulated value even with slight change of the control objective.	0 to 30000 (in increments of 10ms)	
	Filter coefficient (α)	Degree of filtering applied to the process value (input value from the A/D converter module). The filtering effect decreases as the value gets closer to 0.	1 to 100	
	Manipulated value lower limit (MVLL)	Lower limit of the manipulated value (MV) calculated by PID operation in automatic mode. If the MV is smaller than the MVLL, the MVLL is used as the MV.	-50 to 2050[*1] -32768 to 32767[*2]	In the case of *1, if the MVLL or MVHL value is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
	Manipulated value upper limit (MVHL)	Upper limit of the manipulated value (MV) calculated by PID operation in automatic mode. If the MV is bigger than the MVHL, the MVHL is used as the MV.	-50 to 2050[*1] -32768 to 32767[*2]	
	Manipulated value variation rate limit (ΔMVL)	Limit of variation in the manipulated values calculated last time and this time. If the variation exceeds the limit, 1 is set to b1 of the alarm device. Note that the variation amount will not be limited actually. Even if the variation exceeds the limit, it is used as it is, and the MV is calculated.	0 to 2000[*1] 0 to 32767[*2]	In the case of *1, if the ΔMVL or ΔPVL value is out of the valid range, it will be converted as follows. • If the value is smaller than 0, 0 is used. • If the value is bigger than 2000, 2000 is used.
	Process value variation rate limit (ΔPVL)	Limit of variation in the process values input last time and this time. If the variation exceeds the limit, 1 is set to b0 of the alarm device. Note that the variation amount will not be limited actually. Even if the variation exceeds the limit, it is used as it is, and the PID operation is performed.	0 to 2000[*1] 0 to 32767[*2]	
	Derivative gain (K_D)[*3]	A time period (operation delay) for derivative action. The time period decreases as the value gets bigger. (The operation becomes closer to exact differential.)	0 to 32767 (in increments of 0.01)	An error occurs, and PID operation is not performed for the corresponding loop.

*1 When the PID limit is restricted
*2 When the PID limit is not restricted
*3 For PID control (inexact differential) only

■Number of loops used and number of execution loops in one scan

The number of loops used is the number of loops where PID operation is performed. The sampling period is measured for the set number of loops when the PIDCONT instruction is executed, and PID operation is performed for the loops that reach the specified sampling period.

Processing time of the PIDCONT instruction increases in proportion to the number of loops where PID operation is performed. The number of execution loops in one scan is the number of loops where PID operation is performed in one scan when there are multiple loops that reach the specified sampling period. If this number is set, PID operation is performed only for the set number of loops, and PID operation for the rest of the loops that reach the sampling period will be performed in the next scan.



A: Time required to measure sampling period

B: Time required to perform PID operation for a single loop

n: Number of loops

Point

When the number of loops that reach the sampling period exceeds the number of execution loops in one scan, the priority order will be as follows:

- The loop with the smallest loop number is given the highest priority.
- If there are loops where PID operation is performed and not performed in the last scan, the priority is given to the ones where PID operation is not performed in the last scan.

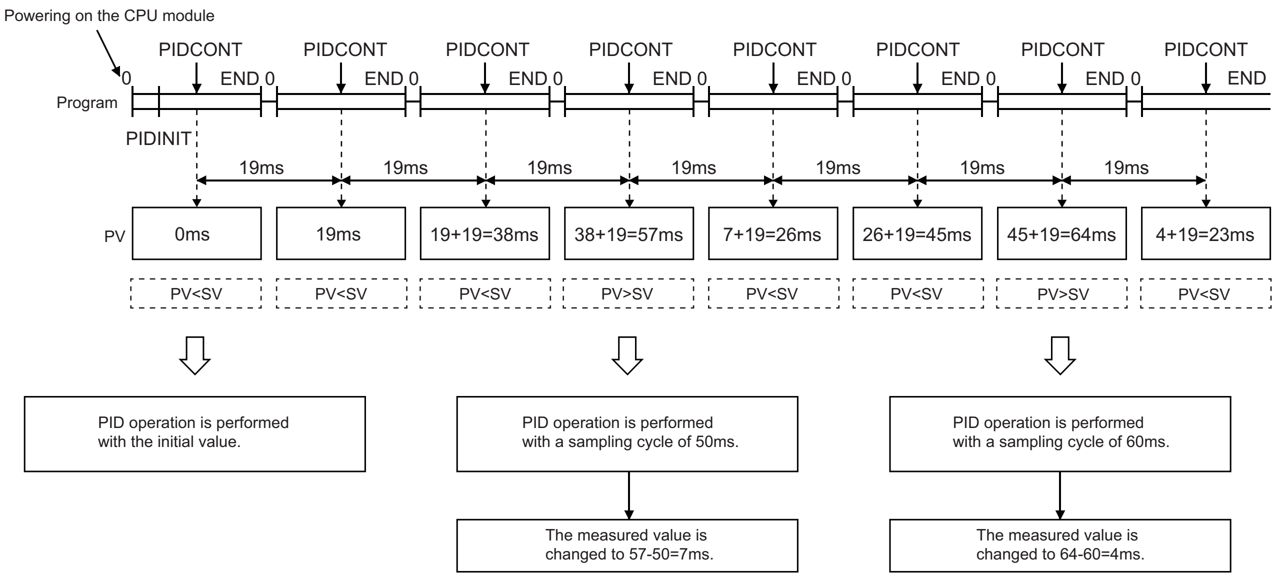
■Sampling period

Sampling period is a cycle in which PID operation is performed. The measurement time of a single scan is added to the total measurement time up to the last scan every time the PIDCONT instruction is executed. The total measurement time reaches or exceeds the set sampling period, PID operation of the corresponding loop is performed.

Set the sampling period used in PID operation in increments of 10ms.

Ex.

When the sampling period is 50ms



PV: Measured value

SV: Set value

Restriction

Since the sampling period is measured at execution of the PIDCONT instruction, a value smaller than the program scan time cannot be set as the sampling period. If set, PID operation is performed with a scan time.

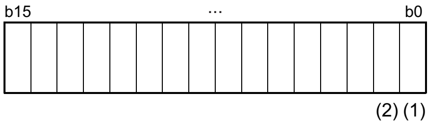
I/O data

I/O data consists of input data, such as the set value (SV) and process value (PV), which are set to perform PID operation, and output data, such as operation results. The I/O data can be set to a word device area with any numbers. Note that all the data for the number of loops used must be set in the area with consecutive device numbers.

For data assignment, refer to the following.

- Inexact differential: Page 1163 S(P).PIDCONT
- Exact differential: Page 1174 PIDCONT(P)

The I/O data area is divided into two areas: data area assigned for each loop and work area used by the system.

Item		Description	Setting range	Remarks
Set value	SV	A target value in PID control	0 to 2000[*1] -32768 to 32767[*2]	In the case of *1, if the SV is out of the valid range, it will be converted as follows. • If the value is smaller than 0, 0 is used. • If the value is bigger than 2000, 2000 is used.
Process value	PV	A feedback data from the control target to the A/D converter module	-50 to 2050[*1] -32768 to 32767[*2]	In the case of *1, if the PV is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
Automatic manipulated value	MV	A manipulated value calculated by PID operation. The value is output from the D/A converter module to the control target.	-50 to 2050[*1] -32768 to 32767[*2]	—
Process value after filtering	PVf	A process value calculated by the following operational expression $PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$	-50 to 2050[*1] -32768 to 32767[*2]	—
Manual manipulated value	MV_{MAN}	Data output from the D/A converter module in manual mode	-50 to 2050[*1] -32768 to 32767[*2]	In the case of *1, if the MV_{MAN} is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
Manual/automatic selection	MAN/AUTO	Select the type of output data to the D/A converter module: manual manipulated value or automatic manipulated value. In manual mode, the automatic manipulated value remains unchanged.	0: Automatic manipulated value 1: Manual manipulated value	An error occurs if the set value is other than 0 or 1, and PID operation of the corresponding loop is not executed.
Alarm	ALARM	Checks whether the variation rate of the automatic manipulated value (MV) or process value (PV) is within the valid range or not. Once the value is set, it is held until the CPU module is reset.  (1) If the PV is out of the limited range, 1 is set to b0. (2) If the MV is out of the limited range, 1 is set to b1.	—	—

[*1] When the PID limit is restricted

[*2] When the PID limit is not restricted

Helpful functions

During PID operation by using PID control instructions, bumpless transfer and manipulated value upper/lower limit control are automatically executed.

Bumpless transfer

This function controls the manipulated value (MV) continuously when the control mode is switched from manual to automatic, or vice-versa. When the mode is switched, data are transferred between the MV storage areas for manual mode and automatic mode.

- From manual to automatic: MV in manual mode is transferred to the MV storage area for automatic mode.
- From automatic to manual: MV in automatic mode is transferred to the MV storage area for manual mode.

Switch the mode in I/O data. (☞ Page 1155 I/O data)

Point

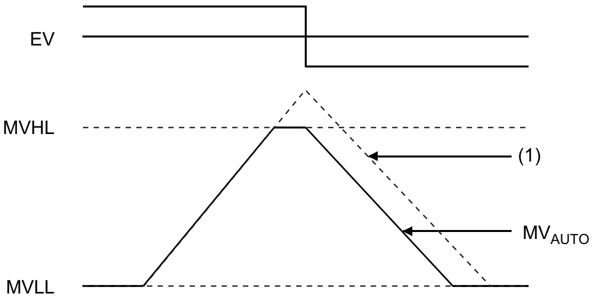
PID control is performed in each mode as described below.

- Automatic mode: The CPU module controls the target with the MV calculated by PID operation using the PID control instructions.
- Manual mode: The CPU module controls the target with the MV calculated without performing PID operation using the PID control instructions. Loops set in manual mode stores the process value (PV) in the set value storage area every sampling cycle.

Manipulated value upper/lower limits control

This function controls the upper or lower limit of the manipulated value (MV) calculated by PID operation. The function is only enabled in automatic mode and is not executed in manual mode.

The manipulated value (MV) calculated by PID operation can be limited within the range set by the manipulated value upper limit (MVHL) and manipulated value lower limit (MVLL). The following figure shows the operation of the function.



(1) MV$_{AUTO}$ without limit control

The manipulated value upper limit (MVHL) and manipulated value lower limit (MVLL) can be set to each loop within the range of -50 to 2050 or within any desired range. The following values are set by default.

- Manipulated value upper limit: 2000
- Manipulated value lower limit: 0

An error occurs if the MVHL is smaller than the MVLL.

Transferring PV to SV in manual mode

This function executes the PIDCONT instruction even in manual mode. In manual mode, whether transferring the process value (PV) input from the A/D converter module to the set value (SV) storage area during execution of the PIDCONT instruction or not can be selected by turning on or off SM792 or SM794. SM792 and SM794 are set to off by default. The PV and SV are stored in the specified device or label in the I/O data area by using the PIDCONT instruction.

SM792, SM794	Operation
Off	• The PV is transferred to the SV storage device or label during execution of the PIDCONT instruction. • When the mode is switched to automatic mode, the MV output in manual mode is continued. • If the SV is changed after the mode is switched to automatic mode, control from the MV output to the SV is enabled.
On	• The PV is not transferred to the SV storage device or label during execution of the PIDCONT instruction. • When the mode is switched to automatic mode, control from the MV output in manual mode to the SV is enabled. • Before switching to automatic mode, store the SV to the SV storage device or label.

Point

Depending on the on/off status of SM792 or SM794, the following differences apply when switching manual mode to automatic mode.

- When SM792 or SM794 is off, the PV is transferred to the SV storage device or label. Therefore, there is no difference between the PV and SV and an abrupt change does not occur in MV when the mode is switched. Note, however, that since the SV after the mode is switched differs from the target value in automatic mode, the user needs to change the SV step by step in the program so that it matches the target value.
- When SM792 or SM794 is on, the PV is not transferred to the SV storage device or label. Therefore, a difference exists between the PV and SV when the mode is switched. If the difference is large when the mode is switched, an abrupt change may occur in MV. Use this method in a system where the mode is switched when the PV has fully neared the SV. PID control in automatic mode can be executed immediately without the SV being changed step by step in the program.

Changing the setting range of PID control data and I/O data

The setting range of PID control data and I/O data can be changed as desired.

To change the range, turn on the bit corresponding to the target loop in SD792, SD793, SD794, and SD795.

Inexact differential	Exact differential	Operation
SD794	SD792	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 b0: Loop 1 b1: Loop 2 ⋮ b14: Loop 15 b15: Loop 16
SD795	SD793	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 b0: Loop 17 b1: Loop 18 ⋮ b14: Loop 31 b15: Loop 32

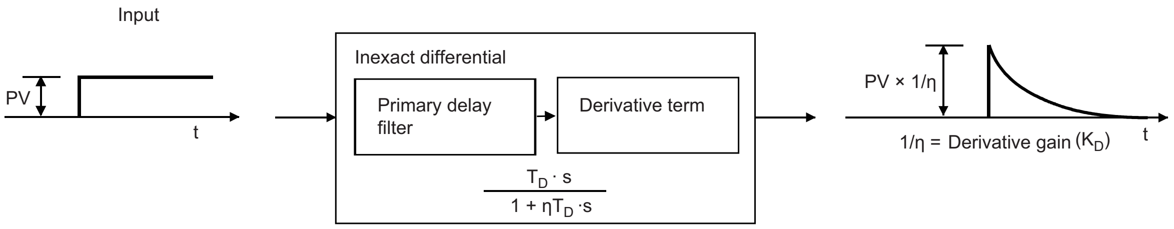
0: PID limit restricted (default)

1: PID limit not restricted

10.2 PID Control Instructions (Inexact Differential)

Inexact differential is PID control that applies a primary delay filter to the input of a differentiation term. Inexact differential is effective in the following cases:

- For control susceptible to high-frequency noise
- When energy effective to actuate an operation end is not provided when a step change occurs in an exact differential system

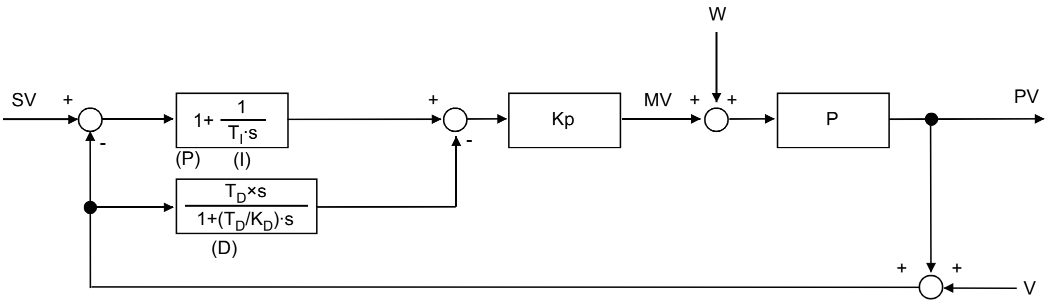


The following table summarizes the performance specifications of PID control instructions for inexact differential.

Item			When the PID limit is restricted	When the PID limit is not restricted
Number of PID control loops		—	32 maximum	
Sampling period		T_S	0.01 to 60.00s	
PID operation method		—	Process value differentiation type (inexact differential) (direct action/reverse action)	
PID constants setting range	Proportional constant	K_P	0.01 to 100.00	
	Integral constant	T_I	0.1 to 3000.0s	
	Derivative constant	T_D	0.00 to 300.00s	
	Derivative gain	K_D	0.00 to 300.00	
Set value setting range		SV	0 to 2000	-32768 to 32767
Process value setting range		PV	-50 to 2050	-32768 to 32767
Manipulated value output range		MV		

The following are the block diagram and operational expressions of PID operation.

- Block diagram of PID operation (inexact differential)



K_P: Gain

W: Disturbance

P: Controlled system

V: Detected noise

- Operational expressions

Action	Operational expression
Direct action	$EV_n = PV_{fn} - SV$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S + \frac{T_D}{K_D}}(PV_{fn} - 2PV_{fn-1} + PV_{fn-2}) + \frac{\frac{T_D}{K_D}}{T_S + \frac{T_D}{K_D}} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$
Reverse action	$EV_n = SV - PV_{fn}$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S + \frac{T_D}{K_D}}(-PV_{fn} + 2PV_{fn-1} - PV_{fn-2}) + \frac{\frac{T_D}{K_D}}{T_S + \frac{T_D}{K_D}} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{fn}	Process value of the sampling period this time (after filtering)
PV_{fn-1}	Process value of the sampling period last time (after filtering)
PV_{fn-2}	Process value of the sampling period two times before (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
D_{n-1}	Derivative term of the sampling period last time
T_S	Sampling period
K_P	Proportional constant
T_I	Integral constant
T_D	Derivative constant
K_D	Derivative gain

The PV_{fn} is calculated by using the following operational expression. If the filter coefficient is not set to the input data, the PV_{fn} will be same as the process value of the input data.

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

PV_n: Process value for the sampling period this time, α: Filter coefficient, PV_{fn-1}: Process value for the sampling period last time (after filtering)

PV_{fn} is stored in the I/O data area. (☞ Page 1155 I/O data)

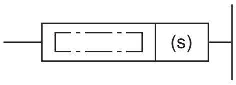
Registering the PID control data to the CPU module

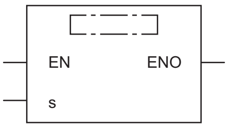
S(P).PIDINIT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

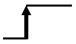
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions store the PID control data by the number of loops used that is set in the specified device number and later altogether in the CPU module.

Ladder	ST
	ENO:=S_PIDINIT(EN,s); ENO:=SP_PIDINIT(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
S.PIDINIT	
SP.PIDINIT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the PID control data is set	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the CPU module to enable PID control. (☞ Page 1151 PID control data)
- The PID control data are assigned as follows.

Item	PID control data	
Common to all loops	(s)+0	Number of loops used
	(s)+1	Number of loops in one scan
For loop No.1 (14 words)	(s)+2	Operational expression selection
	(s)+3	Sampling period (T_S)
	(s)+4	Proportional constant (K_P)
	(s)+5	Integral constant (T_I)
	(s)+6	Derivative constant (T_D)
	(s)+7	Filter coefficient (α)
	(s)+8	Manipulated value lower limit (MVLL)
	(s)+9	Manipulated value upper limit (MVHL)
	(s)+10	Manipulated value variation rate limit (ΔMVL)
	(s)+11	Process value variation rate limit (ΔPVL)
	(s)+12	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+13	Derivative gain (K_D)
	(s)+14	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+15	Fixed to "0". (An error occurs if a value other than "0" is specified.)
⋮	⋮	⋮
For loop No.n (14 words)[1]	(s)+(m+0)	Operational expression selection
	(s)+(m+1)	Sampling period (T_S)
	(s)+(m+2)	Proportional constant (K_P)
	(s)+(m+3)	Integral constant (T_I)
	(s)+(m+4)	Derivative constant (T_D)
	(s)+(m+5)	Filter coefficient (α)
	(s)+(m+6)	Manipulated value lower limit (MVLL)
	(s)+(m+7)	Manipulated value upper limit (MVHL)
	(s)+(m+8)	Manipulated value variation rate limit (ΔMVL)
	(s)+(m+9)	Process value variation rate limit (ΔPVL)
	(s)+(m+10)	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+(m+11)	Derivative gain (K_D)
	(s)+(m+12)	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+(m+13)	Fixed to "0". (An error occurs if a value other than "0" is specified.)

[1] m=(n-1)×14+2

(1) Fixed to "0". An error results if a value other than "0" is specified.

- The number of device points used for PID control data setting is calculated by the following formula.

Number of device points = 2 + 14 × n (n: number of loops used)

- Specify each data in binary.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- If the S(P).PIDINIT instruction is executed at two or more locations during a single scan, the setting value of the S(P).PIDINIT instruction executed nearest to the S(P).PIDCONT instruction will be valid.
- Execute the S(P).PIDINIT instruction before execution of the S(P).PIDCONT instruction. To perform PID control, the S(P).PIDINIT instruction must be executed.

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s). • The value set for the PID control data is out of the setting range. • (Number of loops used) < (Number of loops executed in one scan) • (Manipulated value upper limit) < (Manipulated value lower limit) • The area fixed to 0 in the PID control data is not 0.

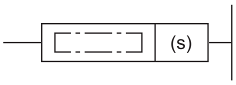
Performing PID operation

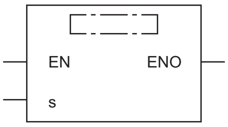
S(P).PIDCONT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

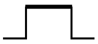
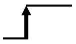
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions measure the sampling cycle and perform PID operation when the execution command turns on.

Ladder	ST
	ENO:=S_PIDCONT(EN,s); ENO:=SP_PIDCONT(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
S.PIDCONT	
SP.PIDCONT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device assigned to the I/O data area	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- The S(P).PIDCONT instructions measure the sampling cycle and perform PID operation.
- Based on the setting value (SV) and process value (PV) in the I/O data area allocated to the device number specified by (s) and later, these instructions perform PID operation and store the operation result in the automatic manipulated value (MV) area in the I/O data area.
- The S(P).PIDCONT instructions perform PID operation when it is executed for the first time after a lapse of the specified sampling cycle.
- During PID control, be sure to turn on the control command to allow the S(P). PIDCONT instruction to be executed every scan. Failure to execute the instruction every scan disables PID operation in normal sampling cycles. The S(P).PIDCONT instruction cannot be executed more than once in a single scan. Executing the instruction more than once in a single scan disables PID operation in normal sampling cycles.
- The S(P).PIDCONT instruction cannot be written and used in interrupt programs. Writing an S(P).PIDCONT instruction in the interrupt program disables PID operation in normal sampling cycles.
- In (s), specify the head of the device number specified in the I/O data area. (☞ Page 1155 I/O data)
- If a file register is specified as an I/O data area, do not apply memory protection for the file register. If memory protection is applied, normal PID operation is disabled although no error results.
- The I/O data are assigned as follows.

Item		I/O data	
Write		(s)+0	Initial processing flag
Read/write disabled		(s)+1 ⋮ (s)+9	Work area for PID control (system use only)
I/O data area for loop No.1 (23 words)	Write	(s)+10	Set value (SV)
		(s)+11	Process value (PV)
	Read	(s)+12	Automatic manipulated value (MV)
		(s)+13	Process value after filtering (PVf)
	Write	(s)+14	Manual manipulated value (MV_{MAN})
		(s)+15	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+16	Alarm (ALARM)
	Read/write disabled	(s)+17 ⋮ (s)+32	Work area for loop No.1 (system use only)
⋮	⋮	⋮	⋮
I/O data area for loop No.n (23 words)[*1]	Write	(s)+(m+0)	Set value (SV)
		(s)+(m+1)	Process value (PV)
	Read	(s)+(m+2)	Automatic manipulated value (MV)
		(s)+(m+3)	Process value after filtering (PVf)
	Write	(s)+(m+4)	Manual manipulated value (MV_{MAN})
		(s)+(m+5)	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+(m+6)	Alarm (ALARM)
	Read/write disabled	(s)+(m+7) ⋮ (s)+(m+22)	Work area for loop No.n (system use only)

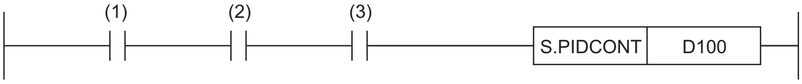
[*1] m=(n-1)×23+10

- The number of device points used for I/O data setting is calculated by the following formula.

Number of device points = 10 + +23 × n (n: number of loops used)

- Specify each data in binary.
- The initial processing flag sets the processing to be performed at the start of PID operation.
  - Initial operation processing is performed assuming that the sampling cycle that has been set has been reached.
  - If the initial processing flag is 0, PID operations for the number of loops used are performed altogether in a single scan. If it is not 0, PID operations for the number of loops used are divided and processed in several scans, and sampling is started sequentially from the loop that has completed initial processing. The number of processing loops per scan is the number of execution loops per scan that has been set.

- Write data to the I/O data "Write" area by users with the program. Users can read data from the I/O data "Read" area with the program. Never attempt to write data to the area indicated by "Read/write disabled" or "Read"; otherwise, normal operation can no longer be performed. Note that, when starting control from the initial status, the data areas must be cleared by the program.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- Even when the manual manipulated value (MV_{MAN}) is output in manual mode, execute the S(P).PIDCONT instruction every scan. Unless the S(P).PIDCONT instruction is executed, the bumpless function cannot be performed.
- Apply an interlock using the READY signal of each module so that the S(P).PIDCONT instruction is executed only when the A/D converter module used to obtain the process value (PV) and the D/A converter module used to output the manipulated value (MV) are normal.



(1) Control command
(2) READY signal of the A/D converter module
(3) READY signal of the D/A converter module

If the instruction is executed when these modules are not normal, PID operation cannot be performed normally as the result of failure in normal acquisition of process values (PV) or in normal output of manipulated values (MV).

Operation error

Error code (SD0)	Description
3405H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3422H	The S(P).PIDINIT instruction is not executed before the S(P).PIDCONT instruction.

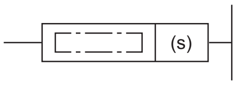
Stopping the operation of specified loop number

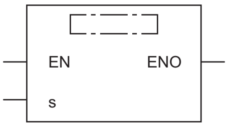
S(P).PIDSTOP

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

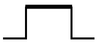
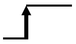
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions stop the PID operation of the specified loop number.

Ladder	ST
	ENO:=S_PIDSTOP(EN,s); ENO:=SP_PIDSTOP(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
S.PIDSTOP	
SP.PIDSTOP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions stop the PID operation of the loop number in the device specified by (s). The loop stopped by the S(P).PIDSTOP instruction does not restart PID operation even if the S(P).PIDINIT instruction is executed.
- Each instruction holds operation data while the loop is stopped.

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3422H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDSTOP instruction.

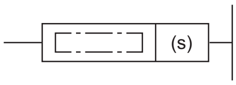
Starting the operation of specified loop number

S(P).PIDRUN

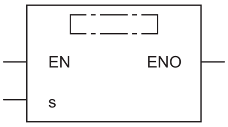
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

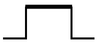
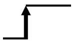
These instructions start the PID operation of the specified loop number.

Ladder	ST
	ENO:=S_PIDRUN(EN,s); ENO:=SP_PIDRUN(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDRUN	
SP.PIDRUN	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions start the PID operation of the loop number in the device specified by (s). These instructions are used to re-execute the PID operation of the loop number that has been stopped by the S(P).PIDSTOP instruction.
- The S(P).PIDRUN instruction, if executed for a loop number already in progress of PID operation, performs no processing.

Operation error

Error code (SD0)	Description
3405H	The loop number specified by (s) does not exist.
	(s) is outside the range from 1 to 32.
3422H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDRUN instruction.

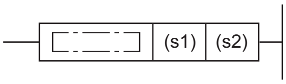
Changing the parameters of specified loop number

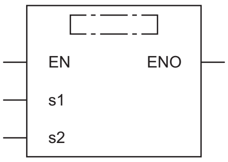
S(P).PIDPRMW

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

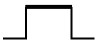
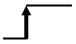
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the operation parameter of the specified loop number to the PID control data stored in the specified device number and later.

Ladder	ST
	ENO:=S_PIDPRMW(EN,s1,s2); ENO:=SP_PIDPRMW(EN,s1,s2);

FBD/LD


■Execution condition

Instruction	Execution condition
S.PIDPRMW	
SP.PIDPRMW	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Loop number to be changed	1 to 32	16-bit unsigned binary	ANY16
(s2)	Start device where the change-target PID control data is stored	—	Word	ANY16[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.
- The following figure shows the configuration of the PID control data in the device specified by (s2) and later.

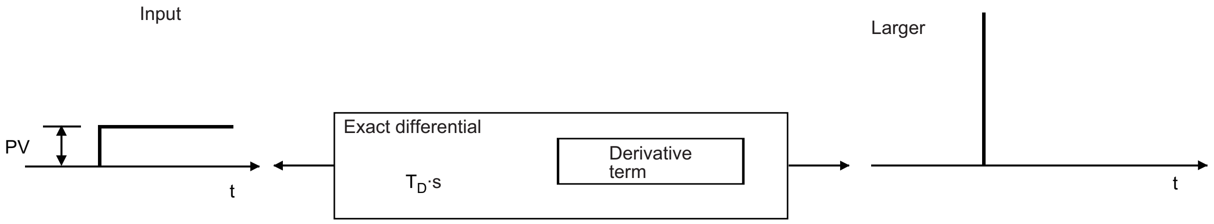
PID control data	
(s2)+0	Operational expression selection
(s2)+1	Sampling period (T_S)
(s2)+2	Proportional constant (K_P)
(s2)+3	Integral constant (T_I)
(s2)+4	Derivative constant (T_D)
(s2)+5	Filter coefficient (α)
(s2)+6	Manipulated value lower limit (MVLL)
(s2)+7	Manipulated value upper limit (MVHL)
(s2)+8	Manipulated value variation rate limit (ΔMVL)
(s2)+9	Process value variation rate limit (ΔPVL)
(s2)+10	0
(s2)+11	Derivative gain (K_D)
(s2)+12	0
(s2)+13	0

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s1). • The specified loop number does not exist. • The specified value is other than 1 to 32.
	Out-of-range data is set in the device specified by (s2). • The PID control data is out of the setting range. • The PID control data in the devices specified by (s2)+10, (s2)+12, and (s2)+13 is not 0.
3422H	The S(P).PIDINIT instruction is not executed before the S(P).PIDPRMW instruction.

10.3 PID Control Instructions (Exact Differential)

Exact differential is PID control that uses the input of a differential term as it is.

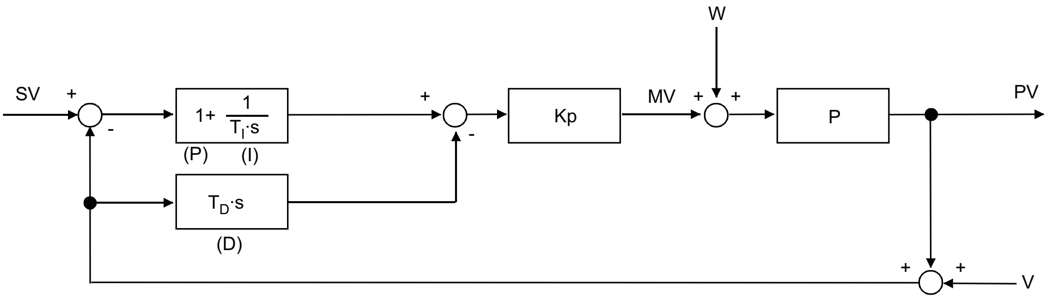


The following table summarizes the performance specifications of PID control instructions for exact differential.

Item			When the PID limit is restricted	When the PID limit is not restricted
Number of PID control loops		—	32 maximum	
Sampling period		T_S	0.01 to 60.00s	
PID operation method		—	Process value differentiation type (exact differential) (direct action/reverse action)	
PID constants setting range	Proportional constant	K_P	0.01 to 100.00	
	Integral constant	T_I	0.1 to 3000.0s	
	Derivative constant	T_D	0.00 to 300.00s	
Set value setting range		SV	0 to 2000	-32768 to 32767
Process value setting range		PV	-50 to 2050	-32768 to 32767
Manipulated value output range		MV		

The following are the block diagram and operational expressions of PID operation.

- Block diagram of PID operation (exact differential)



K_P: Gain
W: Disturbance
P: Controlled system
V: Detected noise

- Operational expressions

Action	Operational expression
Direct action	$EV_n = PV_{fn} - SV$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S}(PV_{fn} - 2PV_{fn-1} + PV_{fn-2})$ $MV_n = \Sigma\Delta MV$
Reverse action	$EV_n = SV - PV_{fn}$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S}(- PV_{fn} + 2PV_{fn-1} - PV_{fn-2})$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{fn}	Process value of the sampling period this time (after filtering)
PV_{fn-1}	Process value of the sampling period last time (after filtering)
PV_{fn-2}	Process value of the sampling period two times before (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
T_S	Sampling period
K_P	Proportional constant
T_I	Integral constant
T_D	Derivative constant

The PV_{fn} is calculated by using the following operational expression. If the filter coefficient is not set to the input data, the PV_{fn} will be same as the process value of the input data.

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

PV_n: Process value for the sampling period this time, α: Filter coefficient, PV_{fn-1}: Process value for the sampling period last time (after filtering)

PV_{fn} is stored in the I/O data area. (☞ Page 1155 I/O data)

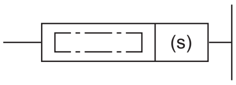
Registering the PID control data to the CPU module

PIDINIT(P)

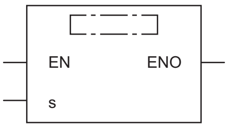
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

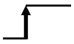
These instructions store the PID control data by the number of loops used that is set in the specified device number and later altogether in the CPU module.

Ladder	ST
	ENO:=PIDINIT(EN,s); ENO:=PIDINITP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDINIT	
PIDINITP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the PID control data is set	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the CPU module to enable PID control. (☞ Page 1151 PID control data)
- The PID control data are assigned as follows.

Item	PID control data	
Common to all loops	(s)+0	Number of loops used
	(s)+1	Number of loops in one scan
For loop No.1 (10 words)	(s)+2	Operational expression selection
	(s)+3	Sampling period (T_S)
	(s)+4	Proportional constant (K_P)
	(s)+5	Integral constant (T_I)
	(s)+6	Derivative constant (T_D)
	(s)+7	Filter coefficient (α)
	(s)+8	Manipulated value lower limit (MVLL)
	(s)+9	Manipulated value upper limit (MVHL)
	(s)+10	Manipulated value variation rate limit (ΔMVL)
	(s)+11	Process value variation rate limit (ΔPVL)
⋮	⋮	⋮
For loop No.n (10 words)[*1]	(s)+(m+0)	Operational expression selection
	(s)+(m+1)	Sampling period (T_S)
	(s)+(m+2)	Proportional constant (K_P)
	(s)+(m+3)	Integral constant (T_I)
	(s)+(m+4)	Derivative constant (T_D)
	(s)+(m+5)	Filter coefficient (α)
	(s)+(m+6)	Manipulated value lower limit (MVLL)
	(s)+(m+7)	Manipulated value upper limit (MVHL)
	(s)+(m+8)	Manipulated value variation rate limit (ΔMVL)
	(s)+(m+9)	Process value variation rate limit (ΔPVL)

[*1] m = (n-1) × 10 + 2

- The number of device points used for PID control data setting is calculated by the following formula.

Number of device points = 2 + 10 × n (n: number of loops used)

- Specify each data in binary.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- If the PIDINIT(P) instruction is executed at two or more locations during a single scan, the setting value of the PIDINIT(P) instruction executed nearest to the PIDCONT(P) instruction will be valid.
- Execute the PIDINIT(P) instruction before execution of the PIDCONT(P) instruction. PID control is disabled unless the PIDINIT(P) instruction has been executed.

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s). • The value set for the PID control data is out of the setting range. • (Number of loops used) < (number of execution loops in one scan) • (Manipulated value upper limit) < (Manipulated value lower limit)

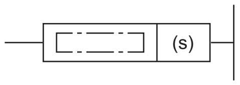
Performing PID operation

PIDCONT(P)

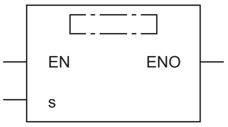
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

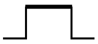
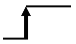
These instructions measure the sampling cycle and perform PID operation when the execution command turns on.

Ladder	ST
	ENO:=PIDCONT(EN,s); ENO:=PIDCONTP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDCONT	
PIDCONTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device assigned to the I/O data area	—	Word	ANY16[*1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- The PIDCONT(P) instructions measure the sampling cycle and perform PID operation.
- Based on the setting value (SV) and process value (PV) in the I/O data area allocated to the device number specified by (s) and later, these instructions perform PID operation and store the operation result in the automatic manipulated value (MV) area in the I/O data area.
- The PIDCONT(P) instructions perform PID operation when it is executed for the first time after a lapse of the specified sampling cycle.
- During PID control, be sure to turn on the control command to allow the PIDCONT(P) instruction to be executed every scan. Failure to execute the instruction every scan disables PID operation in normal sampling cycles. The PIDCONT(P) instruction cannot be executed more than once in a single scan. Executing the instruction more than once in a single scan disables PID operation in normal sampling cycles.
- The PIDCONT(P) instruction cannot be written and used in interrupt programs. Writing a PIDCONT(P) instruction in the interrupt program disables PID operation in normal sampling cycles.
- In (s), specify the head of the device number specified in the I/O data area. (☞ Page 1155 I/O data)
- If a file register is specified as an I/O data area, do not apply memory protection for the file register. If memory protection is applied, normal PID operation is disabled although no error results.
- The I/O data are assigned as follows.

Item		I/O data	
Write		(s)+0	Initial processing flag
Read/write disabled		(s)+1 ⋮ (s)+9	Work area for PID control (system use only)
I/O data area for loop No.1 (18 words)	Write	(s)+10	Set value (SV)
		(s)+11	Process value (PV)
	Read	(s)+12	Automatic manipulated value (MV)
		(s)+13	Process value after filtering (PVf)
	Write	(s)+14	Automatic manipulated value (MV_{MAN})
		(s)+15	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+16	Alarm (ALARM)
	Read/write disabled	(s)+17 ⋮ (s)+27	Work area for loop No.1 (system use only)
⋮	⋮	⋮	⋮
I/O data area for loop No.n (18 words)[*1]	Write	(s)+(m+0)	Set value (SV)
		(s)+(m+1)	Process value (PV)
	Read	(s)+(m+2)	Automatic manipulated value (MV)
		(s)+(m+3)	Process value after filtering (PVf)
	Write	(s)+(m+4)	Automatic manipulated value (MV_{MAN})
		(s)+(m+5)	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+(m+6)	Alarm (ALARM)
	Read/write disabled	(s)+(m+7) ⋮ (s)+(m+17)	Work area for loop No.n (system use only)

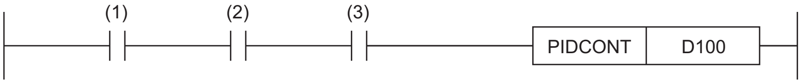
[*1] m=(n-1)×18+10

- The number of device points used for I/O data setting is calculated by the following formula.

Number of device points = 10 + +18 × n (n: number of loops used)

- Specify each data in binary.
- The initial processing flag sets the processing to be performed at the start of PID operation.
  - Initial operation processing is performed assuming that the sampling cycle that has been set has been reached.
  - If the initial processing flag is 0, PID operations for the number of loops used are performed altogether in a single scan. If it is not 0, PID operations for the number of loops used are divided and processed in several scans, and sampling is started sequentially from the loop that has completed initial processing. The number of processing loops per scan is the number of execution loops per scan that has been set.

- Write data to the I/O data "Write" area by users with the program. Users can read data from the I/O data "Read" area with the program. Never attempt to write data to the area indicated by "Read/write disabled" or "Read"; otherwise, normal operation can no longer be performed. Note that, when starting control from the initial status, the data areas must be cleared by the program.
- Even when the manual manipulated value (MV_{MAN}) is output in manual mode, execute the PIDCONT(P) instruction every scan. Unless the PIDCONT(P) instruction is executed, the bumpless function cannot be performed.
- Apply an interlock using the READY signal of each module so that the PIDCONT(P) instruction is executed only when the A/D converter module used to obtain the process value (PV) and the D/A converter module used to output the manipulated value (MV) are normal. If the instruction is executed when these modules are not normal, PID operation cannot be performed normally as the result of failure in normal acquisition of process values (PV) or in normal output of manipulated values (MV).



(1) Control command
(2) READY signal of the A/D converter module
(3) READY signal of the D/A converter module

Operation error

Error code (SD0)	Description
3405H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3422H	The PIDINIT(P) instruction is not executed before the PIDCONT(P) instruction.

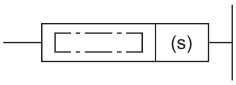
Stopping the operation of specified loop number

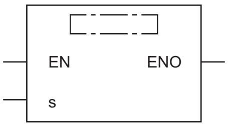
PIDSTOP(P)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

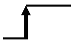
- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions stop the PID operation of the loop number in the device specified by (s).

Ladder	ST
	ENO:=PIDSTOP(EN,s); ENO:=PIDSTOPP(EN,s);

FBD/LD


■Execution condition

Instruction	Execution condition
PIDSTOP	
PIDSTOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions stop the PID operation of the loop number in the device specified by (s). The loop stopped by the PIDSTOP(P) instruction does not restart PID operation even if the PIDINIT(P) instruction is executed.
- Each instruction holds operation data while the loop is stopped.

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3422H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDSTOP(P) instruction.

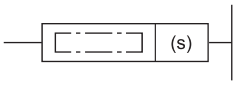
Starting the operation of specified loop number

PIDRUN(P)

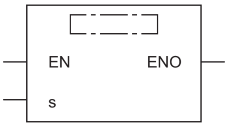
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

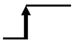
These instructions start the PID operation of the specified loop number.

Ladder	ST
	ENO:=PIDRUN(EN,s); ENO:=PIDRUNP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDRUN	
PIDRUNP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions start the PID operation of the loop number in the device specified by (s). These instructions are used to re-execute the PID operation of the loop number that has been stopped by the PIDSTOP(P) instruction.
- The PIDRUN(P) instruction, if executed for a loop number already in progress of PID operation, performs no processing.

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3422H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDRUN(P) instruction.

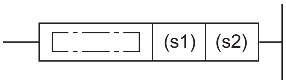
Changing the parameters of specified loop number

PIDPRMW(P)

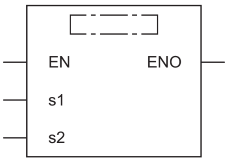
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

- [RnPCPU (redundant)] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the operation parameter of the specified loop number to the PID control data stored in the specified device number and later.

Ladder	ST
 (s1) (s2)	ENO:=PIDPRMW(EN,s1,s2); ENO:=PIDPRMWP(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDPRMW	
PIDPRMWP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Loop number to be changed	1 to 32	16-bit unsigned binary	ANY16
(s2)	Start device where the change-target PID control data is stored	—	Word	ANY16[1]
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.
- The following figure shows the configuration of the PID control data in the device specified by (s2) and later.

PID control data	
(s2)+0	Operational expression selection
(s2)+1	Sampling period (T_S)
(s2)+2	Proportional constant (K_P)
(s2)+3	Integral constant (T_I)
(s2)+4	Derivative constant (T_D)
(s2)+5	Filter coefficient (α)
(s2)+6	Manipulated value lower limit (MVLL)
(s2)+7	Manipulated value upper limit (MVHL)
(s2)+8	Manipulated value variation rate limit (ΔMVL)
(s2)+9	Process value variation rate limit (ΔPVL)

Operation error

Error code (SD0)	Description
3405H	Out-of-range data is set in the device specified by (s1). The specified loop number does not exist. The specified value is other than 1 to 32.
	The PID control data in the device specified by (s2) is out of the setting range.
3422H	The PIDINIT(P) instruction is not executed before the PIDPRMW(P) instruction.

11 PROCESS CONTROL INSTRUCTIONS

Point

When a process control program is created, using process control function blocks is recommended
Process control function blocks have features as follows.

- A process control program can be easily created by placing and connecting FB elements.
- Since the initial value of the function block can be set in the "FB Property" window of the engineering tool, the program for the initial value setting is not required.
- An operation constant can be input to a label indicating a tag name without being conscious of address of a device.
- The operating status of a tag FB can be checked and controlled by accessing the tag data from the faceplate of an engineering tool.

For details on the process control function blocks, refer to the following.

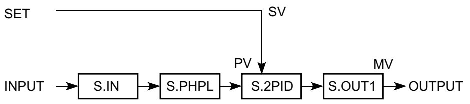
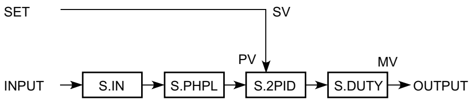
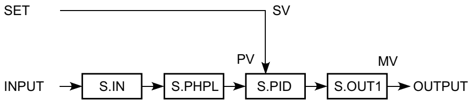
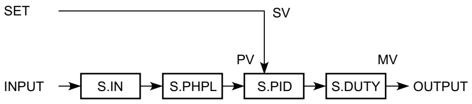
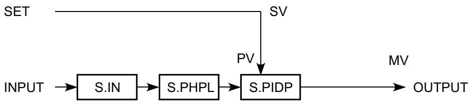
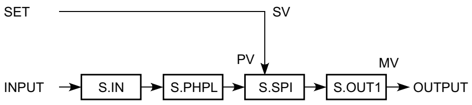
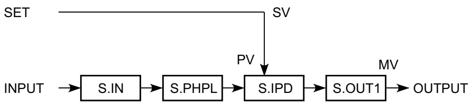
📖 MELSEC iQ-R Programming Manual (Process Control Function Blocks)

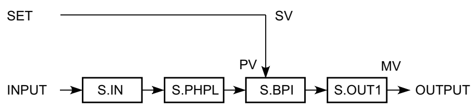
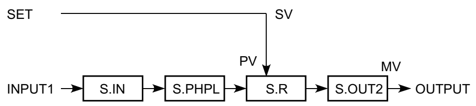
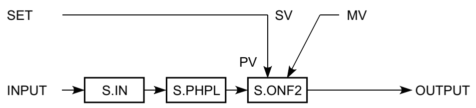
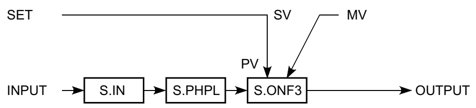
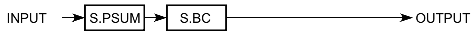
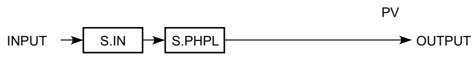
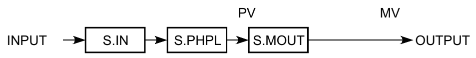
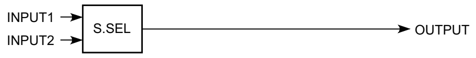
11.1 Overview

This section describes the loop type that can be configured by process control instructions, data configurations of the instructions, instruction execution methods, and precautions.

Basic loop types

The following table summarizes basic loop types configured by combinations of process control instructions.

Loop type	Configuration	Application
Two-degree-of-freedom PID control (S2PID)	 	Used for general PID control (two degrees of freedom). (Speed type) PID operation is performed every control cycle.
PID control (SPID)	 	Used for general PID control. (Speed type) PID operation is performed every control cycle.
PIDP control (SPIDP)		Used for general PID control. (Position type) PID operation is performed every control cycle.
Sample PI control (SSPI)		Used for processes which involve much dead time. PI control is executed only for the control execution time every control cycle, and later output is held constant.
I-PD control (SIPD)		Used to make a slow response so as not to give a shock to the operation terminal and process when the set value is changed.

Loop type	Configuration	Application
Blend PI control (SBPI)		Used for processes which allow a constant manipulated value in the long run even if it varies in the short term.
Ratio control (SR)		Performs control so that a given manipulated value keeps a constant ratio with other rates of change.
Two-position (on/off) control (SONF2)		Performs control so that the manipulated value is turned on or off depending on whether the deviation is positive or negative.
Three-position (on/off) control (SONF3)		Performs control by outputting three-area signals for process values. This control can suppress rapid changes in the manipulated value.
Batch counter (SBC)		Performs valve on/off control in the process of batch charging to the tank.
Program setter (SPGS)		Outputs according to the temporal change of the value that has been set in advance.
Manual output (SMOUT)		Operates the operation terminal for manual output.
Monitor (SMON)		Inputs process values to detect process errors such as upper/ lower limit alarms.
Manual output with monitor (SMWM)		Performs manual operation while inputting process values to check that no error is caused.
Selector (SSEL)		Used for signal selection.

Point

For the loop type processing time, refer to the following.

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

Process control instructions and data configuration

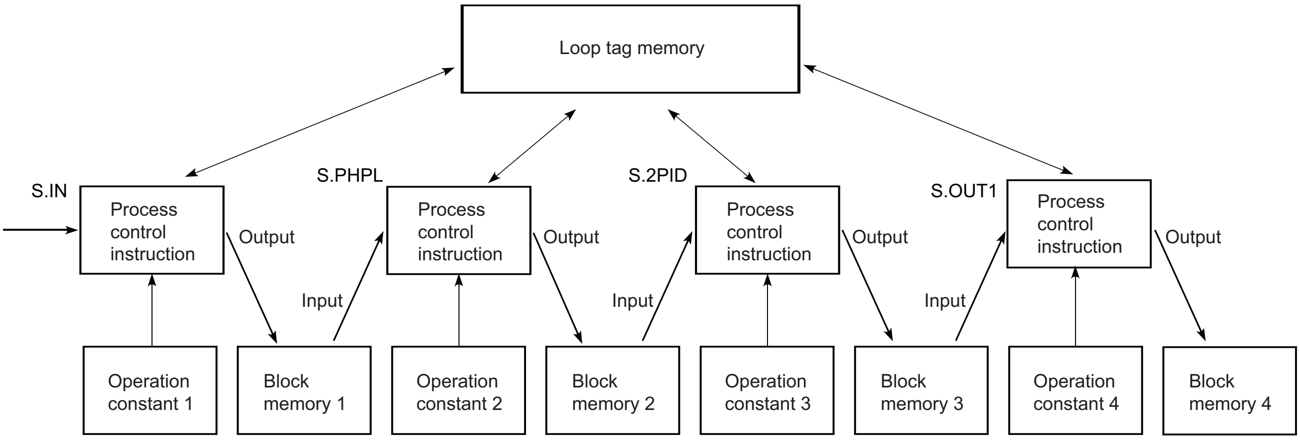
This section describes the data configurations used by process control instructions.

Data configuration in which loop tags are used

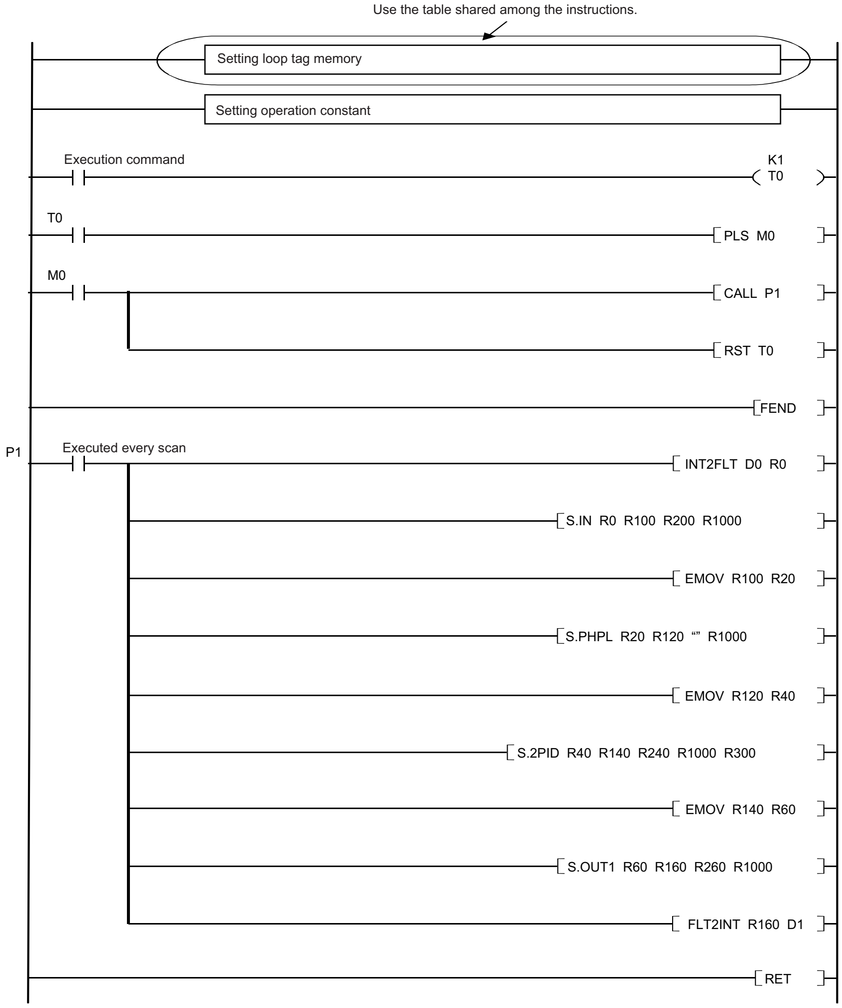
The process control instructions use control information common to each loop by storing it in common memory. The group of this common information is called a loop tag, and the storage memory is called loop tag memory. Loop monitoring and tuning can be performed by monitoring the loop tag.

Ex.

Block diagram of two-degree-of-freedom PID control (S2PID)



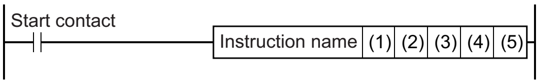
Positioning the loop tag memory and operation constants based on the ladder diagram



- Loop tag memory setting

Loop tag memory (96 words)	Instruction used	Item	Standard value setting	Data type
+0	—	—	—	16-bit unsigned binary
+1	—	MODE	8H	16-bit unsigned binary
+3	—	ALM	4000H	16-bit unsigned binary
+4	—	INH	4000H	16-bit unsigned binary
+10	S.PHPL	PV	0.0	Single-precision real number
+12	S.OUT1	MV	0.0	Single-precision real number
+14	S.2PID	SV	0.0	Single-precision real number
+16	S.2PID	DV	0.0	Single-precision real number
+18	S.OUT1	MH	100.0	Single-precision real number
+20	S.OUT1	ML	0.0	Single-precision real number
+22	S.PHPL	RH	100.0	Single-precision real number
⋮	⋮	⋮	⋮	⋮
+46	S.2PID	CT	1.0	Single-precision real number
+48	S.OUT1	DML	100.0	Single-precision real number
+50	S.2PID	DVL	100.0	Single-precision real number
+52	S.2PID	P	1.0	Single-precision real number
+54	S.2PID	I	10.0	Single-precision real number
+56	S.2PID	D	0.0	Single-precision real number
+58	S.2PID	GW	0.0	Single-precision real number
+60	S.2PID	GG	1.0	Single-precision real number
+62	S.OUT1	MVP	0.0	Single-precision real number
+64	S.2PID	α	0.0	Single-precision real number
+66	S.2PID	β	1.0	Single-precision real number
⋮	⋮	⋮	⋮	⋮
+90	—	—	0.0	Single-precision real number
+92	—	—	0.0	Single-precision real number
+94	—	—	0.0	Single-precision real number

The signs in the ladder diagram mean as follows.



Item	Instruction name			
	S.IN	S.PHPL	S.2PID	S.OUT1
(1) Input data start device	R0	R20	R40	R60
(2) Block memory start device	R100	R120	R140	R160
(3) Operation constant start device	R200	Null character string ("")	R240	R260
(4) Loop tag memory start device	R1000			
(5) Set value start device	—	—	R300	—

Data used by process control instructions

The data used by process control instructions includes the following.

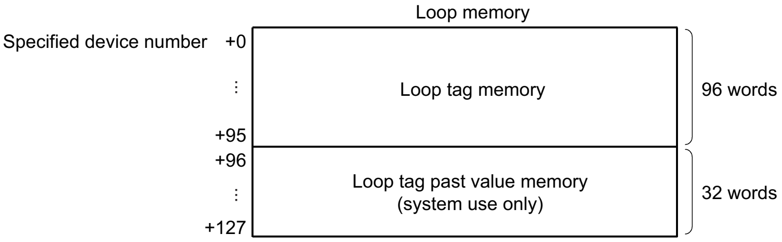
- Loop memory (☞ Page 1186 Loop memory)
- Input data (☞ Page 1187 Input data)
- Block memory (☞ Page 1187 Block memory)
- Operation constant (☞ Page 1188 Operation constant)
- Local work memory (☞ Page 1188 Local work memory)

Loop memory

The loop memory is an area in which the data used commonly by the process control instructions specified by the loop type is stored. The loop memory has also another area in which the data used by the CPU module system during execution of process control instructions is stored.

The loop memory consists of loop tag memory and loop tag past value memory.

- ☞ Page 1186 Loop tag memory
- ☞ Page 1186 Loop tag past value memory



Point

The loop memory is configured with 128 words, and therefore a device which has 128 consecutive words should be specified when the loop memory area is allocated.

■Loop tag memory

The loop tag memory is an area (96 words) in which the control information used commonly by the process control instructions specified by a loop type among the basic loop types (☞ Page 1181 Basic loop types) is stored.

Point

For the applications of the area used by process control instructions in the loop tag memory, refer to the following.

☞ Page 2082 List of Loop Tag Memory Areas Used by Process Control Instructions

■Loop tag past value memory

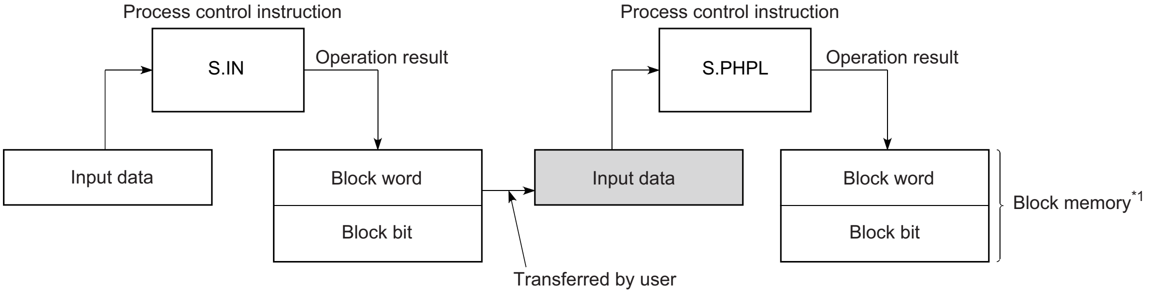
The loop tag past value memory is an area (32 words) used by the system during execution of process control instructions. No data can be written during operation. If data is written to the loop tag past value memory during operation, normal operation is disabled.

Point

When starting a process control instruction, write 0 to the loop tag past value memory.

Input data

Input data is variable data given to each process control instruction. The block word (BW) in the block memory in which the operation result of the previous process control instruction is stored is used as input data.



*1 For the block memory, refer to the following.
☞ Page 1187 Block memory

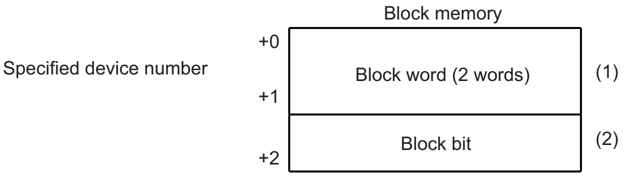
The application of input data varies depending on the instruction used. Refer to the descriptions of individual instructions.

Block memory

The block memory is an area in which the output information of each process control instruction is stored.

The block memory consists of a block word (BW) and a block bit (BB).

- ☞ Page 1187 Block word (BW)
- ☞ Page 1187 Block bit (BB)

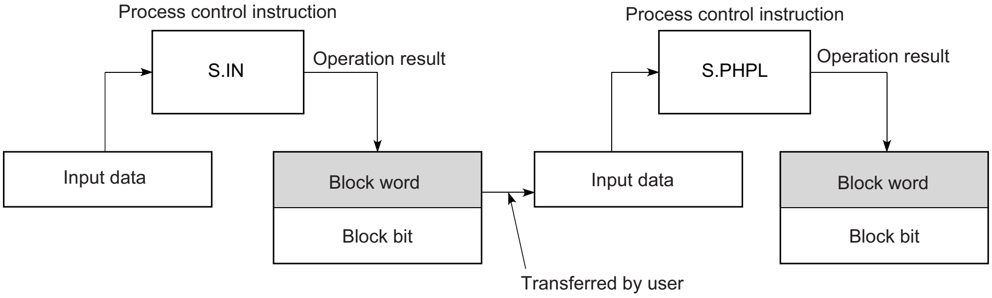


(1) When storing a real number in the block word, use two words.
(2) The block bit is used by setting each bit of one word to on or off.

The application of block memory varies depending on the instruction used. Refer to the descriptions of individual instructions.

■Block word (BW)

The block word (BW) is an area in which the operation result of each process control instruction is stored. The data stored in the block word (BW) is specified for the input data for the next process control instruction connected by the loop.



■Block bit (BB)

The block bit (BB) is an area in which alarm data during execution of each process control instruction is stored. Sixteen bits from b0 to b15 are represented as BB1 to BB16. If an alarm occurs in any of b1 to b15 (BB2 to BB16) used by instructions, 1 is stored in b0 (BB1). The bits that are not used are cleared to zero.

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BB16	BB15	BB14	BB13	BB12	BB11	BB10	BB9	BB8	BB7	BB6	BB5	BB4	BB3	BB2	BB1

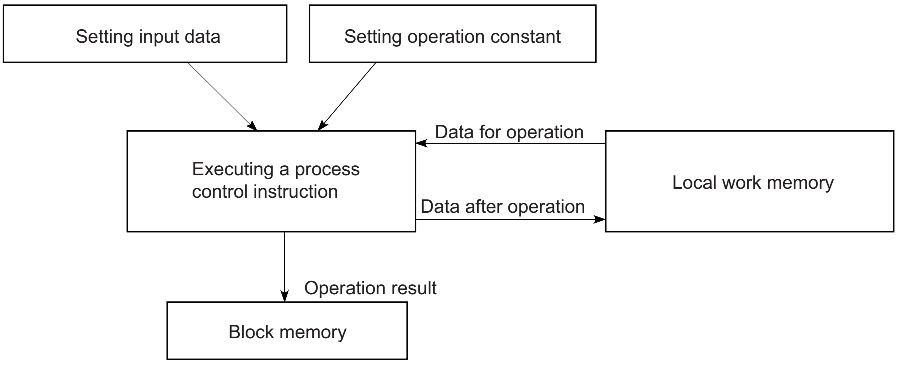
Operation constant

The operation constant is an area in which the data used by only one process control instruction is stored.

The application of the operation constant varies depending on the instruction used. Refer to the descriptions of individual instructions.

Local work memory

The local work memory is an area in which data is temporarily stored during operation of process control instructions.



The application and storage area of the local work memory vary depending on the instruction used. Refer to the descriptions of individual instructions.

Loop tag memory assignments

Loop tag memory assignments are explained below.

Ex.

Loop tag memory assignments for two-degree-of-freedom PID control (S2PID)

Instruction used	Offset[1]	Item	Recommended range	Standard value setting	Data type	Remarks
—	+0	—	—	—	—	Same values are used for all instructions in the same loop tag memory.
—	+1	MODE	0 to FFFFH	8H[2]	16-bit unsigned binary	
—	+3	ALM	0 to FFFFH	4000H[2]	16-bit unsigned binary	
—	+4	INH	0 to FFFFH	4000H	16-bit unsigned binary	
S.PHPL	+10	PV	RL to RH	0.0[2]	Single-precision real number	The offset values are fixed for each instruction.
S.OUT1	+12	MV	-10 to 110	0.0[2]	Single-precision real number	
S.2PID	+14	SV	RL to RH	0.0	Single-precision real number	
S.2PID	+16	DV	-110 to 110	0.0[2]	Single-precision real number	
S.OUT1	+18	MH	-10 to 110	100.0	Single-precision real number	
S.OUT1	+20	ML	-10 to 110	0.0	Single-precision real number	
S.PHPL	+22	RH	-999999 to 999999	100.0	Single-precision real number	
S.PHPL	+24	RL	-999999 to 999999	0.0	Single-precision real number	
S.PHPL	+26	PH	RL to RH	100.0	Single-precision real number	
S.PHPL	+28	PL	RL to RH	0.0	Single-precision real number	
S.PHPL	+30	HH	RL to RH	100.0	Single-precision real number	
S.PHPL	+32	LL	RL to RH	0.0	Single-precision real number	
S.IN	+38	α	0 to 1	0.2	Single-precision real number	
S.PHPL	+40	HS	0 to 999999	0.0	Single-precision real number	
S.PHPL	+42	CTIM	0 to 999999	0.0	Single-precision real number	
S.PHPL	+44	DPL	0 to 999999	100.0	Single-precision real number	
S.2PID	+46	CT	0 to 100	1.0	Single-precision real number	
S.OUT1	+48	DML	0 to 100	100.0	Single-precision real number	
S.2PID	+50	DVL	0 to 100	100.0	Single-precision real number	
S.2PID	+52	P	0 to 999999	1.0	Single-precision real number	
S.2PID	+54	I	0 to 999999	10.0	Single-precision real number	
S.2PID	+56	D	0 to 999999	0.0	Single-precision real number	
S.2PID	+58	GW	0 to 100	0.0	Single-precision real number	
S.2PID	+60	GG	0 to 999999	1.0	Single-precision real number	
S.OUT1	+62	MVP	-999999 to 999999	0.0[2]	Single-precision real number	
S.2PID	+64	α	0 to 1	0.0	Single-precision real number	
S.2PID	+66	β	0 to 1	1.0	Single-precision real number	

[1] This is the number of words from the start of the loop tag.

[2] Values may change depending on the operation result.

For details on the loop tag memory assignments, refer to the following.

☞ Page 2082 List of Loop Tag Memory Areas Used by Process Control Instructions

Common items

This section describes the common items in loop types.

■Alarm detection (ALM)

Alarm detection (ALM) indicates loop alarm information. The default setting is 4000H which indicates manual operation in loop stop state. To enable auto alarm, set it to 0000H.

b15	b14	…		b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	SPA			DMLA	OOPA	SEA	HHA	LLA	PHA	PLA	DPPA	DPNA	DVLA	MHA	MLA

Details of Alarm detection (ALM) are shown below.

Item	Name	Status	Set by
SPA	Stop alarm	Loop STOP state • Loop mode MAN is entered. • Loop stop processing is performed for the output value (BW) and alarm signal.	User
DMLA	Output variation rate limit alarm	As the result of checking input data using the variation rate limiter, the output variation rate limit has been exceeded.	System
OOPA	Output open alarm	The operation output signal is in open state due to disconnection.	System
SEA	Sensor alarm	Sensor error alarm	System
HHA	Upper upper limit alarm	The process value exceeds the upper limit defined for the process equipment.	System
LLA	Lower lower limit alarm	The process value underruns the lower limit defined for the process equipment.	System
PHA	Upper limit alarm	As the result of upper limit checking, the process value exceeds the upper limit.	System
PLA	Lower limit alarm	As the result of lower limit checking, the process value underruns the lower limit.	System
DPPA	Positive direction variation rate alarm	The variation rate exceeds the variation rate range on an upward trend.	System
DPNA	Negative direction variation rate alarm	The variation rate underruns the variation rate range on a downward trend.	System
DVLA	Large deviation alarm	The result of a deviation check shows that the deviation limit is exceeded. (The deviation check determines whether the deviation has been reduced completely underrunning the alarm value. To do so, when the deviation has been reduced to a certain value range from the alarm value, the large deviation alarm is released.)	System
MHA	Output upper limit alarm	As the result of checking with the upper/lower limiter, the value output by the limiter exceeds the output upper limit.	System
MLA	Output lower limit alarm	As the result of checking with the upper/lower limiter, the value output by the limiter underruns the output lower limit.	System

■Disable alarm detection (INH)

This disable alarm detection of each item. The alarms whose detection is disabled by INH are not detected.

INH bits b0 to b11 correspond to ALM bits b0 to b11.

b15	···	b13	···	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ERRI		TRKF		DMLI	OOPI	SEI	HHI	LLI	PHI	PLI	DPPI	DPNI	DVLI	MHI	MLI
(2)		(1)													

(1) Tracking flag (Do not rewrite the tracking flag.)
(2) Alarm detection disabled

■Control mode (MODE)

The process control instructions have the following control modes that satisfy the following operations in a system connected to the operator station, programmable controller, host computer, and machine side operation panel. Set only 1 bit of flag for the control mode (MODE).

b15		···			b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
					CSV	CMV	CCB	CAB	CMB	CAS	AUT	MAN	LCC	LCA	LCM

Details of the control mode (MODE) are given below.

Control mode	Description	Application
MAN (MANUAL)	• Manual operation from operator station • SV and MV can be set.	Used for monitoring and control from the operator station.
AUT (AUTOMATIC)	• Automatic operation • SV can be set. • MV cannot be set.	
CAS (CASCADE)	• Cascade operation • SV nor MV cannot be set.	
CMV (COMPUTER MV)	Automatic MV setting from host computer	Loop control from the host computer is possible. Used to operate and monitor the control mode on the operator station.
CSV (COMPUTER SV)	Automatic SV setting from host computer	

Control mode	Description	Application
CMB (COMPUTER MANUAL BACK UP)	Manual operation backup when the host computer is abnormal	When the computer fails during loop control by the host computer, backup is performed by the predetermined operator station.
CAB (COMPUTER AUTOMATIC BACK UP)	Automatic operation backup performed when the host computer is abnormal	
CCB (COMPUTER CASCADE BACK UP)	Cascade operation backup performed when the host computer is abnormal	
LCM (LOCAL MANIPULATED)	Local manual operation	When the plant is started, the operation and start-up are performed by using equipment such as the machine side operation panel away from the operator station, and the control mode is monitored at the operator station.
LCA (LOCAL AUTOMATIC)	Local automatic operation	
LCC (LOCAL CASCADE)	Local cascade operation	

How to execute process control instructions

Execution cycle and control cycle

■Execution cycle

The execution cycle is an interval at which process control instructions are executed.

The following methods can be used to execute process control instructions in each execution cycle.

Execution method	Description
Timer	A timer is used to measure the execution cycle and a process control instruction is executed when the time of the timer is up.
Interrupt program	Interrupt programs I28 to I31 are each executed every execution cycle.
Fixed scan execution type program	Fixed scan execution type programs are each executed every execution cycle.

Point

Specify the execution cycle value used by process control instructions in SD816 and SD817 in a single-precision real number.

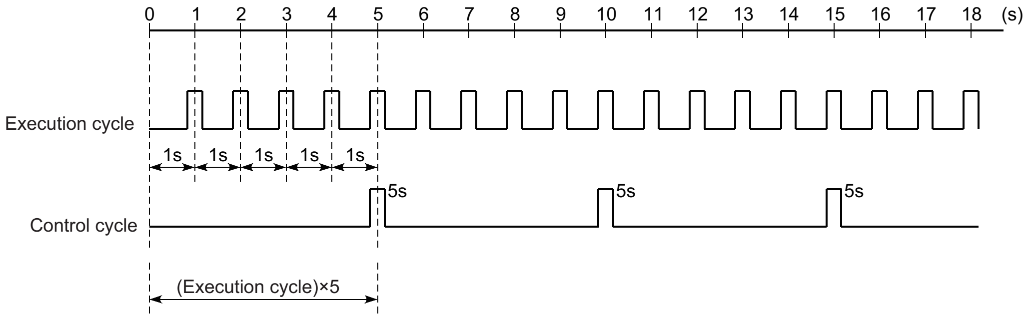
■Control cycle

The control cycle is a cycle in which PID control is performed by instructions such as S.2PID. For the control cycle, specify an integral multiple of the execution cycle. Execution cycles are counted during execution, and PID operation is performed when the specified control cycle is reached.

Specify the control cycle used in the loop tag memory. Instructions such as S.2PID perform PID control based on the control cycle value specified in the loop tag memory.

Ex.

The S.2PID instruction performs monitoring every second and implements PID control every 5 seconds.



Point

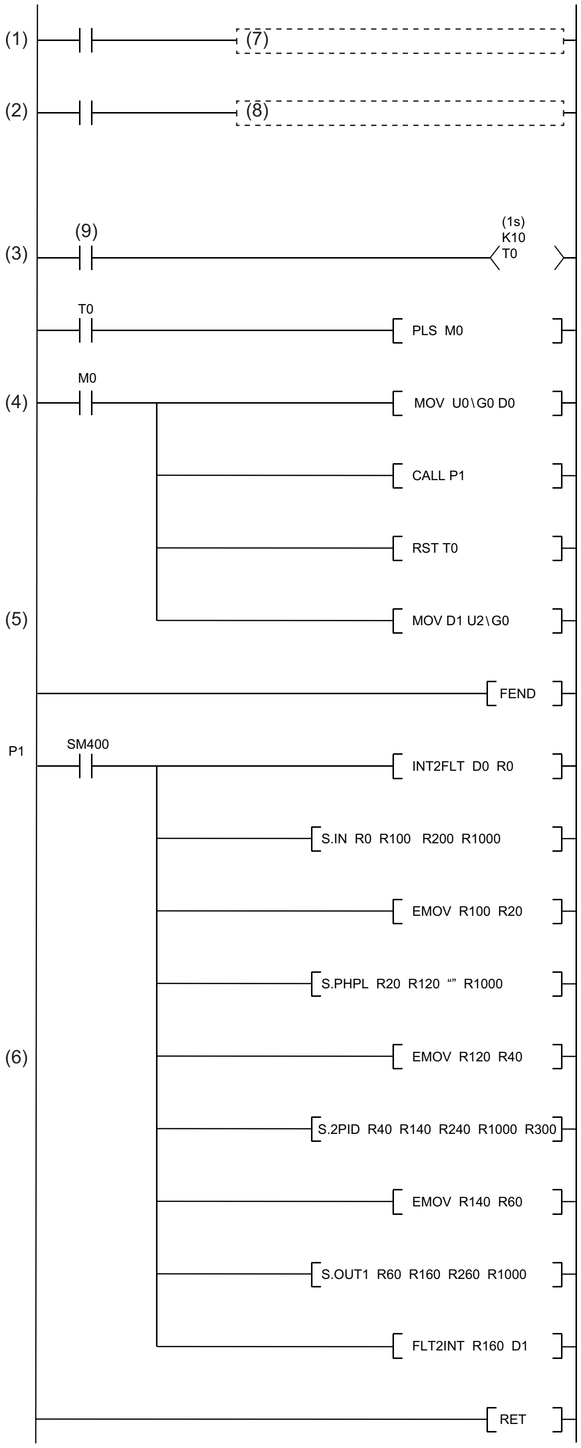
Setting the control cycle to an integral multiple of the execution cycle enables monitoring such as for checking the process value range every execution cycle.

Concept of program

This section describes the concept of programs using process control instructions.

Ex.

Program example using the S.2PID instruction in an execution cycle of 1 second



(1) Setting loop tag memory
(2) Setting operation constants
- Setting data for S.IN, S.PHPL, S.2PID, and S.OUT1

(3) Measuring the execution cycle
(4) Setting input data (PV)
- Reading PV from such as the A/D conversion module

(5) MV output
- Outputting MV from such as the D/A conversion module

(6) Specifying process control instructions
- S.IN instruction
- S.PHPL instruction
- S.2PID instruction
- S.OUT1 instruction

(7) Setting loop tag memory
(8) Setting operation constants
(9) Executing a command

For specific program examples using process control instructions, refer to the following.

☞ Page 2077 Process Control Program Examples

Execution condition switching

Loop RUN/STOP

If any loop component such as a detector or operation terminal other than the programmable controller fails, each loop can be run and stopped independently for the purpose of maintenance. SPA of the alarm detection (ALM) is used to run/stop the applicable loop.

■Basic operation to stop a loop

- Output status is retained. (Example: Output of S.2PID instruction = 0)
- No alarm is detected.
- The control mode is MAN.

Tracking

Tracking refers to making a certain signal follow and match another signal.

Tracking function

The tracking function used by process control instructions includes the bumpless function and output limiter processing function.

■Bumpless function

The bumpless function prevents manipulated value (MV) output stepping changes when switching from the automatic mode to manual mode, and continuously and smoothly controls MV output.

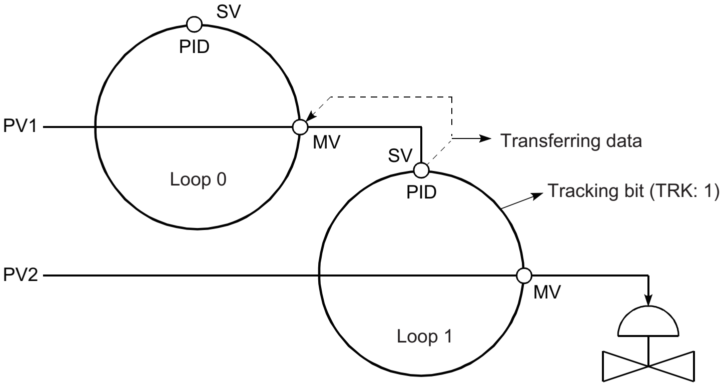
■Output limiter processing function

The output limiter processing function limits the upper or lower limit of the manipulated value (MV) output by the PID operation in automatic mode. This output limiter processing function is only valid in automatic mode and is not executed in manual mode. The output limiter processing function is not executed either even in automatic mode if the tracking bit (TRK) is set to 0 (Disable tracking).

Cascade loop tracking

The process control loops making up a cascade loop use the manipulated value (MV) of a primary loop (Loop 0) as the set value (SV) of a secondary loop (Loop 1). Tracking is performed to prevent a sudden change in the set value (SV) when the control mode of the secondary loop (Loop1) is changed.

Cascade loop tracking processing is outlined below.



In cascade operation, the manipulated value (MV) of Loop 0 is transferred to the set value (SV) of Loop 1. When cascade operation is not performed, the set value (SV) of Loop 1 is transferred to the manipulated value (MV) of Loop 0. (Tracking to the source specified as the input terminal of the set value (SV) of Loop 1)

Tracking is performed when the control mode is switched to one other than CAS, CSV, or CCB.

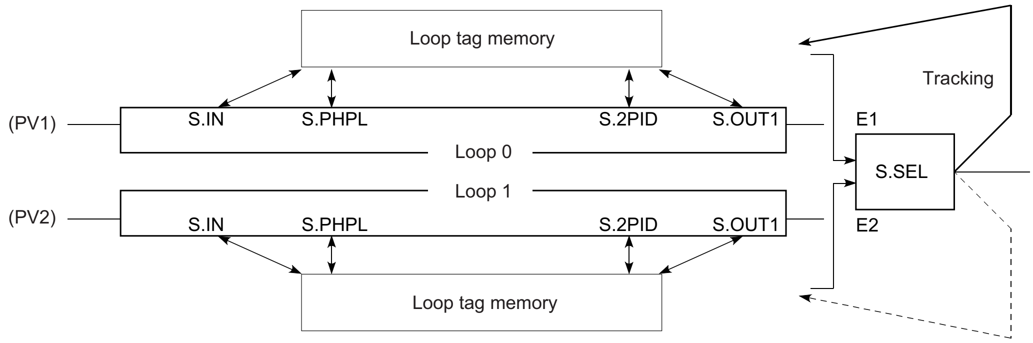
For S.2PID (two-degree-of-freedom PID control), set the following operation constant items to specify tracking.

Setting item		Setting
Tracking bit (TRK)		1 (Tracking performed)
Set value pattern (SVPTN)	Set value pattern	0 (Set value is upper loop MV.)
	Set value used	0 (E2 is specified.)

Loop selector tracking

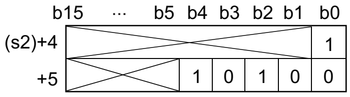
Tracking processing is performed under the following conditions.

- The control mode is MAN, CMB, CMV, or LCM, and the tracking bit (TRK) is 1.
- The control mode is AUT, CAS, CAB, CCB, CSV, LCA, or LCC, and the tracking bit (TRK) is 1 and BB1 of the block bit (BB) is 1.



Ex.

When the input value (E1) of the S.SEL instruction uses the manipulated value (MV) of the upper loop (Loop 0), the manipulated value (MV) of the S.SEL instruction is tracked to the manipulated value (MV) of Loop 0. The S.SEL instruction specifies tracking according to the following operation constant items.



Operation constant		Bit position	Stored value	
(s2)+4	Tracking bit	b0	0: Tracking not performed 1: Tracking performed	
(s2)+5	Set value pattern	b0	e1	0: E1 1: E2
		b1	E1	0: Use 1: Not use
		b2	E2	0: Use 1: Not use
		b3	E1	0: E1 is the upper loop MV. 1: E1 is not the upper loop MV.
		b4	E2	0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.

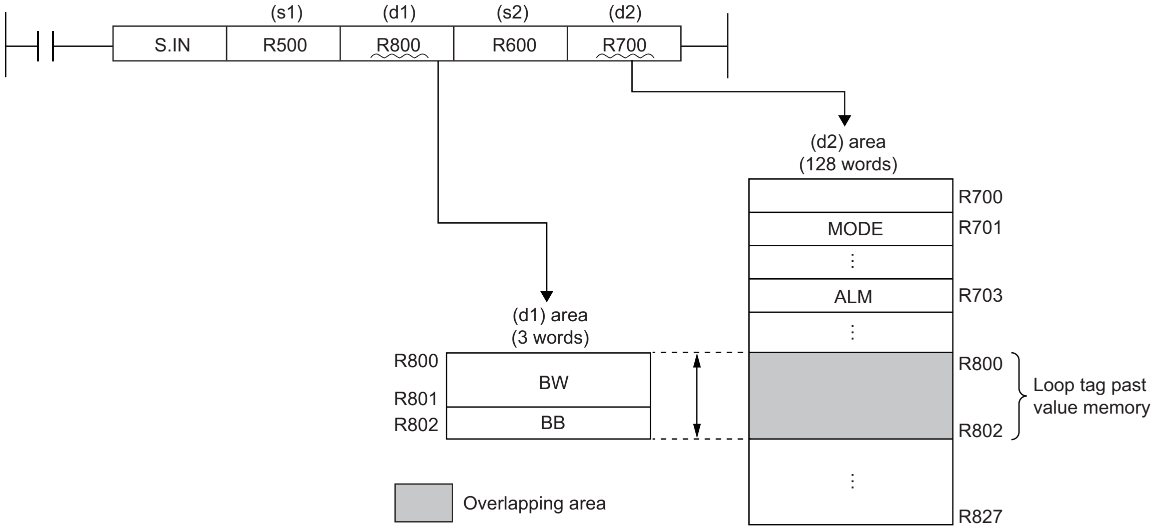
Precautions

Overlapping of specified data areas

The process control instructions check for area overlapping of input and output data. Specify input and output data after checking their areas are not overlapped.

Ex.

Example of area overlapping caused by a process control instruction



In the above example, the areas of (d1) and (d2) specified in the setting data are overlapping and an error results.

Operand specification

Regardless of the program language used, specify devices for the operations of process control instructions. Do not specify labels.

Errors of process control instructions

When an error occurs in process control instructions, the detailed information of the error is stored in SD81 to SD111 (detailed information 1) and SD113 to SD143 (detailed information 2). For the information to be stored, refer to the following.

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

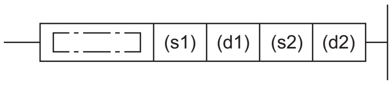
11.2 I/O Control Instructions

Analog input processing

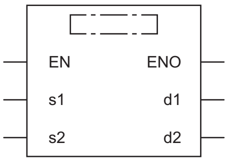
S.IN

RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

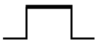
This instruction performs following processing to the input data (PV): range check, input limiter, engineering value transformation, and digital filter.

Ladder	ST
	ENO:=S_IN(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.IN	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

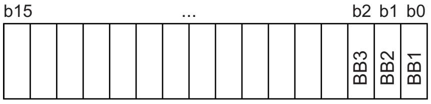
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

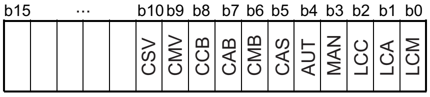
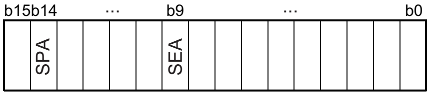
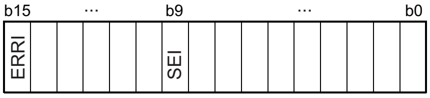
Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Input upper limit alarm				
		BB3	Input lower limit alarm				

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	EMAX	Engineering value transformation upper limit	-999999 to 999999 [%]	Single-precision real number	100.0	User
+2 +3	EMIN	Engineering value transformation lower limit	-999999 to 999999 [%]	Single-precision real number	0.0	User
+4 +5	NMAX	Input upper limit	-999999 to 999999	Single-precision real number	100.0	User
+6 +7	NMIN	Input lower limit	-999999 to 999999	Single-precision real number	0.0	User
+8 +9	HH	Upper limit range error occurrence	-999999 to 999999	Single-precision real number	110.0	User
+10 +11	H	Upper limit range error return	-999999 to 999999	Single-precision real number	100.0	User
+12 +13	L	Lower limit range error return	-999999 to 999999	Single-precision real number	0.0	User
+14 +15	LL	Lower limit range error occurrence	-999999 to 999999	Single-precision real number	-10.0	User

■Loop tag memory

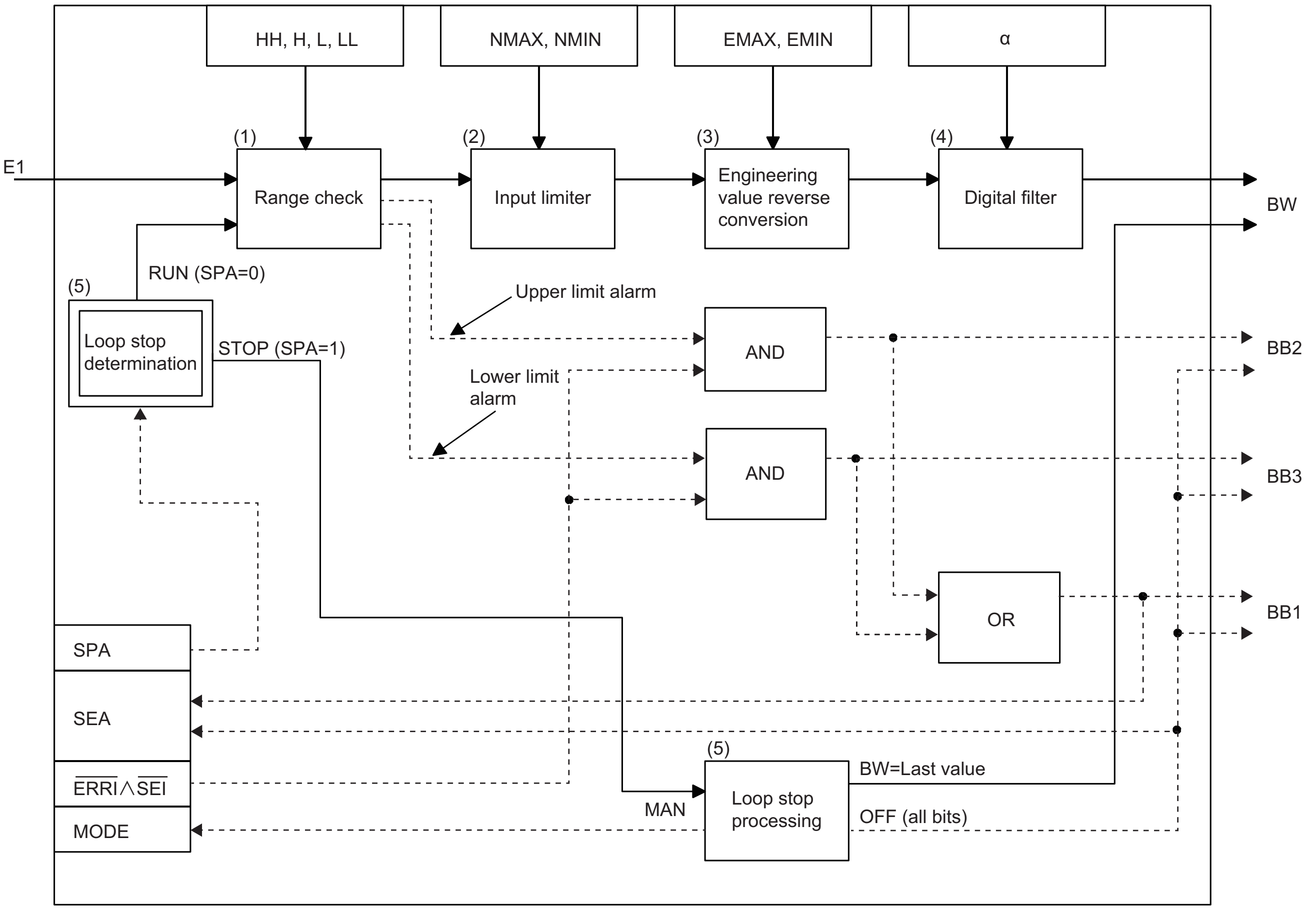
The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP SEA 0: No alarm 1: Alarm	16-bit unsigned binary	0H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	0H	User/ system
+38 +39	α	Filter coefficient	0 to 1	Single-precision real number	0.2	User

Processing details

This instruction performs engineering value transformation of the input value (E1) in the device specified by (s1), and stores the result in the device specified by (d1). The instruction also performs input value (E1) range check, input limiter, and digital filter processing.

The following is the processing block diagram of the S.IN instruction. (The numbers (1) to (5) in the diagram indicate the order of the processing.)



■Range check (1)

- The instruction checks the range of the input value (E1).

If the input value (E1) goes beyond the upper or lower limits, an alarm is output.

Range check	Condition	Range check result (alarm output)		
		BB2	BB3	BB1, SEA
Upper limit check	E1 ≥ HH	1[*1]	—	1[*1]
	E1 ≤ H	0	—	0
	H < E1 < HH	Last value	—	Last value
Lower limit check	E1 ≤ LL	—	1[*1]	1[*1]
	E1 ≥ L	—	0	0
	LL < E1 < L	—	Last value	Last value

*1 When ERRI or SEI of Disable alarm detection (INH) is set to 1, alarm output is disabled and therefore ALM SEA, BB2, and BB3 are set to 0.

- Last value hold processing

When a range excess occurs (BB1 = 1) in the range check, whether to continue operation or terminate the S.IN instruction is determined by whether SM816 is on or off.

Condition	Description
SM816 is off (not in hold mode)	"Input limiter (2)" is performed even if a range excess occurs (BB1 = 1).
SM816 is on (in hold mode)	If a range excess occurs (BB1 = 1), the following processing is performed to terminate the S.IN instruction. • The last output value (BW) is held. • Error information is stored in BB.

■Input limiter (2)

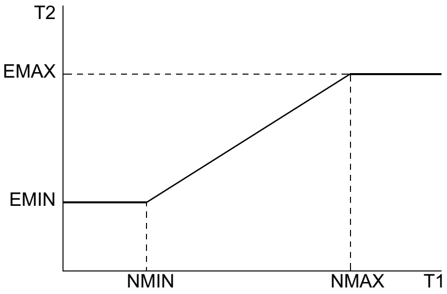
The instruction sets the upper and lower limits for the input value (E1).

Condition	Result (T1)
E1 ≥ NMAX	NMAX
E1 ≤ NMIN	NMIN
NMIN < E1 < NMAX	E1

■Engineering value inverse transformation (3)

The instruction performs engineering value inverse transformation of the input limiter result (T1) according to the following expression.

$$T2=(EMAX-EMIN)\times\frac{T1-NMIN}{NMAX-NMIN}+EMIN$$



■Digital filter (4)

The instruction applies a digital filter to the input value (E1) according to the following expression. The digital filter is used to reduce the effect of noise.

$BW=T2+\alpha\times(\text{last BW value}-T2)$

■Loop stop processing (5)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	Performs the following operations and terminates the S.IN instruction. • The last output value (BW) is held. • The control mode (MODE) is set to MAN. • All of the alarm bits (BB1, BB2, and BB3) are set to 0. • ALM SEA is set to 0.
0	Performs "range check (1)".

Operation error

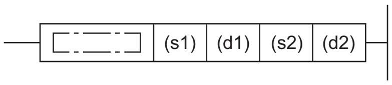
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (d1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Upper limit range error occurrence (HH) < upper limit range error return (H), lower limit range error return (L) < lower limit range error occurrence (LL), or input upper limit (NMAX) < input lower limit (NMIN)

Output processing 1 with mode switching

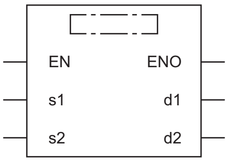
S.OUT1

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

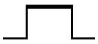
This instruction calculates MV (0 to 100%) from input data (ΔMV), performs variation rate & upper/lower limiter processing, and output conversion.

Ladder	ST
	ENO:=S_OUT1(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.OUT1	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

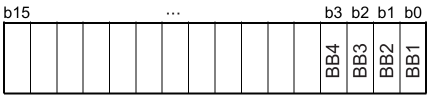
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value (ΔMV)	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

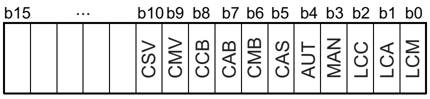
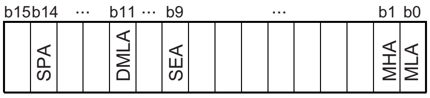
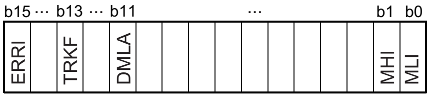
Operand: (d1)							
Position	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Output upper limit alarm				
		BB3	Output lower limit alarm				
		BB4	Output variation rate alarm				

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	NMAX	Output conversion upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	NMIN	Output conversion lower limit	-999999 to 999999	Single-precision real number	0.0	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP DMLA, SEA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  TRKF 0: Tracking not set 1: Tracking set ERRI, DMLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User

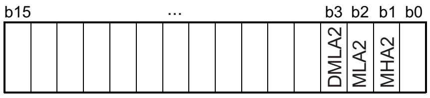
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+48 +49	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	100.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
(d2)+116	ALM2	Alarm detection 2	 MHA2, MLA2, DMLA2 0: No alarm 1: Alarm	—	—	System

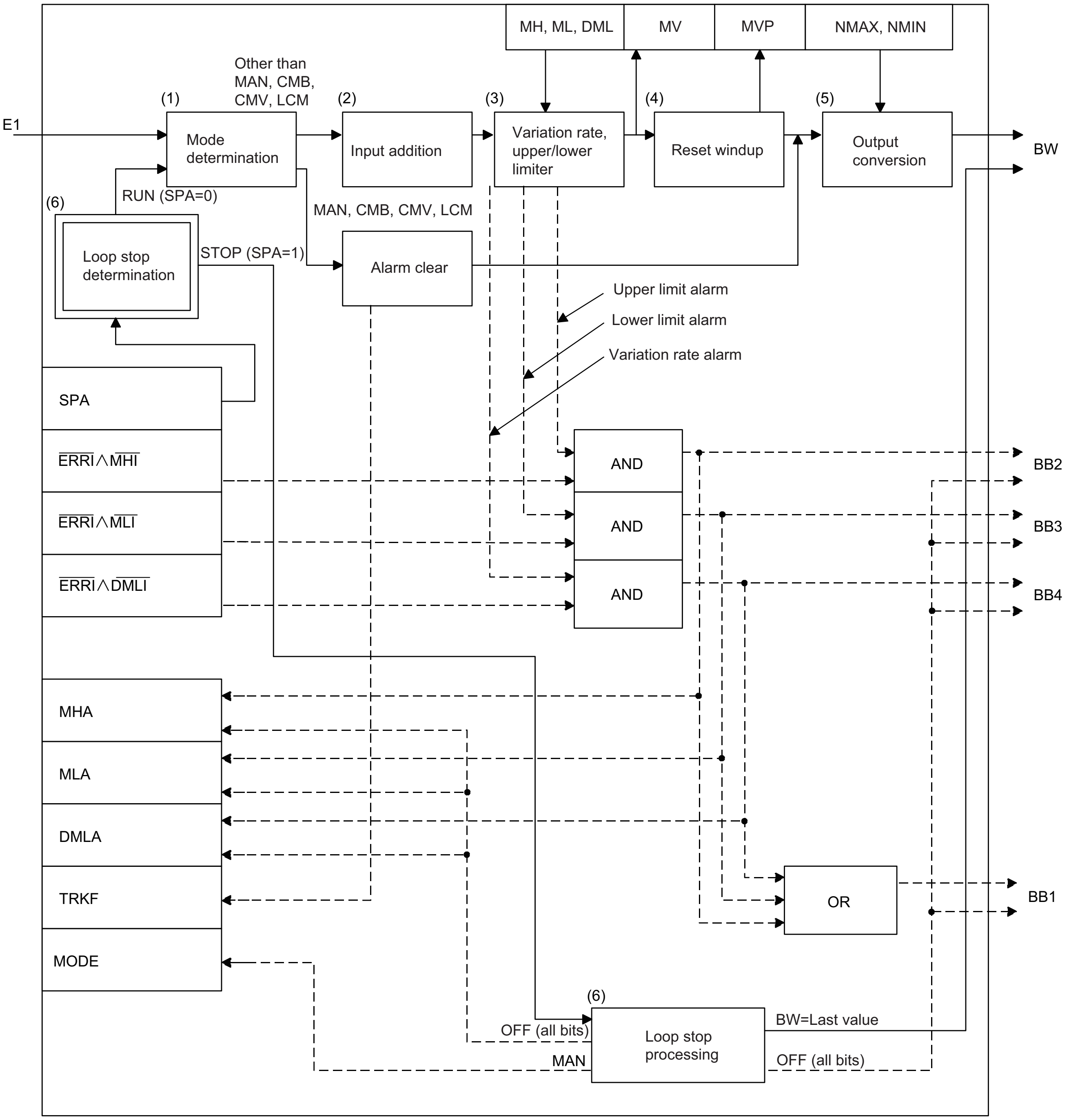
■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction calculates the manipulated value (MV) from the input value (E1=ΔMV) in the device specified by (s1), and stores the result in the device specified by (d1). The instruction also performs variation rate & upper/lower limiter, reset windup, and output conversion processing of the calculated manipulated value (MV).

The following is the processing block diagram of the S.OUT1 instruction. (The numbers (1) to (6) in the diagram indicate the order of the processing.)



■Mode determination (1)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM (alarm clear processing)	• The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0. • TRKF of INH is set to 1. • "Output conversion processing (5)" is performed and the instruction ends.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	"Input addition processing (2)" is performed. However, when ALM SEA is 1 and SM817 is on, alarm bits BB1, BB2, BB3, and BB4 are set to 0 and the S.OUT1 instruction is terminated.

■Input addition processing (2)

A temporary MV(T) is calculated on the basis of the input value (E1=ΔMV). The following processing is performed depending on the TRKF of INH.

Tracking flag (TRKF)	Processing details
1	• The manipulated value (MV) is stored in the MV internal operation value (MVP). • The input value (E1) is set to 0. (ΔMV=0) • TRKF of INH is set to 0. • A temporary MV(T) is calculated according to the following expression. T=E1+MVP MVP=T
0	A temporary MV(T) is calculated according to the following expression. T=E1+MVP MVP=T

■Variation rate & upper/lower limiter (3)

The variation rate and upper/lower limits of the input value (E1=ΔMV) are checked, and the data after the processing and an alarm are output.

- Variation rate limiter processing performs the following operations, and outputs the result to the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM).

Condition	BB4, DMLA	Result (T1)
$\|T-MV\| \leq DML$	0	T
(T - MV) > DML	1[*1]	MV+DML
(T - MV) < -DML	1[*1]	MV-DML

*1 If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM) are set to 0.

- Upper/lower limiter processing performs the following operations, and outputs the result to the output upper limit alarm (BB2); output lower limit alarm (BB3); MHA and MLA of the alarm detection (ALM); and MHA2 and MLA2 of the alarm detection (ALM2).

Condition	BB3, MLA, MLA2	BB2, MHA, MHA2	MV
T1 > MH	0	1[*2]	MH
T1 < ML	1[*3]	0	ML
$ML \leq T1 \leq MH$	0	0	T1

*2 If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB2) and the MHA of the alarm detection (ALM) are set to 0.
Note that the MHA2 of the alarm detection 2 (ALM2) remains 1.

*3 If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB3) and the MLA of the alarm detection (ALM) are set to 0.
Note that the MHA2 of the alarm detection 2 (ALM2) remains 1.

■Reset windup (4)

If the manipulated value (MV) goes beyond the upper or lower limit, the following operation is performed to return it to the upper or lower limit and enable immediate response when the deviation is inverted. However, when the integral constant (I) is 0, reset windup processing is not performed.

Condition	Operational expression
$T1>MH,\ \frac{\triangle T}{I}<=1$	$MVP=(\frac{\triangle T}{I})(MH-T)+T$
$T1<MH,\ \frac{\triangle T}{I}<=1$	$MVP=(\frac{\triangle T}{I})(ML-T)+T$

■Output conversion processing (5)

The output value (BW) is calculated from the following expression.

$$BW=\frac{NMAX-NMIN}{100}\times MV+NMIN$$

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.OUT1 instruction ends. • The last output value (BW) is held. • The DMLA, MHA, and DLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The control mode (MODE) is set to MAN. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0.
0	The loop runs and "mode determination (1)" is performed.

■Hold processing (7)

This processing specifies whether to hold the output value (BW) by the S.OUT1 instruction when a sensor error occurs (detected by the S.IN instruction). The hold processing is performed when the value is determined as RUN by "Loop Stop Determination". SM817 is used to specify whether to hold the manipulated value (MV) when a sensor error occurs.

- SM817 = OFF: Do not hold the manipulated value (MV).
- SM817 = ON: Hold the manipulated value (MV).

Operation error

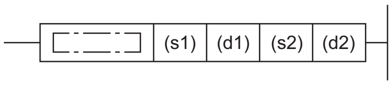
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Output processing 2 with mode switching

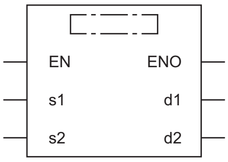
S.OUT2

RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

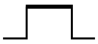
This instruction performs variation rate & upper/lower limiter processing and output conversion on the basis of input data (MV).

Ladder	ST
	ENO:=S_OUT2(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.OUT2	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value (MV)	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)

Position	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB	BB1	Alarm	b15 … b3 b2 b1 b0 BB4 BB3 BB2 BB1 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Output upper limit alarm				
		BB3	Output lower limit alarm				
		BB4	Output variation rate alarm				

■Operation constant

Operand: (s2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	NMAX	Output conversion upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	NMIN	Output conversion lower limit	-999999 to 999999	Single-precision real number	0.0	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

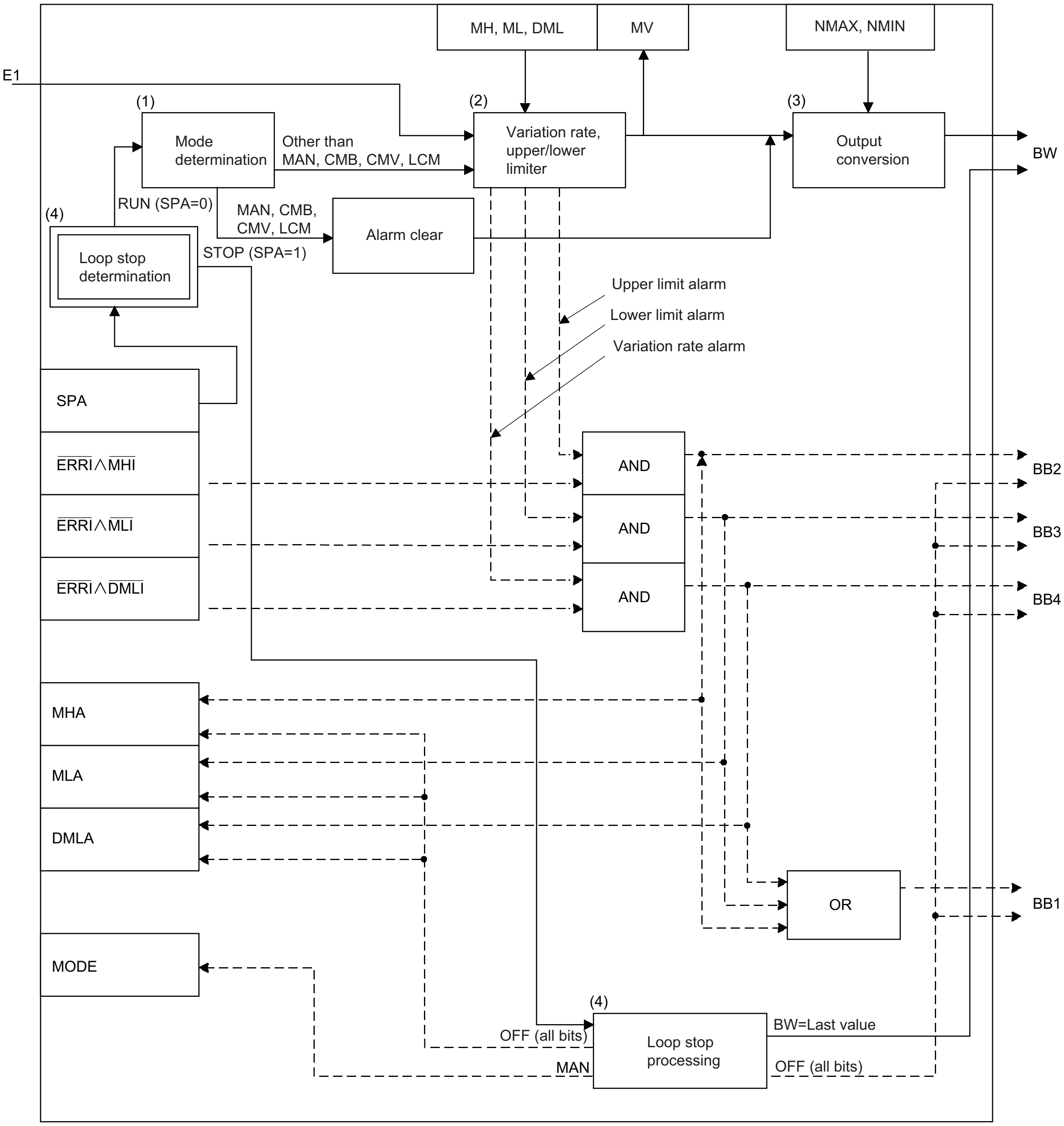
Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b11 … b9 … b1 b0 SPA DMLA SEA MHA MLA SPA 0: Loop RUN 1: Loop STOP DMLA, SEA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b11 … b1 b0 ERRI DMLI MHI HLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+48 +49	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	100.0	User

Processing details

This instruction performs output conversion of the input value (E1=MV) in the device specified by (s1), and stores the result in the device specified by (d1). The instruction also performs variation rate & upper/lower limiter and output conversion processing of the input value at that time.

The following is the processing block diagram of the S.OUT2 instruction. (The numbers (1) to (4) in the diagram indicate the order of the processing.)



■Mode determination (1)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM (alarm clear processing)	• The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0. • "Output conversion processing (3)" is performed and the instruction ends.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	"Variation rate & upper/lower limiter processing (2)" is performed. However, when ALM SEA is 1 and SM817 is on, alarm bits BB1, BB2, BB3, and BB4 are set to 0 and the S.OUT2 instruction is terminated.

■Variation rate & upper/lower limiter (2)

The variation rate and upper/lower limits of the input value (E1=∆MV) are checked, and the data after the processing and an alarm are output.

- Variation rate limiter processing performs the following operations, and outputs the result to the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM).

Condition	BB4, DMLA	Result (T1)
|E1-MV|≤DML	0	E1
(E1-MV) > DML	1[*1]	MV + DML
(E1-MV) <- DML	1[*1]	MV - DML

*1 If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM) are set to 0.

- Upper/lower limiter processing performs the following operations, and outputs the result to the output upper limit alarm (BB2); output lower limit alarm (BB3); and MHA and MLA of the alarm detection (ALM).

Condition	BB3, MLA	BB2, MHA	MV
T1 > MH	0	1[*2]	MH
T1 < ML	1[*3]	0	ML
ML ≤ T1 ≤ MH	0	0	T1

*2 If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB2) and the MHA of the alarm detection (ALM) are set to 0.

*3 If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB3) and the MLA of the alarm detection (ALM) are set to 0.

■Output conversion processing (3)

The output value (BW) is calculated from the following expression.

$$\text{BW}=\frac{\text{NMAX-NMIN}}{100}\times\text{MV+NMIN}$$

■Loop stop processing (4)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.OUT2 instruction ends. • The last output value (BW) is held. • The DMLA, MHA, and DLA of the alarm detection (ALM) are set to 0. • The control mode (MODE) is set to MAN. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0.
0	The loop runs and "mode determination (1)" is performed.

■Hold processing (5)

This processing specifies whether to hold the output value (BW) by the S.OUT2 instruction when a sensor error occurs (detected by the S.IN instruction). The hold processing is performed when the value is determined as RUN by "Loop Stop Determination". SM817 is used to specify whether to hold the manipulated value (MV) when a sensor error occurs.

- SM817 = OFF: Do not hold the manipulated value (MV).
- SM817 = ON: Hold the manipulated value (MV).

Operation error

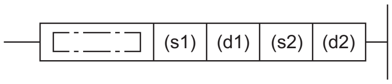
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Manual output

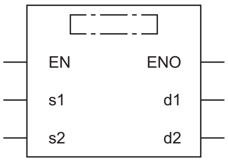
S.MOUT

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction reads the manipulated value (MV) from the loop tag memory and performs output conversion.

Ladder	ST
	ENO:=S_MOUT2(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.MOUT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	—	—	—	—	—	—	—	—	○	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	NMAX	Output conversion upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	NMIN	Output conversion lower limit	-999999 to 999999	Single-precision real number	0.0	User

■Loop tag memory

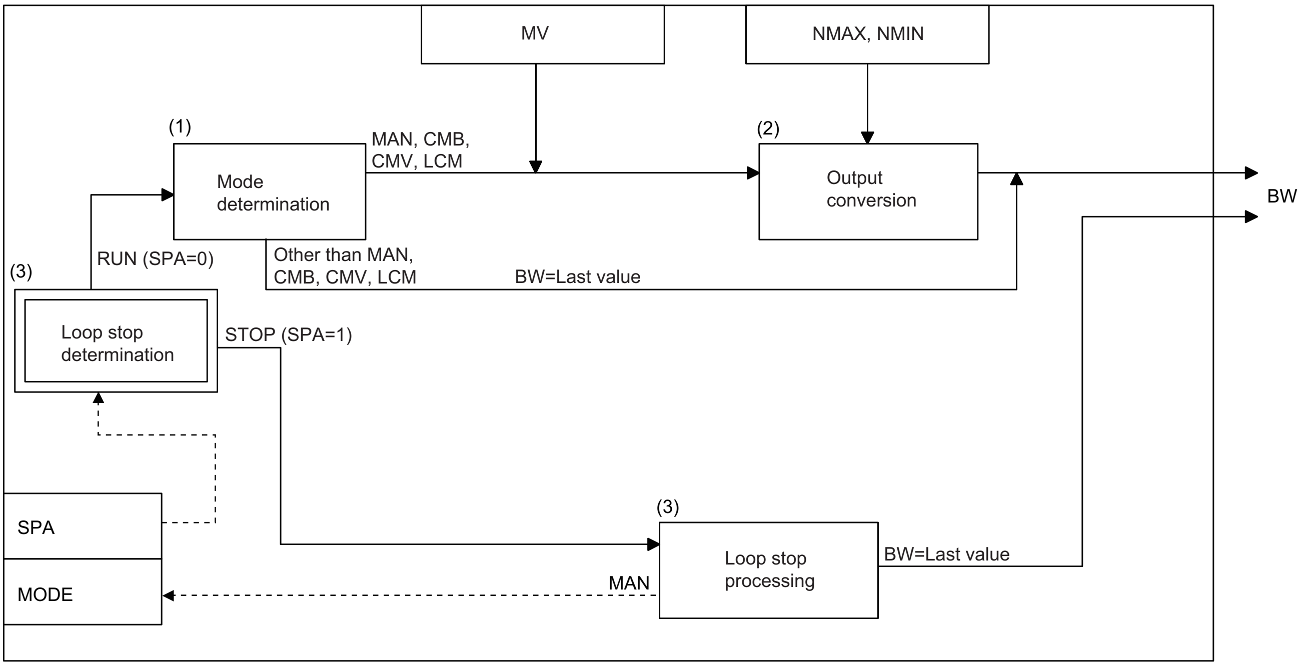
The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 b10: CSV, b9: CMV, b8: CCB, b7: CAB, b6: CMB, b5: CAS, b4: AUT, b3: MAN, b2: LCC, b1: LCA, b0: LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b0 b14: SPA SPA 0: Loop RUN 1: Loop STOP	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User

Processing details

This instruction performs output conversion of the manipulated value (MV) in the device specified by (d2), and stores the result in the device specified by (d1).

The following is the processing block diagram of the S.MOUT instruction. (The numbers (1) to (3) in the diagram indicate the order of the processing.)



■Mode determination (1)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM	• The manipulated value (MV) is used for the output value (BW). • "Output conversion processing (2)" is performed.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	The last output value (BW) is held.

■Output conversion processing (2)

The output value (BW) is calculated from the following expression.

$$BW=\frac{NMAX-NMIN}{100}\times MV+NMIN$$

■Loop stop processing (3)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
0	Performs "Mode determination (1)".
1	Performs the following operations and terminates the S.MOUT instruction. • The last output value (BW) is held. • The control mode (MODE) is set to MAN.

Operation error

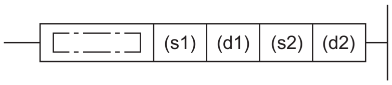
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s2) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Time proportioning

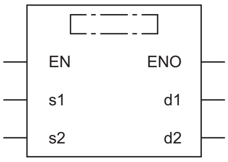
S.DUTY

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

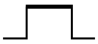
This instruction outputs ON and OFF by changing the ON/OFF ratio in a given cycle in proportion to the input data (0 to 100%).

Ladder	ST
	ENO:=S_DUTY(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DUTY	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

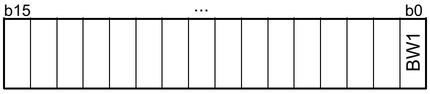
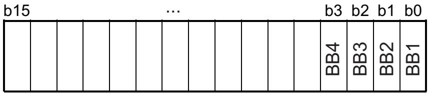
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value (ΔMV)	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

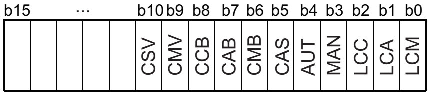
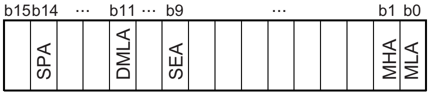
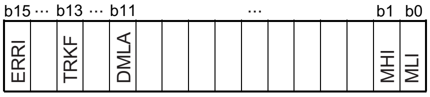
Operand: (d1)

Position	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0	BW	BW1	Output bit	 0: Off 1: ON	16-bit unsigned binary	—	System
+1	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Output upper limit alarm				
		BB3	Output lower limit alarm				
		BB4	Output variation rate alarm				

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP DMLA, SEA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  TRKF 0: Tracking not set 1: Tracking set ERRI, DMLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+48 +49	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	100.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

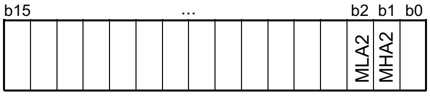
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+68 +69	CTDUTY	Control output cycle	0 to 999999 [s] Set a value within the following range. $\frac{CTDUTY}{\triangle T}<=32767$	Single-precision real number	1.0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm	—	—	System
+118	—	Control output cycle counter initialization completion flag	—			
+119		Control output cycle counter (The value is rounded off to the nearest whole number.)				
+120		Output counter				
+121		Output ON counter				

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

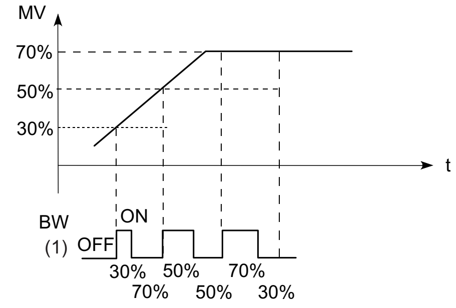
This instruction calculates the manipulated value (MV) from the input value (E1=ΔMV) in the device specified by (s1) by performing input addition processing.

The instruction also turns ON or OFF the device specified by (d1) in proportion to the manipulated value (MV).

The ON/OFF time is a value determined by assuming the time specified by the control output cycle (CTDUTY) as 100%.

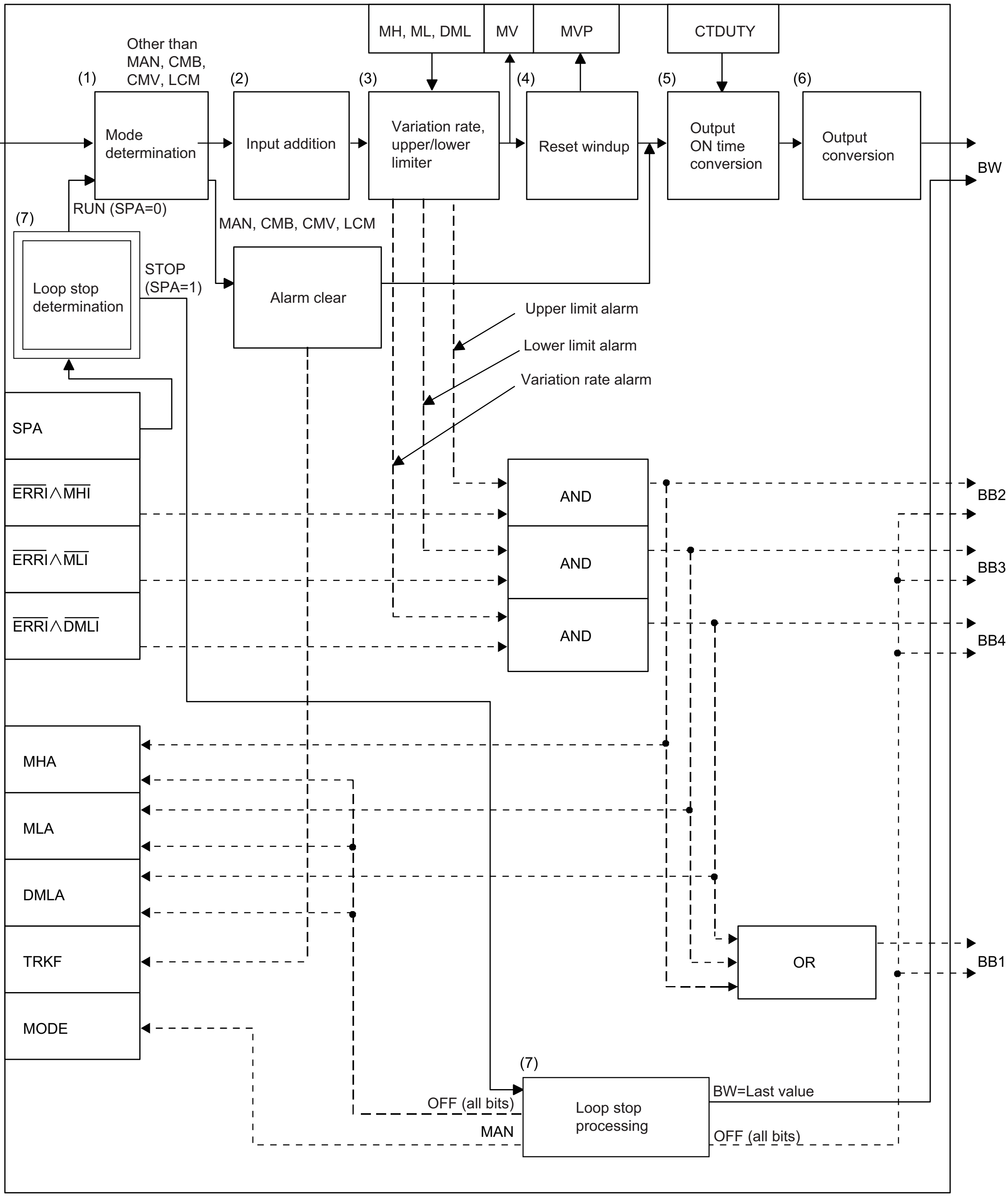
The ON/OFF time is switched every execution cycle.

The instruction also performs variation rate & upper/lower limiter and reset windup of the calculated manipulated value (MV).



(1) Device specified by (d1)

The following is the processing block diagram of the S.DUTY instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■Mode determination (1)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM (alarm clear processing)	• The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0. • TRKF of INH is set to 1. • "Output ON time conversion processing (5)" is performed.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	"Input addition processing (2)" is performed. However, when ALM SEA is 1 and SM817 is on, alarm bits BB1, BB2, BB3, and BB4 are set to 0 and the S.DUTY instruction is terminated.

■Input addition processing (2)

A temporary MV(T) is calculated on the basis of the input value (E1=ΔMV). The following processing is performed depending on the TRKF of INH.

Tracking flag (TRKF) status	Processing details
1	• The manipulated value (MV) is stored in the MV internal operation value (MVP). • The input value (E1) is set to 0. (ΔMV=0) • TRKF of INH is set to 0. • A temporary MV(T) is calculated according to the following expression. T=E1+MVP MVP=T
0	A temporary MV(T) is calculated according to the following expression. T=E1+MVP MVP=T

■Variation rate & upper/lower limiter (3)

Variation rates and upper/lower limits are checked for the difference between the temporary MV(T) and manipulated value (MV), and the data after limiter processing and an alarm are output.

- Variation rate limiter processing performs the following operations, and outputs the result to the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM).

Condition	BB4, DMLA	Result (T1)
|T-MV| ≤ DML	0	T
(T - MV) > DML	1[*1]	MV+DML
(T - MV) < -DML	1[*1]	MV-DML

*1 If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM) are set to 0.

- Upper/lower limiter processing performs the following operations, and outputs the result to the output upper limit alarm (BB2); output lower limit alarm (BB3); MHA and MLA of the alarm detection (ALM); and MHA2 and MLA2 of the alarm detection (ALM2).

Condition	BB3, MLA, MLA2	BB2, MHA, MHA2	MV
T1 > MH	0	1[*2]	MH
T1 < ML	1[*3]	0	ML
ML ≤ T1 ≤ MH	0	0	T1

*2 If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB2) and the MHA of the alarm detection (ALM) are set to 0.
Note that the MHA2 of the alarm detection 2 (ALM2) remains 1.

*3 If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB3) and the MLA of the alarm detection (ALM) are set to 0.
Note that the MLA2 of the alarm detection 2 (ALM2) remains 1.

■Reset windup (4)

If the manipulated value (MV) goes beyond the upper or lower limit, the following operation is performed to return it to the upper or lower limit and enable immediate response when the deviation is inverted. However, when the integral constant (I) is 0, reset windup processing is not performed.

Condition	Operational expression
$T1>MH,\ \frac{\triangle T}{I}<=1$	$MVP=(\frac{\triangle T}{I})(MH-T)+T$
$T1<MH,\ \frac{\triangle T}{I}<=1$	$MVP=(\frac{\triangle T}{I})(ML-T)+T$

■Output ON time conversion processing (5)

The following processing is performed by output ON time conversion processing.

Condition	Processing details
The control output cycle (CTDUTY) has been reached.	The output ON counter is calculated using the following expression. The output counter is cleared to 0 at this time. $\frac{CTDUTY}{\triangle T}\times MV\times\frac{1}{100}$ The output ON counter is rounded off to the nearest whole number.
The control output cycle (CTDUTY) has not been reached.	The output counter is incremented by, 1 and "output conversion processing (6)" is performed.

■Output conversion processing (6)

The following processing is performed by output conversion processing.

Condition	BW1
Output counter < Output ON counter	1
Output counter ≥ Output ON counter	0

■Loop stop processing (7)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.DUTY instruction ends. • The output bit (BW1) is output at the last ON/OFF rate. • The DMLA, MHA, and DLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The control mode (MODE) is set to MAN. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0.
0	The loop runs and "mode determination (1)" is performed.

■Hold processing (8)

This processing specifies whether to hold the output value by the S.DUTY instruction when a sensor error occurs (detected by the S.IN instruction). The hold processing is performed when the value is determined as RUN by "Loop Stop Determination". SM817 is used to specify whether to hold the manipulated value (MV) when a sensor error occurs.

- SM817 = OFF: Do not hold the manipulated value (MV).
- SM817 = ON: Hold the manipulated value (MV).

Operation error

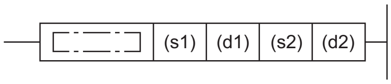
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Control output cycle (CTDUTY) < 0
	The execution cycle (ΔT) setting is less than 0.
	(Control output cycle (CTDUTY)÷execution cycle (ΔT)) > 32767

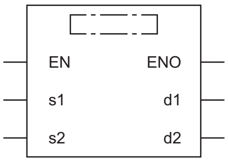
Batch counter

S.BC

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction compares the input data with the set value, and outputs bit data when it reaches the set value.

Ladder	ST
	ENO:=S_BC(EN,s1,s2,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
S.BC	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	32-bit unsigned binary
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

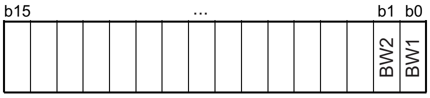
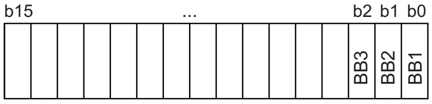
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	0 to 2147483647	32-bit unsigned binary	—	User

■Block memory

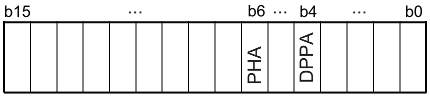
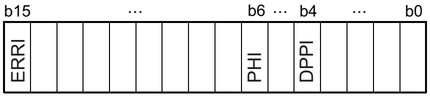
Operand: (d1)

Position	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0	BW	BW1	Output 1	 0: Off 1: ON	16-bit unsigned binary	—	System
		BW2	Output 2				
+1	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Upper limit alarm				
		BB3	Variation rate alarm				

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+3	ALM	Alarm detection	0 to FFFFH  PHA, DPPA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV1	Set value 1	0 to 2147483647	32-bit unsigned binary	0	User
+16 +17	SV2	Set value 2	0 to 2147483647	32-bit unsigned binary	0	User
+26 +27	PH	Upper limit alarm value	0 to 2147483647	32-bit unsigned binary	0	User
+42 +43	CTIM	Variation rate alarm check time	0 to 999999 [s] Set a value within the following range. $\frac{\text{CTIM}}{\triangle \text{T}}$<=32767	Single-precision real number	0.0	User
+44 +45	DPL	Variation rate alarm value	0 to 2147483647	32-bit unsigned binary	0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+124	—	Variation rate monitoring counter initialization completion flag	—	—	—	System
+125		Variation rate monitoring counter (rounded off to the nearest whole number)				
+126 +127	X_{n-m}	—				

■Execution cycle (∆T)

Set the execution cycle (∆T) in SD816 and SD817 in single-precision real number.

Processing details

This instruction compares the input value (E1) with SV1/SV2 and outputs bit data when E1 reaches SV1/SV2.

The instruction also performs upper limit check, variation rate check, and output conversion processing of the input value (E1) at that time.

■Upper limit check (1)

The upper limit check performs the following operations, and outputs the result to the upper limit alarm (BB2) and PHA of the alarm detection (ALM).

Condition	BB2, PHA
E1 > PH	1[*1]
Others	0

*1 If PHI or ERRI of the disable alarm detection (INH) is set to 1, the upper limit alarm (BB2) and the PHA of the alarm detection (ALM) are set to 0.

■Variation rate check processing (2)

A variation rate alarm check is performed during the variation rate alarm check time (CTIM) in the device specified by (d2).

For the variation rate alarm check, the variation of the input value (E1) is compared with the variation rate alarm value (DPL) every execution cycle (∆T).

Condition	BB3, DPPA
$(X_n - X_{n-m}) \geq DPL$	1[*1]
Others	0

*1 If the DPPI or ERRI of the disable alarm detection (INH) is set to 1, the variation rate alarm (BB3) and the DPPA of the alarm detection (ALM) are set to 0.

The variation rate alarm counter is calculated using the following expression.

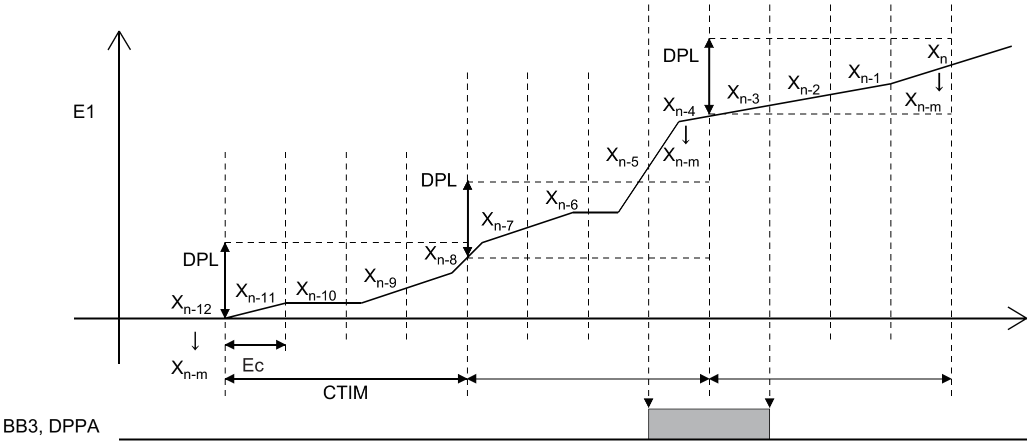
$$m=\frac{CTIM}{\triangle T}$$

Set CTIM and ΔT so that the variation rate alarm counter (m) ≥ 2.

No processing is performed when variation rate alarm counter (m) = 0.

Ex.

When variation rate alarm counter (m) = 4, operations are performed as shown below.



Ec: Execution cycle

■Output conversion processing (3)

In output conversion processing, the following operations are performed and the result is stored in output 1 (BW1)/output 2 (BW2).

Condition	BW1	BW2
E1 < 0	0	0
0≤E1 < SV1	0	—
E1 ≥ SV1	1	—
0≤E1 < SV2	—	0
E1 ≥ SV2	—	1

Operation error

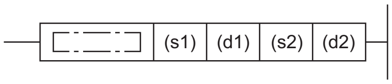
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Variation rate alarm check time (CTIM) < 0
	The execution cycle (ΔT) setting is less than 0.
	(Variation rate alarm check time (CTIM)÷execution cycle (ΔT)) > 32767

Pulse integration

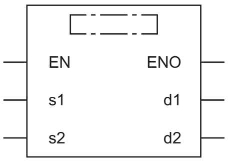
S.PSUM

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction integrates and outputs the number of input pulses.

Ladder	ST
	ENO:=S_PSUM(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PSUM	

Setting data

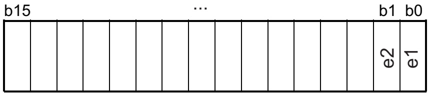
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)							
Position	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1		Input value	Use a ring counter consisting of 16 bits or more. • 16-bit ring counter 00000000H → 0000FFFFH → 00000000H[pulse] • 24-bit ring counter 00000000H → 00FFFFFFH → 00000000H[pulse] • 32-bit ring counter 00000000H → FFFFFFFFH → 00000000H[pulse] However, set 32767 (7FFFH) or less for the pulse increment in each instruction execution.	32-bit unsigned binary	—	User
+2	e	e1	Integration start signal	 Integration start signal 0: Integration stop/reset 1. Integration start Integration hold signal 0: Integration hold clear 1: Integration hold	16-bit unsigned binary	—	User
		e2	Integration hold signal				

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW1	Output value (integral part)	0 to 2147483647	32-bit unsigned binary	—	System
+2 +3	BW2	Output value (decimal part)	0 to 2147483647			

■Operation constant

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	W	Weight per pulse	1 to 999	16-bit unsigned binary	1	User
+1	U	Unit conversion constant	1, 10, 100, 1000	16-bit unsigned binary	1	User
+2 +3	HILMT	Integration upper limit	1 to 2147483647	32-bit unsigned binary	2147483647	User
+4	SUMPTN	Integration pattern	• 0: Return to 0 when the integration upper limit (HILMIT) is exceeded. • 1: Holds the integration upper limit (HILIMT) when it is exceeded.	16-bit unsigned binary	0	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+10 +11	SUM1	Integrated value (integral part)	0 to 2147483647	32-bit unsigned binary	0	System
+12 +13	SUM2	Integrated value (decimal part)	0 to 2147483647	32-bit unsigned binary	0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+116 +117	$E1_{n-1}$	Last input value	—	—	—	System

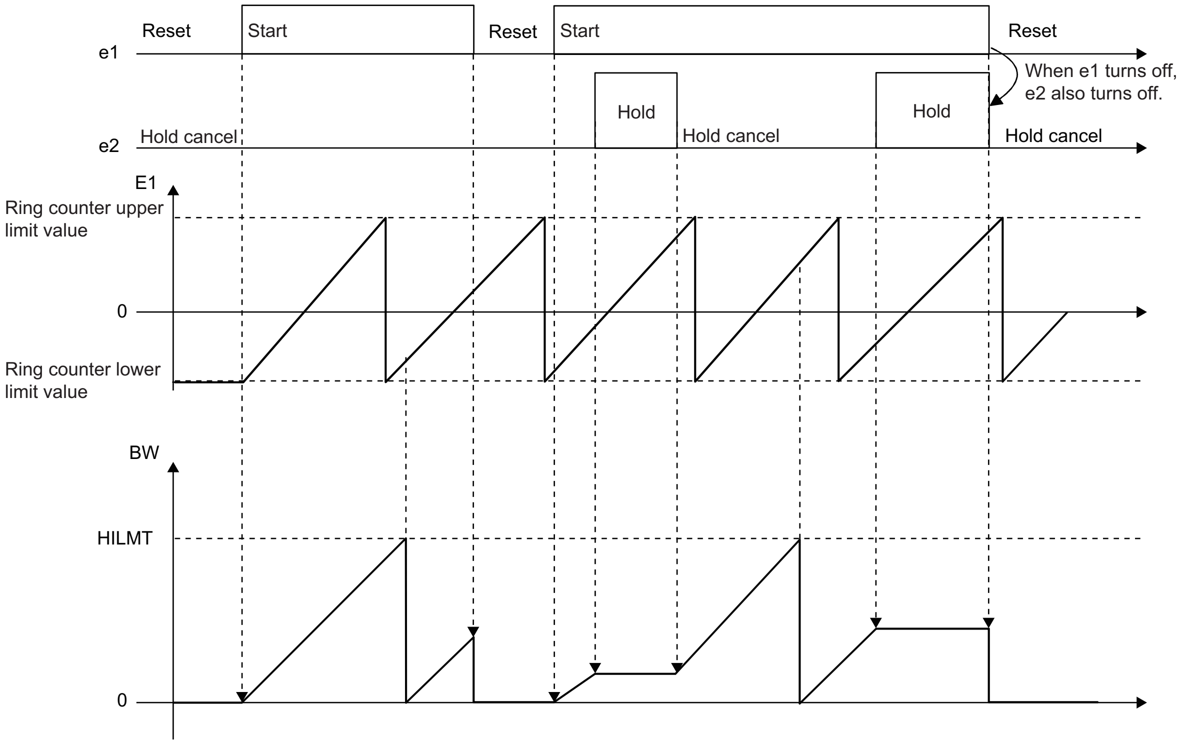
Processing details

This instruction integrates the input value (E1) in the device specified by (s1), and stores the result in the device specified by (d1).

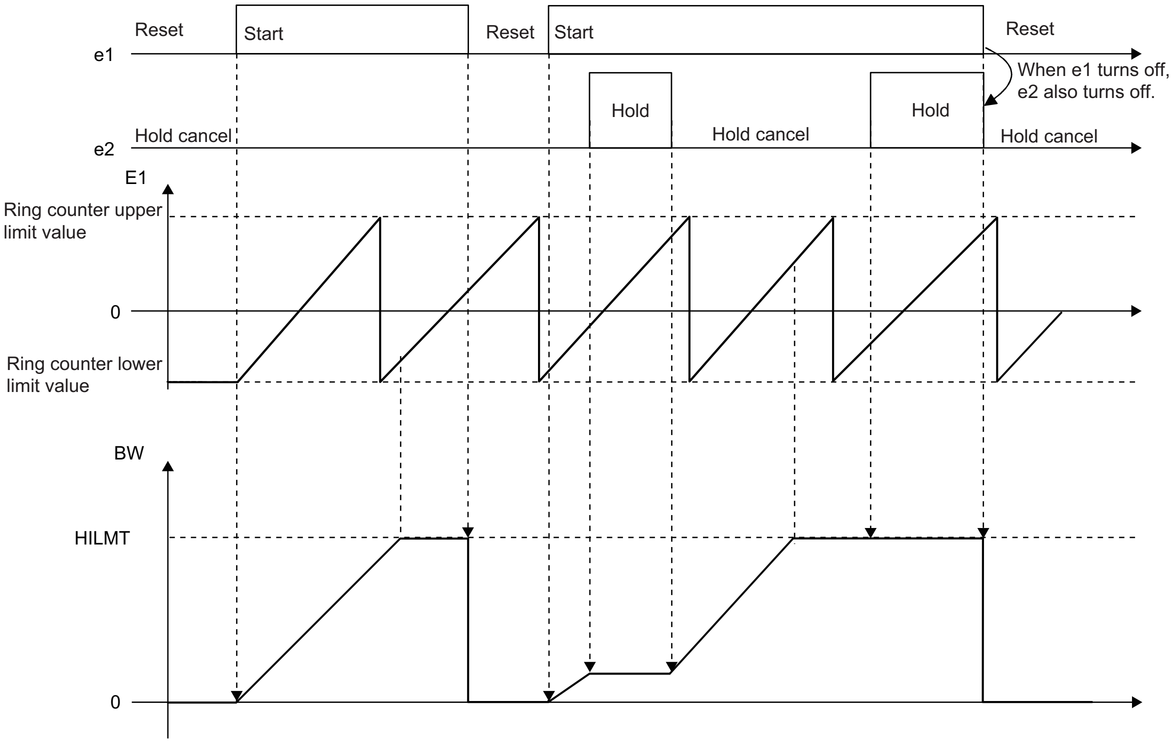
HILMT and SUMPTN can be used to specify whether to return SUM1/SUM2 to 0 or hold the HILMT value when BW1/BW2 exceeds HILMT.

e1 and e2 can be used to start or stop integration of E1.

- Operation performed when the integration pattern is set to "Return to 0 when HILMT is exceeded"



- Operation performed when the integration pattern is set to "Hold the HILMT value when HILMT is exceeded"



■Input value increment operation processing

In input value increment operation processing, the following processing is performed for the input value (E1).

e1	e2	Input value increment (T1)
0	0	—
0	1	—
1	0	$E1-E1_{n-1}$
1	1	—

■Integration value calculation processing

In integrated value calculation processing, the following processing is performed for the input value increment (T1).

e1	e2	Integrated value (decimal part) (T2), integrated value (decimal part) (T3)
0	0	T2 = 0 T3 = 0
0	1	T2 = 0 T3 = 0 [*1]
1	0	T4 = {(T1×W)÷U} quotient <integral part> T5 = {(T1×W)÷U} remainder <decimal part> T2 = SUM1 + T4 + [{(SUM2 + T5)÷U} quotient] <integral part> T3 = {(SUM2 + T5)÷U} remainder <decimal part>
1	1	T2 = SUM1 T3 = SUM2

[*1] In the case of integration stop/reset (e1 = 0), processing is performed by assuming it as integration hold clear (e2 = 0).

■Output conversion processing

In output conversion processing, the following processing is performed for the integrated values (T2, T3).

SUMPTN	Condition	BW1, SUM1	BW2, SUM2
0	T2 ≥ HILMT	BW1 = T2÷remainder of HILMT SUM1 = T2÷remainder of HILMT	BW2 = T3 SUM2 = T3
	Others	BW1 = T2 SUM1 = T2	BW2 = T3 SUM2 = T3
1	T2 ≥ HILMT	BW1 = HILMT SUM1 = HILMT	BW2 = 0 SUM2 = 0
	Others	BW1 = T2 SUM1 = T2	BW2 = T3 SUM2 = T3

Operation error

Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.

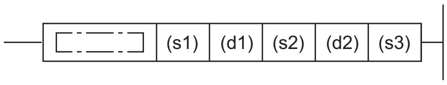
11.3 Control Operation Instructions

Basic PID control

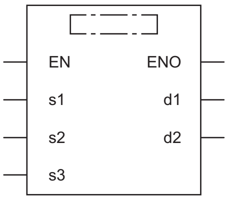
S.PID

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction performs process value differential type (inexact differential) PID operation. The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, PID operation, and deviation check.

Ladder	ST
	ENO:=S_PID(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PID	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

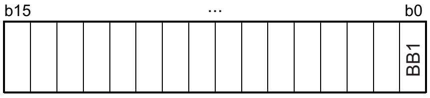
■Input data

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

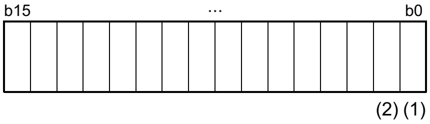
■Block memory

Operand: (d1)

<table>
<tr><th>Position</th><th colspan="2">Symbol</th><th>Name</th><th>Recommended range</th><th>Data type</th><th>Standard value</th><th>Set by</th></tr>
<tr><td>+0
+1</td><td colspan="2">BW</td><td>Output value (∆MV)</td><td>-999999 to 999999</td><td>Single-precision real number</td><td>—</td><td>System</td></tr>
<tr><td>+2</td><td>BB</td><td>BB1</td><td>Large deviation alarm</td><td>
0: No alarm
1: Alarm</td><td>16-bit unsigned binary</td><td>—</td><td>System</td></tr>
</table>

■Operation constant

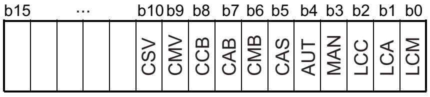
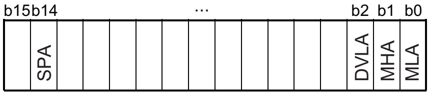
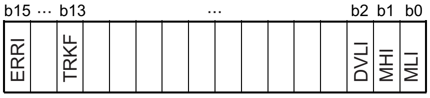
Operand: (s2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	MTD	Derivative gain	0 to 999999	Single-precision real number	8.0	User
+2 +3	DVLS	Large deviation alarm Hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+4	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+5	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+6	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

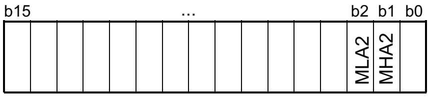
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH  	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP DVLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  TRKF 0: Tracking not set 1: Tracking set ERRI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T}$<=32767	Single-precision real number	1.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	D	Derivative constant	0 to 999999 [s]	Single-precision real number	0.0	User
+58 +59	GW	Gap width	0 to 100[%]	Single-precision real number	0.0	User
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+102 +103	B_{n-1}	Last value				
+104 +105	PV_n	Process value				
+106 +107	PV_{n-1}	Last process value				
+108 +109	PV_{n-2}	Last-but-one process value				
+110 +111	DV_{n-1}	Last deviation value				
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

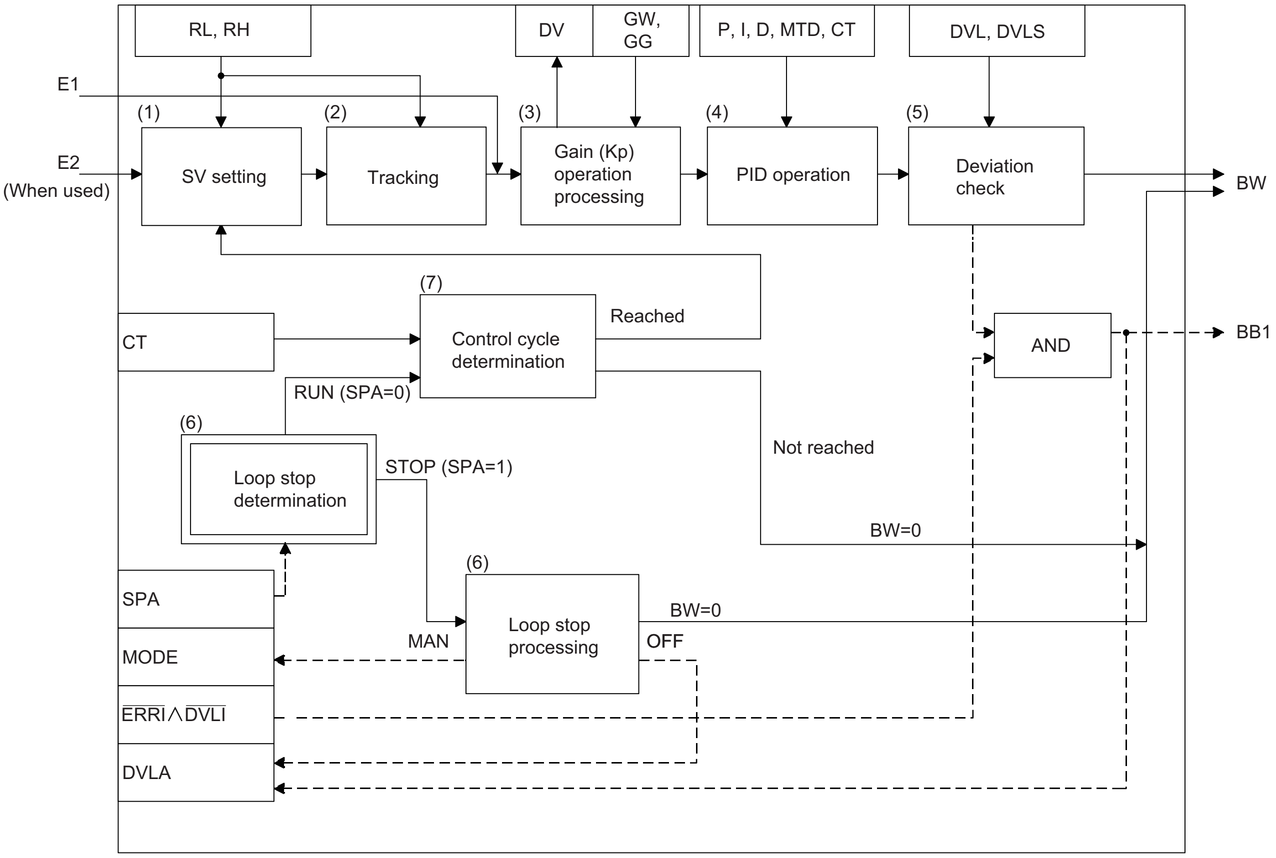
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs PID operation when the specified control cycle is reached. (The PID operation is of the velocity type/process value differential type (inexact differential).)

At this time, the instruction also performs the following processing steps: SV setting, tracking, gain (Kp) operation, and deviation check.

The following is the processing block diagram of the S.PID instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n'=\frac{100}{RH-RL}\times(SV_n-RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

  $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\lvert DV\rvert \leq GW$	$K = GG$
$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

■PID operation (4)

The PID operation is performed with the following operational expression.

Item		Operational expression
B_n	Direct action (PN = 1)	$B_{n-1}+\frac{M_D\times T_D}{M_D\times CT+T_D}\times\{(PV_n-2PV_{n-1}+PV_{n-2})-\frac{CT\times B_{n-1}}{T_D}\}$
	Reserve action (PN = 0)	$B_{n-1}+\frac{M_D\times T_D}{M_D\times CT+T_D}\times\{-(PV_n-2PV_{n-1}+PV_{n-2})-\frac{CT\times B_{n-1}}{T_D}\}$
BW(ΔMV)		$K_P\times\{(DV_n-DV_{n-1})+\frac{CT}{T_I}\times DV_n+B_n\}$

K_P: K × Gain (P), M_D: Derivative gain (MTD), T_I: Integral constant (I), T_D: Derivative constant (D)

Note that special processing is performed in the following cases.

Condition	Processing
In either of the following cases: 1. Derivative constant (D) = 0 ($T_D = 0$) 2. Control mode (MODE) = MAN, LCM, or CMV	$B_n = 0$ (Note that the loop tag past value memory is set.)
In any of the following cases: 1. Integral constant (I) = 0 ($T_I = 0$) 2. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP>MH and $\frac{CT}{T_I}\times DV_n>0$ 3. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP<ML and $\frac{CT}{T_I}\times DV_n<0$	$\frac{CT}{T_I}\times DV_n=0$

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1).

Condition	Result
$DVL < \lvert DV\rvert$	DVLA = BB1 = 1[*1]
$(DVL - DVLS) < \lvert DV\rvert \leq DVL$	DVLA = BB1 = Last value status hold[*1]
$\lvert DV\rvert \leq (DVL - DVLS)$	DVLA = BB1 = 0

[*1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) are set to 0.

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.PID instruction ends. • The output value (BW (ΔMV)) is set to 0. • The DVLA of alarm detection (ALM) is set to 0. • The control mode (MODE) is set to MAN. • BB1 of BB is set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

If the specified control cycle is not reached, BW (ΔMV) is set to 0 and the S.PID instruction is terminated.

If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

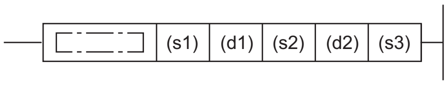
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Two-degree-of-freedom PID control

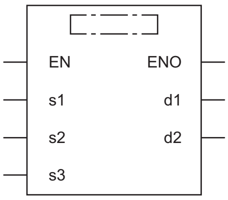
S.2PID

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

This instruction performs two-degree-of-freedom PID control operation (inexact differential). The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, two-degree-of-freedom PID control operation, and deviation check.

Ladder	ST
	ENO:=S_2PID(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.2PID	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

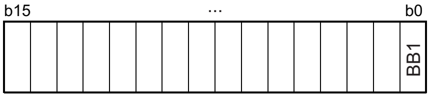
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

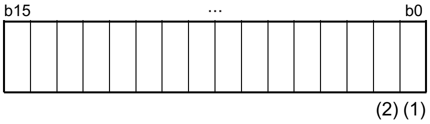
■Input data

Operand: (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value (ΔMV)	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB BB1	Large deviation alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	MTD	Derivative gain	0 to 999999	Single-precision real number	8.0	User
+2 +3	DVLS	Large deviation alarm hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+4	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+5	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+6	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b2 b1 b0 SPA DVLA MHA MLA SPA 0: Loop RUN 1: Loop STOP DVLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b13 … b2 b1 b0 ERRI TRKF DVLI MHI MLI TRKF 0: Tracking not set 1: Tracking set ERRI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T} <= 32767$	Single-precision real number	1.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	D	Derivative constant	0 to 999999 [s]	Single-precision real number	0.0	User
+58 +59	GW	Gap width	0 to 100 [%]	Single-precision real number	0.0	User
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

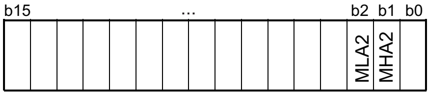
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+64 +65	α	Two-degree-offreedom parameter α	0 to 1 Increasing α decreases the manipulated value variation relative to the set value change. (It will take time to stabilize.) Decreasing α increases the manipulated value variation relative to the set value change. However, it strengthens the compensation operation and accordingly makes hunting greater.	Single-precision real number	0.0	User
+66 +67	β	Two-degree-offreedom parameter β	0 to 1 Increasing β decreases the effect of derivative control on the set value change. Decreasing β increases the effect of derivative control on the set value change.	Single-precision real number	1.0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+102 +103	B_{n-1}	Last value				
+104 +105	PV_n	Process value				
+106 +107	PV_{n-1}	Last process value				
+108 +109	PV_{n-2}	Last-but-one process value				
+110 +111	DV_{n-1}	Last deviation value				
+112 +113	DV_{n-2}	Last but one deviation value				
+114 +115	D_{n-1}	Last value				
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

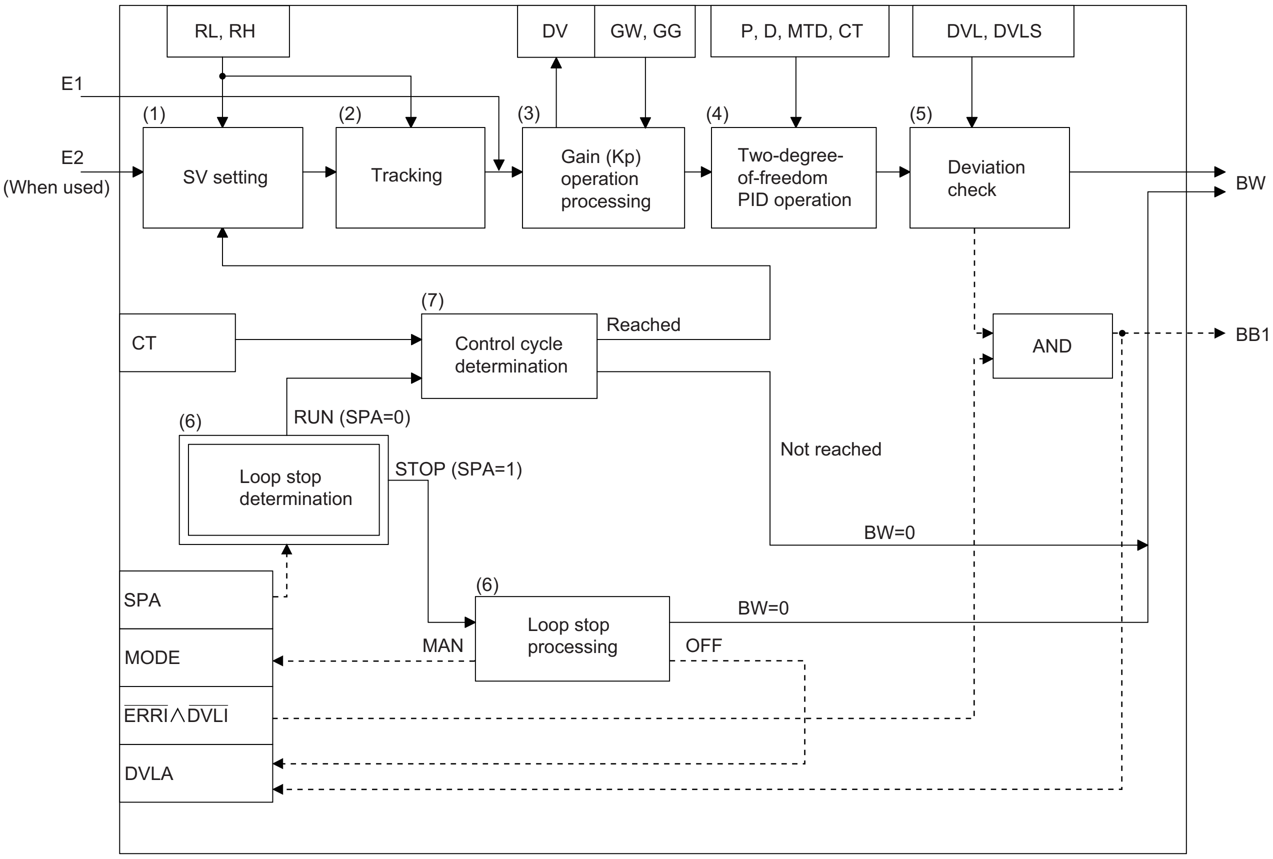
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs two-degree-of-freedom PID control operation when the specified control cycle is reached.

At this time, the instruction also performs the following processing steps: SV setting, tracking, gain (Kp) operation, and deviation check.

The following is the processing block diagram of the S.2PID instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n' = \frac{100}{RH-RL} \times (SV_n - RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

    $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\lvert DV \rvert \leq GW$	$K = GG$
$\lvert DV \rvert > GW$	$K = 1 - \frac{(1-GG) \times GW}{\lvert DV \rvert}$

■Two-degree-of-freedom PID control (4)

The two-degree-of-freedom PID control operation is performed with the following operational expression.

Item		Operational expression
B_n		$B_{n-1} + \frac{M_D \times T_D}{M_D \times CT + T_D} \times \{(DV_n - 2DV_{n-1} + DV_{n-2}) - \frac{CT \times B_{n-1}}{TD}\}$
C_n	Direct action (PN = 1)	$PV_n - PV_{n-1}$
	Reserve action (PN = 0)	$-(PV_n - PV_{n-1})$
D_n	Direct action (PN = 1)	$D_{n-1} + \frac{M_D \times T_D}{M_D \times CT + T_D} \times \{(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{TD}\}$
	Reserve action (PN = 0)	$D_{n-1} + \frac{M_D \times T_D}{M_D \times CT + T_D} \times \{-(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{T_D}\}$
BW(ΔMV)		$K_P \times \{(1-\alpha) \times (DV_n - DV_{n-1}) + \frac{CT}{T_I} \times DV_n + (1-\beta) \times B_n + \alpha \times C_n + \beta \times D_n\}$

K_P: K × Gain (P), M_D: Derivative gain (MTD), T_I: Integral constant (I), T_D: Derivative constant (D)

Note that special processing is performed in the following cases.

Condition	Processing
In either of the following cases: 1. Derivative constant (D) = 0 (T_D = 0) 2. Control mode (MODE) = MAN, LCM, or CMV	B_n=0, D_n=0 (Note that the loop tag past value memory is set.)
In any of the following cases: 1. Integral constant (I) = 0 (T_I = 0) 2. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP>MH and $\frac{CT}{T_I}\times DV_n>0$ 3. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP<ML and $\frac{CT}{T_I}\times DV_n<0$	$\frac{CT}{T_I}\times DV_n=0$

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) in the block memory.

Condition	Result
DVL < |DV|	DVLA = BB1 = 1[*1]
(DVL - DVLS) < |DV| ≤ DVL	DVLA = BB1 = Last value status hold[*1]
|DV| ≤ (DVL - DVLS)	DVLA = BB1 = 0

[*1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) are set to 0.

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.2PID instruction ends. • The output value (BW) is set to 0. • The DVLA of alarm detection (ALM) is set to 0. • The control mode (MODE) is set to MAN. • BB1 of BB is set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

If the specified control cycle is not reached, output value BW (ΔMV) is set to 0 and the S.2PID instruction is terminated.

If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

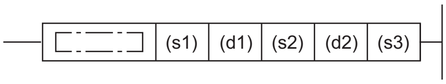
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Position type PID control

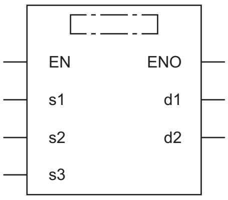
S.PIDP

✕ RnCPU ✕ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

Performs position type PID operation. The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, PID operation, deviation check, and control mode determination. Depending on the operation result up to the mode determination processing, the instruction decides next processing: variation rate & upper/lower limiter and output conversion, or alarm clear and output conversion.

Ladder	ST
 (s1) (d1) (s2) (d2) (s3)	ENO:=S_PIDP(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

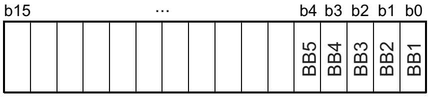
■Input data

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

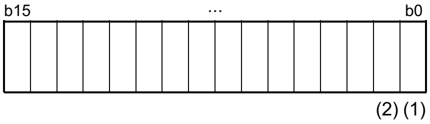
■Block memory

Operand: (d1)

Position	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Large deviation alarm				
		BB3	Output upper limit alarm				
		BB4	Output lower limit alarm				
		BB5	Output variation rate alarm				

■Operation constant

Operand: (s2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	MTD	Derivative gain	0 to 999999	Single-precision real number	8.0	User
+2 +3	DVLS	Large deviation alarm hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+4	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+5	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+6	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User
+7 +8	NMAX	Output conversion upper limit	-999999 to 999999	Single-precision real number	100.0	User
+9 +10	NMIN	Output conversion lower limit	-999999 to 999999	Single-precision real number	0.0	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH b15 ··· b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 ··· b11 ··· b2 b1 b0 SPA DMLA DVLA MHA MLA SPA 0: Loop RUN 1: Loop STOP DVLA, DMLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 ··· b13 ··· b11 ··· b2 b1 b0 ERRI TRKF DMLI DVLI MHI MLI TRKF 0: Tracking not set 1: Tracking set ERRI, DMLI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T}$<=32767	Single-precision real number	1.0	User
+48 +49	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	100.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	D	Derivative constant	0 to 999999 [s]	Single-precision real number	0.0	User
+58 +59	GW	Gap width	0 to 100 [%]	Single-precision real number	0.0	User

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User

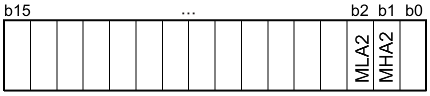
■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+100 +101	I_{n-1}	Last value				
+102 +103	B_{n-1}	Last value				
+104 +105	PV_n	Process value				
+106 +107	PV_{n-1}	Last process value				
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

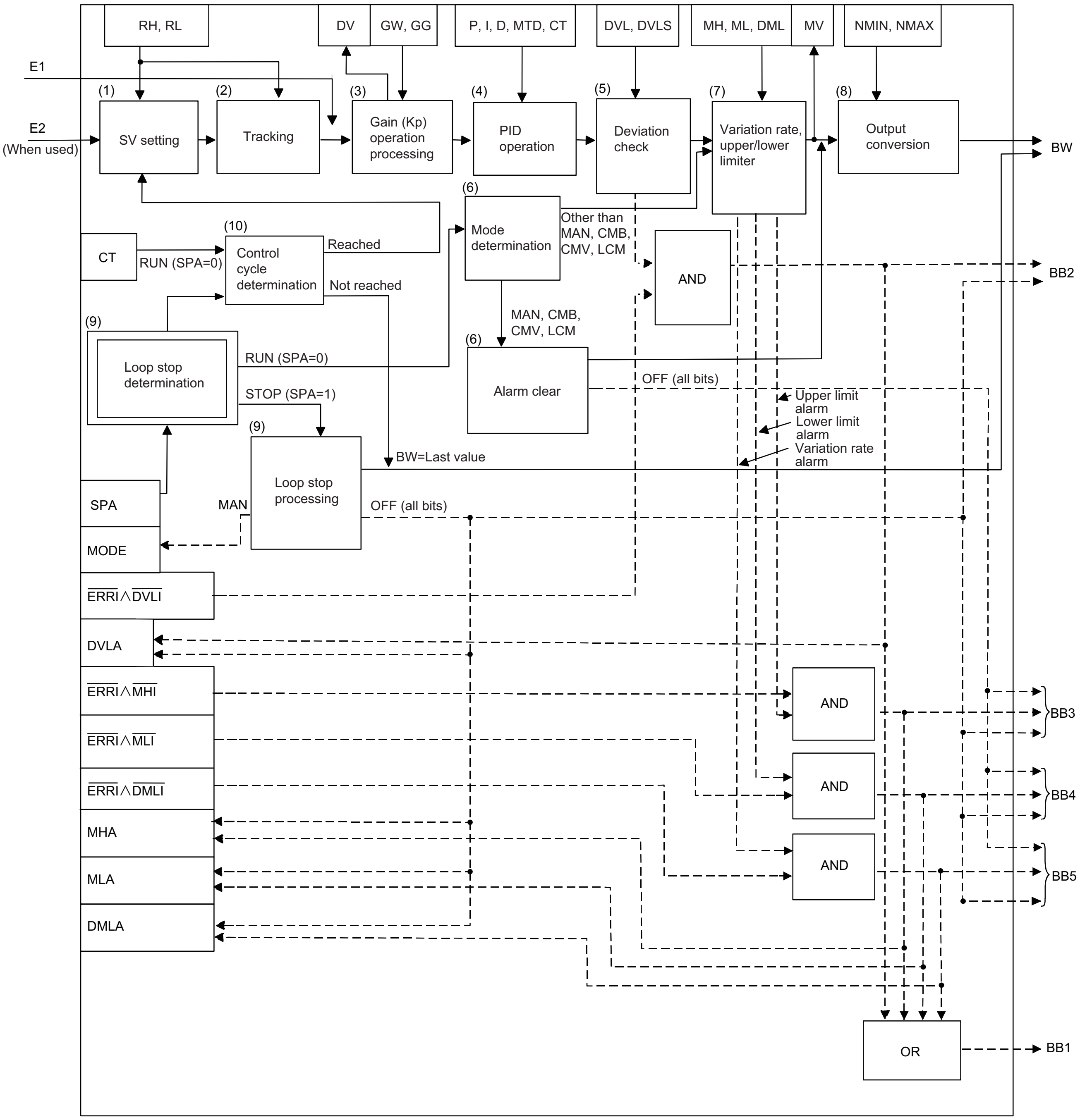
Processing details

This instruction performs position type PID operation when the specified control cycle is reached.

At this time, the instruction also performs the following processing steps: SV setting, tracking, gain (Kp) operation, deviation check, and control mode (MODE) determination.

Depending on the operation result up to the mode determination processing, the instruction performs either variation rate & upper/lower limiter and output conversion processing, or alarm clear and output conversion processing.

The following is the processing block diagram of the S.PIDP instruction. (The numbers (1) to (10) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n'=\frac{100}{RH-RL}\times(SV_n-RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

  $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\lvert DV\rvert \leq GW$	$K = GG$
$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

■PID operation (4)

The PID operation is performed with the following operational expression.

Item		Operational expression
B_n	Direct action (PN = 1)	$B_{n-1}+\frac{M_D×T_D}{M_D×CT+T_D}×\{(PV_n-PV_{n-1})-\frac{CT×B_{n-1}}{T_D}\}$
	Reserve action (PN = 0)	$B_{n-1}+\frac{M_D×T_D}{M_D×CT+T_D}×\{-(PV-PV_{n-1})-\frac{CT×B_{n-1}}{T_D}\}$
I_n		$I_{n-1}+\frac{CT}{T_I}×DV_n$
T		$K_p × (DV_n + I_n + B_n)$

K_P: K × Gain (P), M_D: Derivative gain (MTD), T_I: Integral constant (I), T_D: Derivative constant (D)

Note that special processing is performed in the following cases.

Condition	Processing
In either of the following cases: 1. Derivative constant (D) = 0 (T_D = 0) 2. Control mode (MODE) = MAN, LCM, or CMV	B_n = 0 (Note that the loop tag past value memory is set.)
In any of the following cases: 1. Integral constant (I) = 0 (T_I = 0) 2. MHA2 of the alarm detection 2 (ALM2) = 1 $\frac{CT}{T_I}×DV_n>0$ 3. MLA2 of the alarm detection 2 (ALM2) = 1 $\frac{CT}{T_I}×DV_n<0$ 4. Control mode (MODE) = MAN, LCM, or CMV	$\frac{CT}{T_I}×DV_n=0$
When all of the following conditions are satisfied: 1. b0 of SD818 (bumpless switching function of S.PIDP control) = 1. 2. TRKF of the disable alarm detection (INH) = 1 3. Control mode (MODE) = Other than MAN, LCM, and CMV	$I_{n-1}=\frac{MV}{K_p}-(DV_n+B_n)$ TRKF = 0

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB2).

Condition	Result
DVL < |DV|	DVLA = BB2 = 1[1]
(DVL - DVLS) < |DV| ≤ DVL	DVLA = BB2 = Last value[1]
|DV| ≤ (DVL - DVLS)	DVLA = BB2 = 0

[1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB2) are set to 0.

■Mode determination (6)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM (alarm clear processing)	• The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The alarm bits (BB3, BB4, and BB5) are set to 0. • The data of BB2 is transferred to BB1. (BB1 = BB2) • When b0 of SD818 (bumpless switching function of S.PIDP control) is 1, TRKF of INH is set to 1. • "Output conversion processing (8)" is performed and the instruction ends.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	"Variation rate & upper/lower limiter processing (7)" is performed.

■Variation rate & upper/lower limiter (7)

The variation rate and upper/lower limits of the input value (E1) are checked, and the data after the processing and an alarm are output.

Variation rate limiter processing performs the following operations, and outputs the result to the output variation rate alarm (BB5) of (d1) and the DMLA of the alarm detection (ALM).

Condition	BB5, DMLA	T1
$\|T-MV\| \le DML$	0	T
(T - MV) > DML	1[*1]	MV + DML
(T - MV) < -DML	1[*1]	MV - DML

[*1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the output variation rate alarm (BB5) and the DMLA of the alarm detection (ALM) are set to 0.

Upper/lower limiter processing performs the following operations, and outputs the result to the output upper limit alarm (BB3); output lower limit alarm (BB4); MHA and MLA of the alarm detection (ALM); and MHA2 and MLA2 of the alarm detection (ALM2).

Condition	BB4, MLA, MLA2[*4]	BB3, MHA, MHA2[*4]	MV
T1 > MH	0	1[*2]	MH
T1 < ML	1[*3]	0	ML
$ML \le T1 \le MH$	0	0	T1

[*2] If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB3) and the MHA of the alarm detection (ALM) are set to 0.
Note that the MHA2 of the alarm detection 2 (ALM2) remains 1.

[*3] If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB4) and the MLA of the alarm detection (ALM) are set to 0.
Note that the MLA2 of the alarm detection 2 (ALM2) remains 1.

[*4] If the specified control cycle is not reached, the MHA2 and MLA2 status of the alarm detection 2 (ALM2) are held.

■Output conversion processing (8)

The output value (BW) is calculated from the following expression.

$$BW=\frac{NMAX-NMIN}{100}\times MV+NMIN$$

■Loop stop processing (9)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. The following operations are performed and the instruction ends. • The last output value (BW) is held. • The DVLA, MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The MHA2 and MLA2 of the alarm detection 2 (ALM2) are set to 0. • The control mode (MODE) is set to MAN. • All of the alarm bits (BB1, BB2, BB3, BB4, and BB5) are set to 0.
0	The loop runs and "control cycle determination processing (10)" is performed.

■Control cycle determination (10)

If the specified control cycle is not reached, "mode determination processing (6)" is performed regarding T as MV.

If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

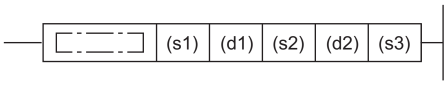
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

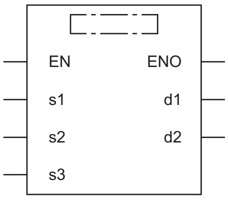
Sample PI control

S.SPI

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

The instruction checks whether ST or HT is applicable and, if ST is applicable, performs the following processing steps: SV setting, tracking, gain (Kp) operation, SPI operation, and deviation check.

Ladder	ST
	ENO:=S_SPI(EN,s1,s2,s3,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
S.SPI	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

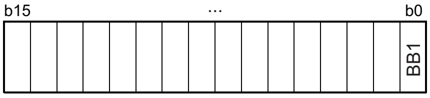
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

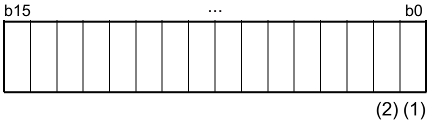
■Input data

Operand: (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value (∆MV)	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB BB1	Large deviation alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System

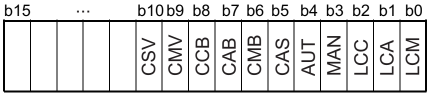
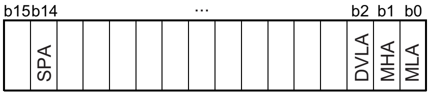
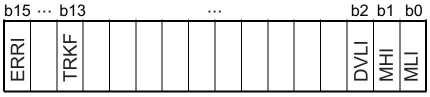
■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	DVLS	Large deviation alarm Hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+2	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+3	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+4	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

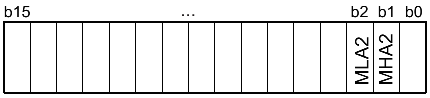
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP DVLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  TRKF 0: Tracking not set 1: Tracking set ERRI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	ST	Operating time	0 to 999999 [s] Set a value within the following range. $\frac{ST}{\triangle T}$<=32767	Single-precision real number	1.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	STHT	Sampling cycle	0 to 999999 [s] Set a value within the following range. $\frac{STHT}{\triangle T}$<=32767	Single-precision real number	0.0	User
+58 +59	GW	Gap width	0 to 100 [%]	Single-precision real number	0.0	User
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Sample counter (The value is rounded off to the nearest whole number.)				
+98		Operating counter (The value is rounded off to the nearest whole number.)				
+99		Hold counter				
+100 +101	DV_{n-1}	Last deviation value				
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

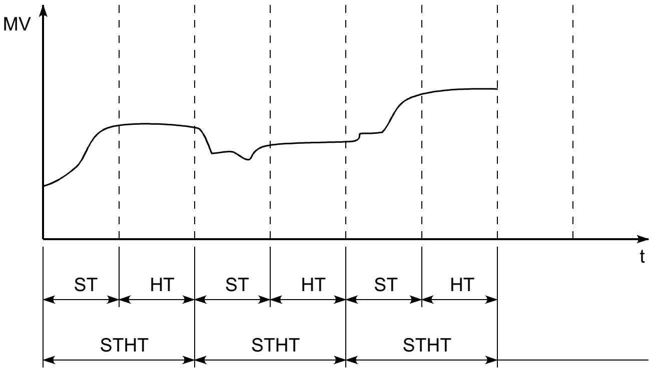
■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

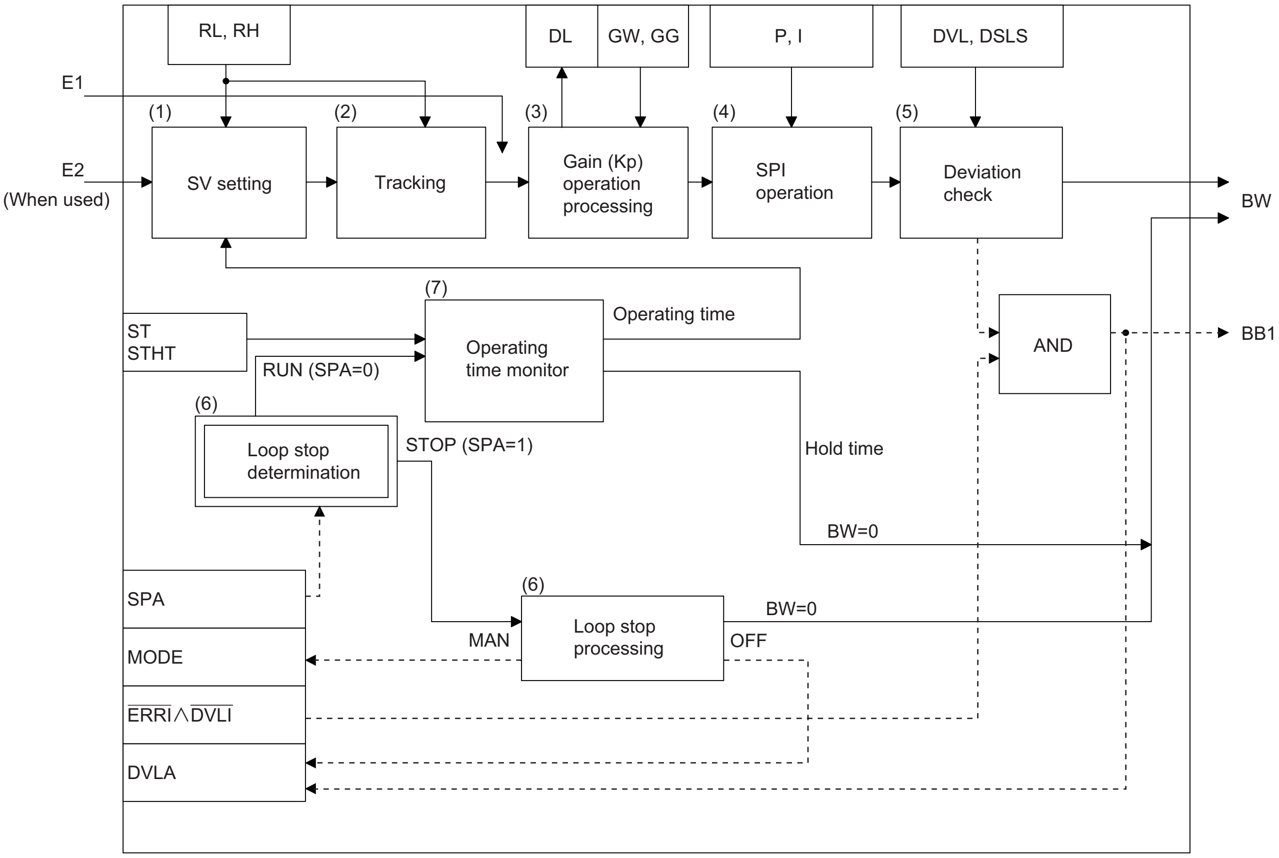
Processing details

This instruction performs normal PI operation during the operating time (ST).

The instruction checks whether ST or HT is applicable and, if ST is applicable, performs the following processing steps: SV setting, tracking, gain (Kp) operation, SPI operation, and deviation check.



The following is the processing block diagram of the S.SPI instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE)	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH\text{-}RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n'=\frac{100}{RH\text{-}RL}\times(SV_n\text{-}RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

    $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\|DV\| \leq GW$	$K = GG$
$\|DV\| > GW$	$K=1-\frac{(1-GG)\times GW}{\|DV\|}$

■SPI operation (4)

The SPI operation is performed with the following operational expression.

Item	Operational expression
During operating time (ST)	$BW=K_P\times\{(DV_n-DV_{n-1})+\frac{BT}{T_I}\times DV_n\}$
During hold time (sample cycle (STHT) - operating time (ST))	BW = 0 (Note that the loop tag past value memory is not set.)

K_P: K×gain (P),T_I: integral constant (I),BT: execution cycle(ΔT)

Note that special processing is performed in the following cases.

Condition	Processing
In any of the following cases: 1. Integral constant (I) = 0 ($T_I = 0$) 2. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP>MH and $\frac{BT}{T_I} \times DV_n > 0$ 3. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP<ML and $\frac{BT}{T_I} \times DV_n < 0$	$\frac{BT}{T_I} \times DV_n = 0$

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1).

Condition	Result
DVL < |DV|	DVLA = BB1 = 1[*1]
(DVL - DVLS) < |DV| ≤ DVL	DVLA = BB1 = Last value status hold[*1]
|DV| ≤ (DVL - DVLS)	DVLA = BB1 = 0

[*1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) are set to 0.

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.SPI instruction ends. • The output value (BW) is set to 0. • The DVLA of alarm detection (ALM) is set to 0. • The control mode (MODE) is set to MAN. • BB1 of BB is set to 0.
0	The loop runs and "operating time/hold time determination (7)" is performed.

■Operating time monitoring (7)

The instruction checks whether ST or HT (= STHT - ST) is applicable and performs the following processing.

Condition	Processing details
Operating time (ST)	The instruction performs the following processing steps: SV setting, tracking, gain (Kp) operation, PI operation (operating time), and deviation check.
Hold time (HT) (= STHT - ST)	The instruction performs the following processing steps: tracking, SPI operation (hold time), and deviation check. Under the following conditions, however, PI control is performed continuously with the hold time set to 0. $\frac{STHT}{\triangle T} <= \frac{ST}{\triangle T}$ If the integral part of the left side of the above expression is 0, no processing is performed. (BW also remains unchanged.)

Operation error

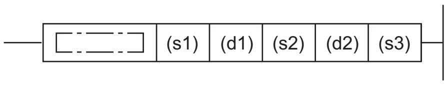
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Operating time (ST) < 0
	The execution cycle (ΔT) setting is less than 0.
	Sampling cycle (STHT) < 0
	(Operating time (ST)÷execution cycle (ΔT)) > 32767
	(Sample cycle (STHT)÷ execution cycle (ΔT)) > 32767

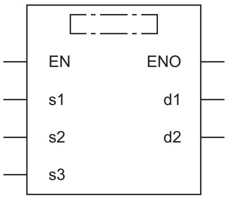
I-PD control

S.IPD

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction performs I-PD operation. The instruction performs the following processing steps: SV setting, tracking, gain K_P operation, IPD operation, and deviation check.

Ladder	ST
	ENO:=S_IPD(EN,s1,s2,s3,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
S.IPD	⎍

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy data	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

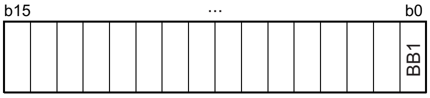
■Input data

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

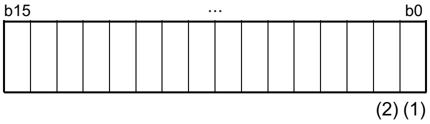
■Block memory

Operand: (d1)

Position	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value (∆MV)	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Large deviation alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	MTD	Derivative gain	0 to 999999	Single-precision real number	8.0	User
+2 +3	DVLS	Large deviation alarm hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+4	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+5	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+6	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

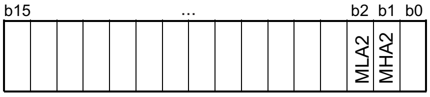
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b2 b1 b0 SPA DVLA MHA MLA SPA 0: Loop RUN 1: Loop STOP DVLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b13 … b2 b1 b0 ERRI TRKF DVLI MHI MLI TRKF 0: Tracking not set 1: Tracking set ERRI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T} <= 32767$	Single-precision real number	1.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	D	Derivative constant	0 to 999999 [s]	Single-precision real number	0.0	User
+58 +59	GW	Gap width	0 to 100 [%]	Single-precision real number	0.0	User
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User
+62 +63	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	0.0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+102 +103	B_{n-1}	Last value				
+104 +105	PV_n	Process value				
+106 +107	PV_{n-1}	Last process value				
+108 +109	PV_{n-2}	Last-but-one process value				
+116	ALM2	Alarm detection 2	 MHA2, MLA2 0: No alarm 1: Alarm			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

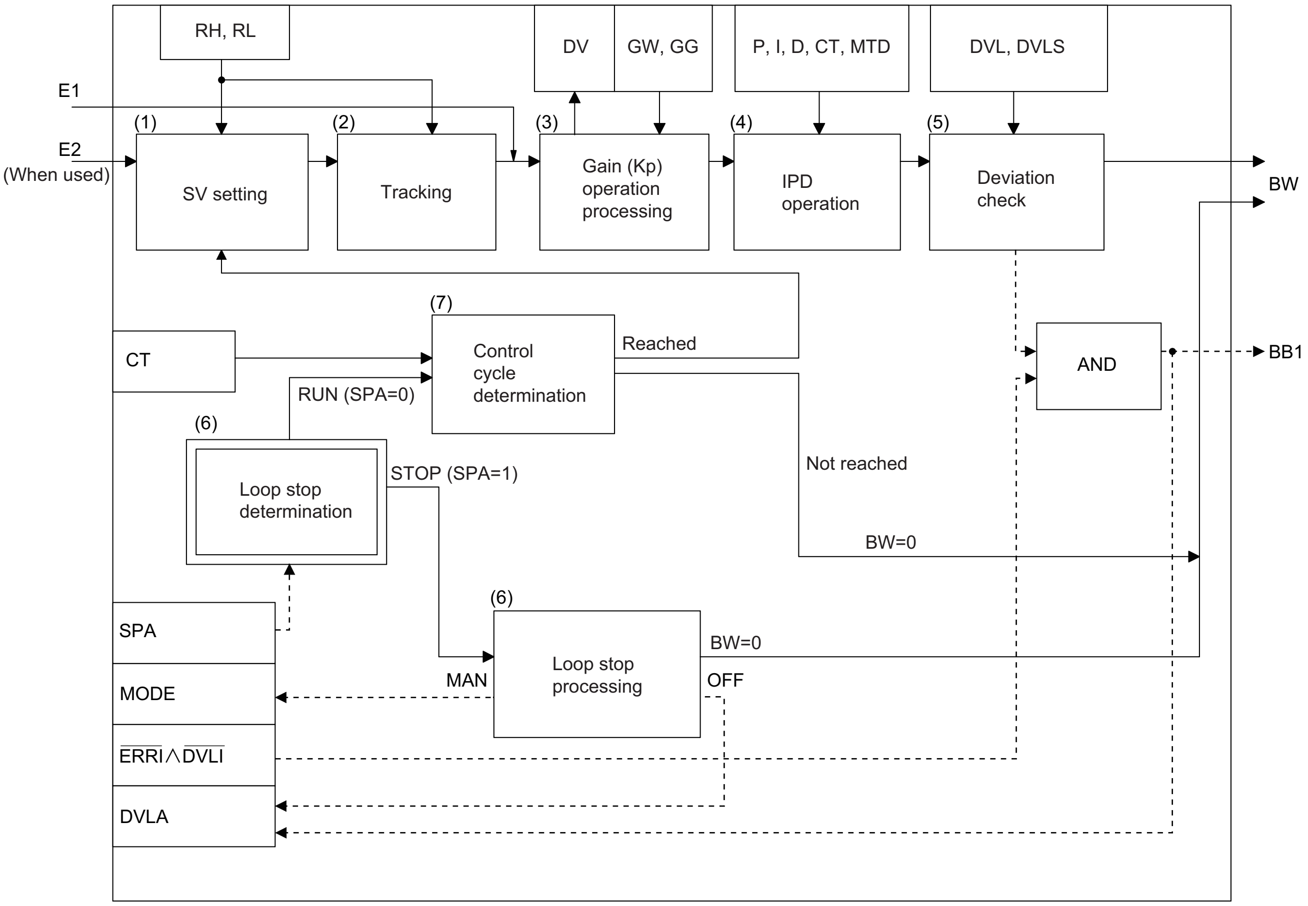
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs I-PD control when the specified control cycle is reached.

At this time, the instruction also performs the following processing steps: SV setting, tracking, gain (Kp) operation, and deviation check.

The following is the processing block diagram of the S.IPD instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n'=\frac{100}{RH-RL}\times(SV_n-RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

    $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\|DV\| \le GW$	$K = GG$
$\|DV\| > GW$	$K=1-\frac{(1-GG)\times GW}{\|DV\|}$

■I-PD operation (4)

The PID operation is performed with the following operational expression.

Item		Operational expression
B_n	Direct action (PN = 1)	$B_{n-1}+\frac{M_D\times T_D}{M_D\times CT+T_D}\times\{(PV_n-2PV_{n-1}+PV_{n-2})-\frac{CT\times B_{n-1}}{T_D}\}$
	Reserve action (PN = 0)	$B_{n-1}+\frac{M_D\times T_D}{M_D\times CT+T_D}\times\{-(PV_n-2PV_{n-1}+PV_{n-2})-\frac{CT\times B_{n-1}}{T_D}\}$
BW(∆MV)	Direct action (PN = 1)	$K_P\times\{\frac{CT}{T_I}\times DV_n+(PV_n-PV_{n-1})+B_n\}$
	Reserve action (PN = 0)	$K_P+\{\frac{CT}{T_I}\times DV_n-(PV_n-PV_{n-1})+B_n\}$

K_P: K × Gain (P), M_D: Derivative gain (MTD), T_I: Integral constant (I), T_D: Derivative constant (D)

Note that special processing is performed in the following cases.

Condition	Processing
In either of the following cases: 1. Derivative constant (D) = 0 ($T_D = 0$) 2. Control mode (MODE) = MAN, LCM, or CMV	$B_n = 0$ (Note that the loop tag past value memory is set.)
In any of the following cases: 1. Integral constant (I) = 0 ($T_I = 0$) 2. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP>MH and $\frac{CT}{T_I}\times DV_n>0$ 3. MHA2 or MLA2 of alarm detection 2 (ALM2) is 1. MVP<ML and $\frac{CT}{T_I}\times DV_n<0$	$\frac{CT}{T_I}\times DV_n=0$

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1).

Condition	Result
$DVL < \|DV\|$	DVLA = BB1 = 1[1]
$(DVL - DVLS) < \|DV\| \le DVL$	DVLA = BB1 = Last value status hold[1]
$\|DV\| \le (DVL - DVLS)$	DVLA = BB1 = 0

*1 If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) are set to 0.

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.IPD instruction ends. • The output value (BW) is set to 0. • The DVLA of alarm detection (ALM) is set to 0. • The control mode (MODE) is set to MAN. • BB1 of BB is set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

If the specified control cycle is not reached, output value (BW) is set to 0 and the S.IPD instruction is terminated.

If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

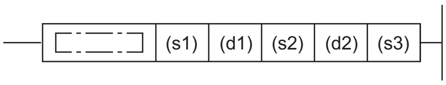
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Blend PI control

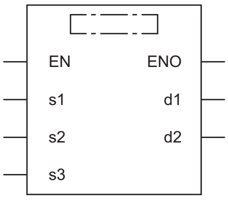
S.BPI

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

This instruction performs blend PI operation. The instruction performs the following processing steps: SV setting, tracking, gain K_P operation, BPI operation, and deviation check.

Ladder	ST
	ENO:=S_BPI(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.BPI	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

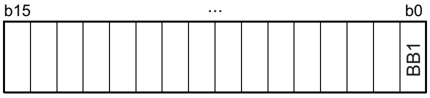
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

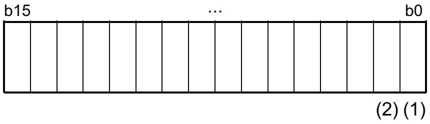
■Input data

Operand: (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value (∆MV)	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB BB1	Large deviation alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	DVLS	Large deviation alarm hysteresis	0 to 100 [%]	Single-precision real number	2.0	User
+2	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+3	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+4	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b2 b1 b0 SPA … DVLA MHA MLA SPA 0: Loop RUN 1: Loop STOP DVLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b13 … b2 b1 b0 ERRI TRKF … DVLI MHI MLI TRKF 0: Tracking not set 1: Tracking set ERRI, DVLI, MHI, MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T}<=32767$	Single-precision real number	1.0	User
+50 +51	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	100.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User
+56 +57	SDV	DV cumulative total (ΣDV)	-999999 to 999999 [%]	Single-precision real number	0.0	System
+58 +59	GW	Gap width	0 to 100 [%]	Single-precision real number	0.0	User
+60 +61	GG	Gap gain	0 to 999999	Single-precision real number	1.0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+98 +99		—	$\frac{CT}{T_I}\times\Sigma DV_I$			

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

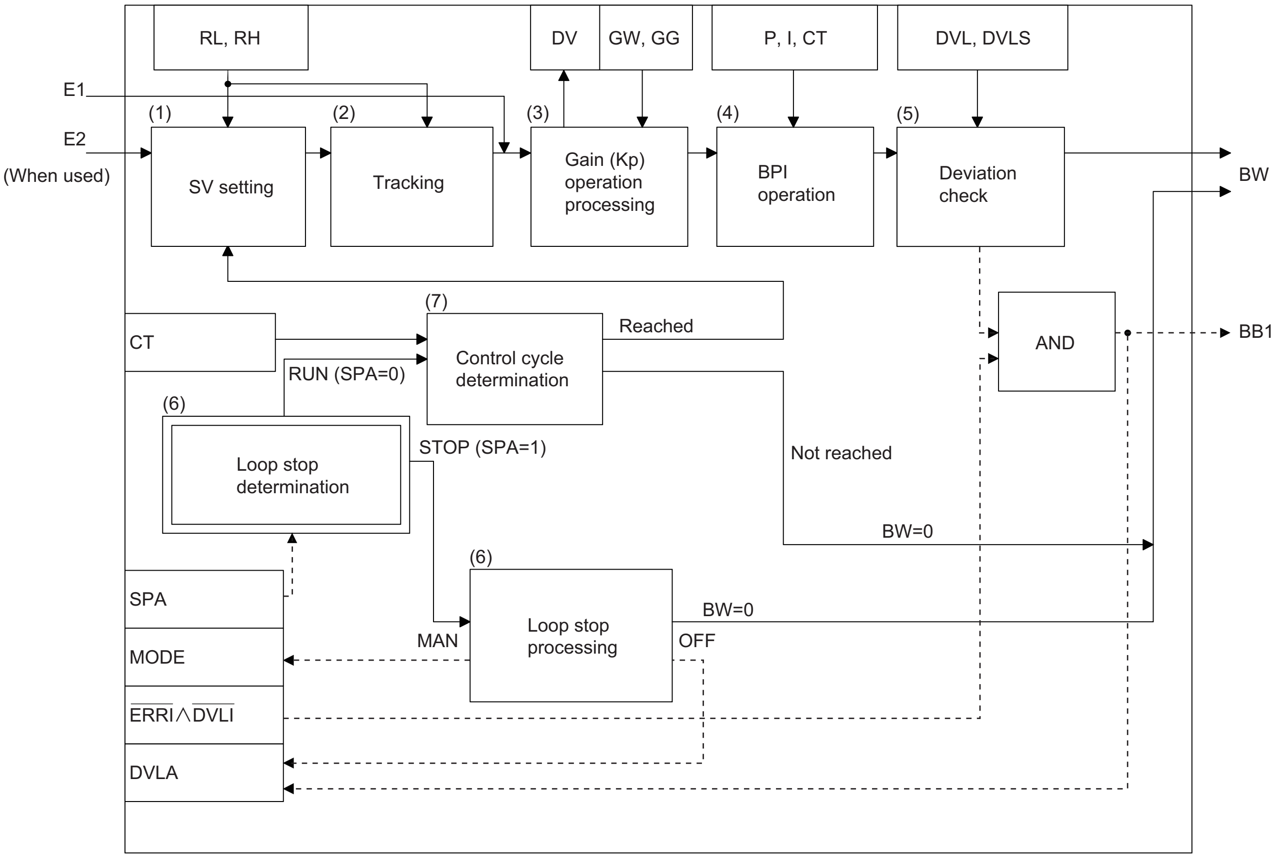
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs BPI operation when the specified control cycle is reached.

At this time, the instruction also performs the following processing steps: SV setting, tracking, gain (Kp) operation, and deviation check.

The following is the processing block diagram of the S.BPI instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ • If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SV_n' is calculated (refer to the following expression).

$$SV_n'=\frac{100}{RH-RL}\times(SV_n-RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

  $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Gain (Kp) operation processing (3)

- The deviation (DV) is calculated under the following conditions.

Condition	Operational expression
Direct action (PN = 1)	$DV = E1 - SV_n'$
Reserve action (PN = 0)	$DV = SV_n' - E1$

- The output gain (K) is calculated under the following conditions.

Condition	Operational expression
$\lvert DV\rvert \le GW$	$K = GG$
$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

■BPI operation (4)

The BPI operation is performed with the following operational expression.

Item	Operational expression
BW(ΔMV)	$K_P\times BT\times(DV_n+\frac{CT}{T_I}\times\Sigma DV_I)$

K_P: K×gain (P), BT: execution cycle, T_I: integral constant (I), ΣDV_I: DV_n cumulative value, DV_n: deviation

Note that special processing is performed in the following cases.

Condition	Processing
In either of the following cases: 1. Integral constant (I) = 0 ($T_I = 0$) 2. MHA or MLA of alarm detection (ALM) is 1.	$\frac{CT}{T_I}\times\Sigma DV_I$= Last value
Integral constant (I) ≠ 0 ($T_I \ne 0$)	$\frac{CT}{T_I}\times\Sigma DV_I=\frac{CT}{T_I}\times(\Sigma DV_I+DV_n)$

■Deviation check (5)

A deviation is checked under the following conditions, and the result is output to the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) in the device specified by (d2).

Condition	Result
$DVL < \lvert DV\rvert$	DVLA = BB1 = 1[1]
$(DVL - DVLS) < \lvert DV\rvert \le DVL$	DVLA = BB1 = Last value status hold[1]
$\lvert DV\rvert \le (DVL - DVLS)$	DVLA = BB1 = 0

[1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the DVLA of the alarm detection (ALM) and the large deviation alarm (BB1) are set to 0.

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.BPI instruction ends. • The output value (BW) is set to 0. • The DVLA of alarm detection (ALM) is set to 0. • The control mode (MODE) is set to MAN. • BB1 of BB is set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

If the specified control cycle is not reached, output value (BW) is set to 0 and the S.BPI instruction is terminated.

If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

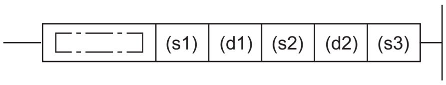
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Ratio calculation

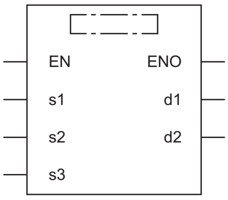
S.R

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction performs the following steps for the input data: engineering value transformation, tracking, variation rate limiter, and ratio calculation.

Ladder	ST
	ENO:=S_R(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.R	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	When E2 is used: Set value start device When E2 is not used: Dummy device	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

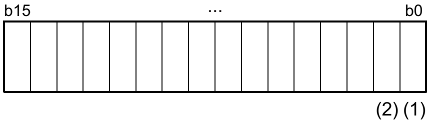
■Input data

Operand: (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999 [%]	Single-precision real number	—	System

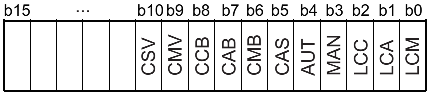
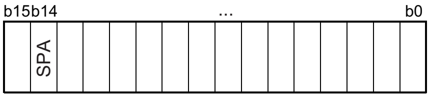
■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+1	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP	16-bit unsigned binary	4000H	User/ system
+14 +15	SPR	Set value	-999999 to 999999	Single-precision real number	0.0	User
+16 +17	BIAS	Bias	-999999 to 999999 [%]	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T} <= 32767$	Single-precision real number	1.0	User
+50 +51	DR	Variation rate limit value	0 to 999999	Single-precision real number	100.0	User
+52 +53	RMAX	Ratio upper limit value	-999999 to 999999	Single-precision real number	100.0	User
+54 +55	RMIN	Ratio lower limit value	-999999 to 999999	Single-precision real number	0.0	User
+56 +57	R_n	Ratio current value	-999999 to 999999	Single-precision real number	0.0	System

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				
+98 +99	R_{n-1}	Last value				

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User

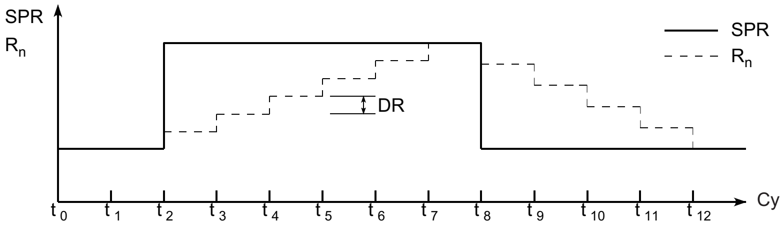
■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

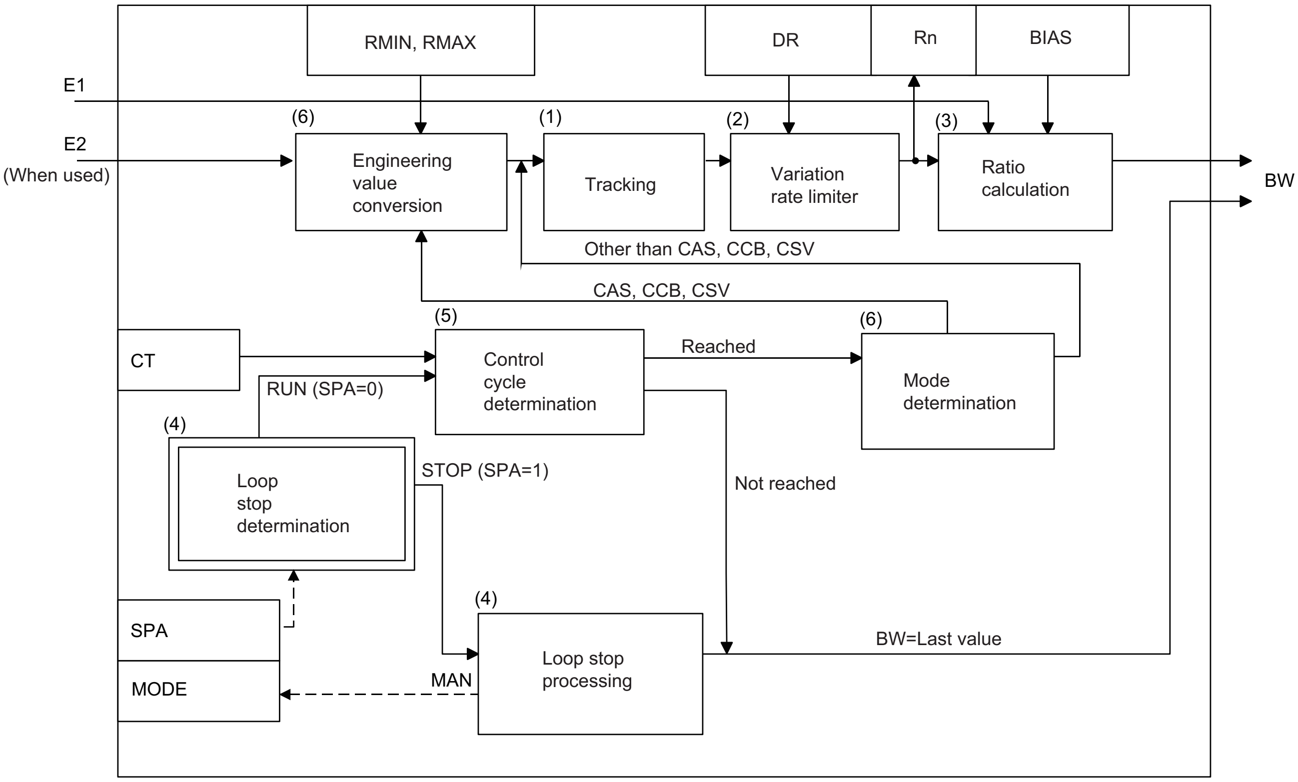
This instruction performs ratio calculation when the specified control cycle is reached.

At this time, the instruction also performs the following steps: control mode (MODE) determination, engineering value transformation, tracking, and variation rate limiter.



SPR: Set value, R_n: Current ratio value, DR: Variation rate limit value, Cy: Control cycle

The following is the processing block diagram of the S.R instruction. (The numbers (1) to (6) in the diagram indicate the order of the processing.)



■Tracking processing (1)

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

$$E2=\frac{100}{RMAX-RMIN}\times(SPR-RMIN)$$

- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■Variation rate limiter (2)

Variation rate limiter processing performs the following operations, and stores the result in the current ratio value (R_n).

Condition	Operational expression
$(SPR-R_n) \geq DR$	$R_n=R_{n-1}+DR$
$(SPR-R_n) \leq -DR$	$R_n=R_{n-1}-DR$
$\lvert SPR-R_n\rvert<DR$	$R_n=SPR$

■Ratio calculation (3)

The ratio calculation is performed with the following operational expression.

$$BW=\frac{R_n-RMIN}{RMAX-RMIN}\times E1+BIAS$$

■Loop stop processing (4)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.R instruction ends. • The last output value (BW) is held. • The control mode (MODE) is set to MAN.
0	The loop runs and "control cycle determination processing (5)" is performed.

■Control cycle determination (5)

If the specified control cycle is not reached, output value (BW) is set to 0 and the S.R instruction is terminated.

If the specified control cycle is reached, "mode determination (6)" is performed.

■Mode determination (6)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
CAS, CCB, CSV	• If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "variation rate limiter (2)" is performed. $SPR=\frac{RMAX-RMIN}{100}\times E2+RMIN$ • If the set value (E2) is not specified, "variation rate limiter (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (1)" is performed.

Operation error

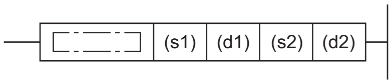
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Upper/lower limit alarm

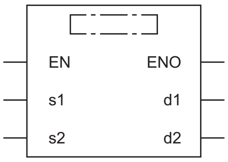
S.PHPL

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

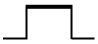
Checks whether the PV that has been output by the S.IN instruction exceeds the upper limit or underruns the lower limit.

Ladder	ST
 (s1) (d1) (s2) (d2)	ENO:=S_PHPL(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PHPL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

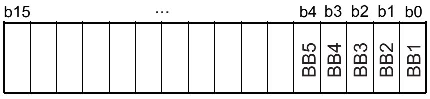
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

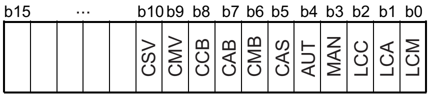
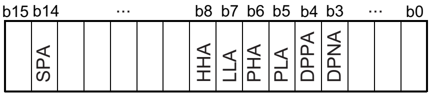
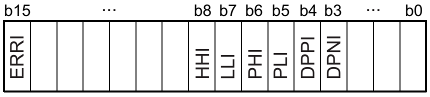
Operand: (s1)						
Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Position	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Upper limit alarm				
		BB3	Lower limit alarm				
		BB4	Positive direction variation rate alarm				
		BB5	Negative direction variation rate alarm				

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH 	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP HHA, LLA, PHA, PLA, DPPA, DPNA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+10 +11	PV	Process value	RL to RH	Single-precision real number	0.0	System
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+26 +27	PH	Upper limit alarm value	RL to RH	Single-precision real number	100.0	User
+28 +29	PL	Lower limit alarm value	RL to RH	Single-precision real number	0.0	User
+30 +31	HH	Upper upper limit alarm value	RL to RH	Single-precision real number	100.0	User
+32 +33	LL	Lower lower limit alarm value	RL to RH	Single-precision real number	0.0	User

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+40 +41	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User
+42 +43	CTIM	Variation rate alarm check time	0 to 999999 [s] Set a value within the following range. $\frac{\text{CTIM}}{\triangle\text{T}}$<=32767	Single-precision real number	0.0	User
+44 +45	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	100.0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

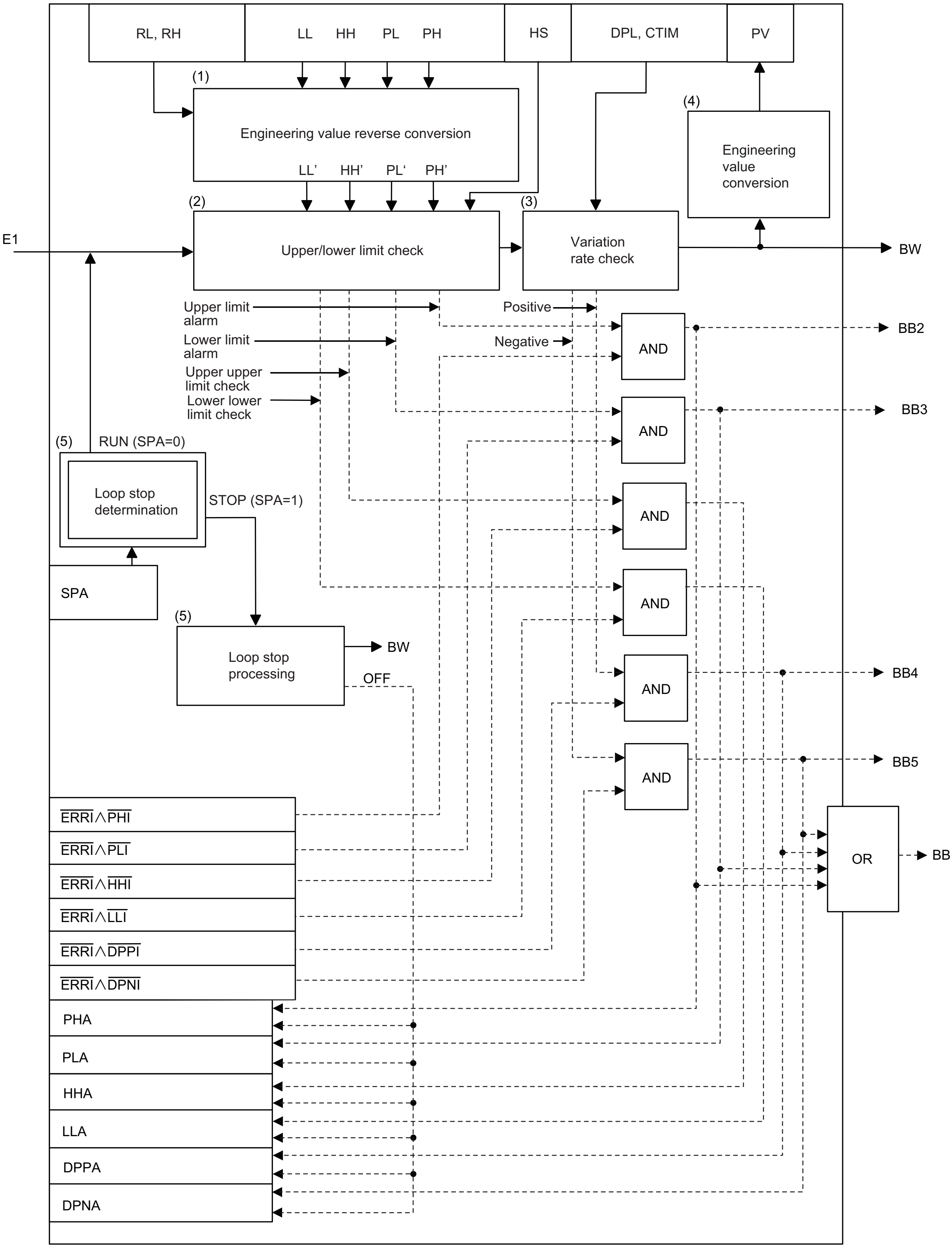
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+124	—	Variation rate monitoring counter initialization completion flag	—	—	—	System
+125		Variation rate monitoring counter (rounded off to the nearest whole number)				
+126 +127	$\text{E1}_{\text{n-m}}$	—				

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction checks whether the input value (E1) exceeds the upper limit or underruns the lower limit to output an alarm. The following is the processing block diagram of the S.PHPL instruction. (The numbers (1) to (5) in the diagram indicate the order of the processing.)



■Engineering value inverse transformation (1)

The instruction performs the following operations to match the ranges of PH, PL, HH, and LL to the input values (E1).

$$PH'=\frac{100}{RH-RL}\times(PH-RL)$$

$$PL'=\frac{100}{RH-RL}\times(PL-RL)$$

$$HH'=\frac{100}{RH-RL}\times(HH-RL)$$

$$LL'=\frac{100}{RH-RL}\times(LL-RL)$$

■Upper/lower limit check (2)

The upper and lower limits of the input value (E1) are checked under the following conditions.

Check item	Condition	ALM	BB2	BB3
Upper limit check	E1 > PH'	PHA = 1[*1]	1[*1]	—
	E1≤PH' - HS	PHA = 0	0	—
	Others	PHA: The last value is held.[*1]	Hold[*1]	—
Lower limit check	E1 < PL'	PLA = 1[*2]	—	1[*2]
	E1≥PL' + HS	PLA = 0	—	0
	Others	PLA: The last value is held.[*2]	—	Hold[*2]
Upper upper limit check	E1 > HH'	HHA = 1[*3]	—	—
	E1≤HH' - HS	HHA = 0	—	—
	Others	HHA: The last value is held.[*3]	—	—
Lower lower limit check	E1 < LL'	LLA = 1[*4]	—	—
	E1≥LL'+HS	LLA = 0	—	—
	Others	LLA: The last value is held.[*4]	—	—

[*1] If ERRI or PHI of disable alarm detection (INH) is set to 1, PHA of ALM and BB2 are set to 0.
[*2] If ERRI or PLI of disable alarm detection (INH) is set to 1, the PLA of ALM and BB3 are set to 0.
[*3] If ERRI or HHI of disable alarm detection (INH) is set to 1, HHA of ALM is set to 0.
[*4] If ERRI or LLI of disable alarm detection (INH) is set to 1, LLA of ALM is set to 0.

■Variation rate check (3)

- A variation rate check is performed for the duration specified by the variation rate alarm check time (CTIM). The number of executions of a variation rate check is determined by the following expression.

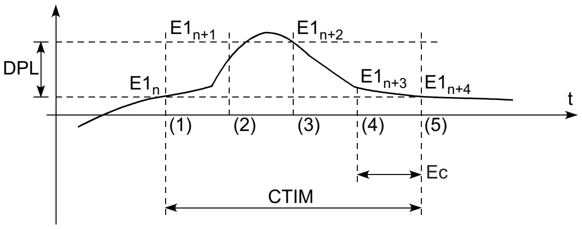
$$m=\frac{CTIM}{\triangle T}$$

Set CTIM and ΔT so that $m \geq 2$.

No processing is performed when m = 0 (integral part).

Ex.

When m = 4, operations are performed as shown below.



(1) 0th time: $E1_n$-$E1_{n-4}$
(2) 1st time: $E1_n$-$E1_{n-4}$
(3) 2nd time: $E1_n$-$E1_{n-4}$
(4) 3rd time: $E1_n$-$E1_{n-4}$
(5) 4th time: $E1_{n+4}$-$E1_n$
Ec: Execution cycle

- A change in the input data and variation rate alarm value (DPL) are compared every execution cycle (ΔT).

Check item	Condition	ALM	BB4	BB5
Variation rate check	$E1_{n+m}-E1_n \geq DPL$	DPPA = 1[*1]	1[*1]	—
	Others	DPPA = 0	0	—
	$E1_{n+m}-E1_n \leq -DPL$	DPNA = 1[*2]	—	1[*2]
	Others	DPNA = 0	—	0

*1 If ERRI or DPPI of disable alarm detection (INH) is set to 1, the DPPA of ALM and BB4 are set to 0.

*2 If ERRI or DPNI of disable alarm detection (INH) is set to 1, the DPNA of ALM and BB5 are set to 0.

■Engineering value transformation (4)

The instruction performs engineering value transformation using the following expression.

$$PV=\frac{RH-RL}{100}\times E1+RL$$

■Loop stop processing (5)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.PHPL instruction ends. • The instruction performs engineering value inverse transformation using the following expression. $BW=\frac{100}{RH-RL}\times(PV-RL)$ • The alarm bits (BB1, BB2, BB3, BB4, and BB5) are set to 0. • The DMLA, MHA, and DLA of alarm detection (ALM) are set to 0.
0	The loop runs and "engineering value inverse transformation (1)" is performed.

Operation error

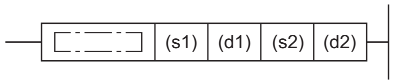
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Variation rate alarm value (DPL) <- Variation rate alarm value (DPL)
	The execution cycle (ΔT) setting is less than 0.
	Variation rate alarm check time (CTIM) < 0
	(Variation rate alarm check time (CTIM)÷execution cycle (ΔT)) > 32767

Lead-lag compensation

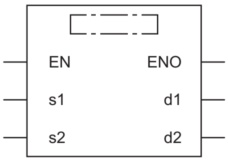
S.LLAG

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

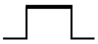
This instruction performs lead-lag compensation for the input data and outputs the operation result.

Ladder	ST
	ENO:=S_LLAG(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.LLAG	

Setting data

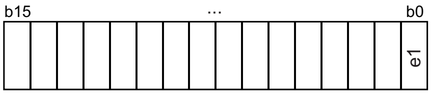
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Local work memory start device	Refer to "Local work memory".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User
+2	e1	Actuating signal	 0: Lead-lag compensation 1: No lead-lag compensation	16-bit unsigned binary	—	User

■Block memory

Operand: (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999 [%]	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	T_1	Lag time	0 to 999999 [s]	Single-precision real number	1.0	User
+2 +3	T_2	Lead time	0 to 999999 [s]	Single-precision real number	1.0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

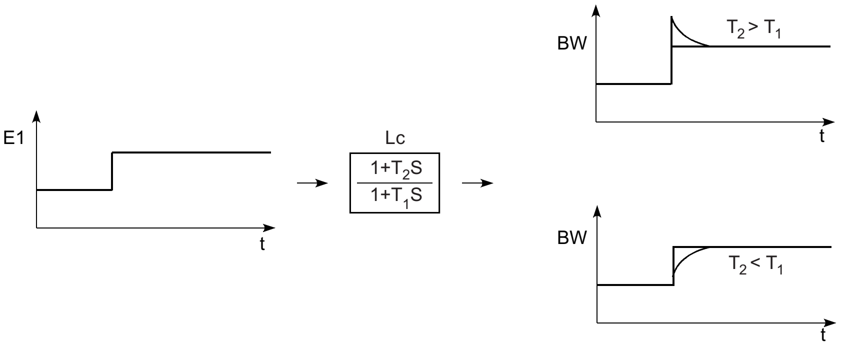
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	$E1_{n-1}$	Last input value	—	Single-precision real number	—	System

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction sets the lag time (T_1) of (s2) and the lead time (T_2) and performs lead-lag compensation according to the actuating signal (e1).



Lc: Lead-lag compensation

The S.LLAG instruction performs the following operations.

Condition	BW (output value)
e1 = 0	$\mathrm{BW}=\frac{1}{T_1+\triangle T}\times\{T_2\times(\mathrm{E1}-\mathrm{E1}_{n-1})+T_1\times\text{Last BW value}+\triangle T\times \mathrm{E1}\}$ However, BW = 0 when T+ΔT = 0.
e1 = 1	BW = E1 (The input value is output as is.)

Operation error

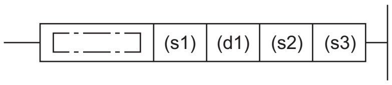
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (d1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Lag time (T_1) < 0 or lead time (T_2) < 0
	The execution cycle (ΔT) setting is less than 0.

Integral control

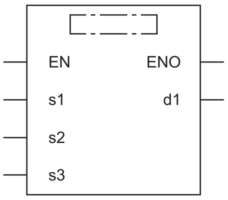
S.I

✕ RnCPU ✕ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

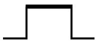
This instruction performs lead-lag compensation for the input data and outputs the operation result.

Ladder	ST
	ENO:=S_I(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.I	

Setting data

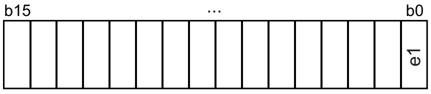
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User
+2	e1	Operation control signal	 0: Integration operation performed 1: Integration operation not performed	16-bit unsigned binary	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

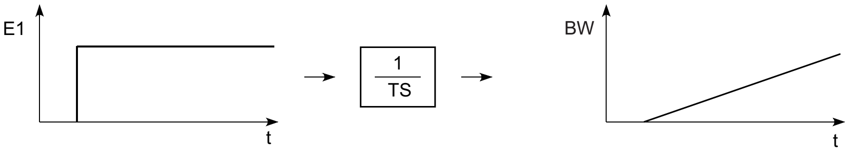
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	T	Integral time	0 to 999999 [s]	Single-precision real number	1.0	User
+2 +3	Ys	Output initial value	-999999 to 999999	Single-precision real number	0.0	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs integral operation according to the operation control signal (e1).



The S.I instruction performs the following operations.

e1	T	BW
0	≠0	$BW=Y_n=\frac{\Delta T}{T}\times E1+Y_{n-1}$
0	0	BW=Yn-1
1	-	BW=Ys

E1: Input value of this time, ΔT: Execution cycle, Yn: Output value of this time, Yn-1: Last output value

Operation error

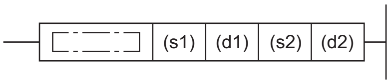
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (d1), or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Derivative control

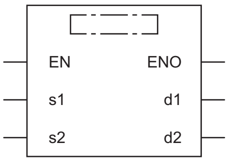
S.D

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction performs differentiation operation for the input data, and outputs the operation result.

Ladder	ST
	ENO:=S_D(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.D	

Setting data

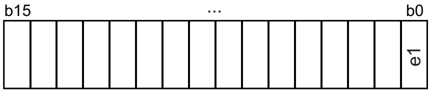
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Local work memory start device	Refer to "Local work memory".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User
+2	e1	Operation control signal	 0: Differentiation operation performed 1: Differentiation operation not performed	16-bit unsigned binary	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	T	Derivative time	0 to 999999 [s]	Single-precision real number	1.0	User
+2 +3	Ys	Output initial value	-999999 to 999999	Single-precision real number	0.0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

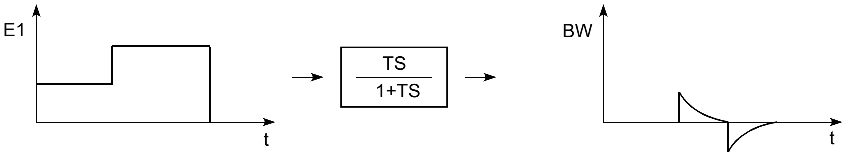
Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	$E1_{n-1}$	Last input value	—	Single-precision real number	—	System

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs differentiation operation according to the operation control signal (e1).



The S.D instruction performs the following operations.

e1	BW
0	$BW=\frac{T}{T+\triangle T}\times(Y_{n-1}-E1_{n-1}+E1)$ However, BW = 0 when T+ΔT = 0.
1	BW=Ys

E1: Input value of this time, ΔT: Execution cycle, $E1_{n-1}$: Last input value, Y_{n-1}: Last output value

Operation error

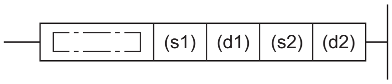
Error code (SD0)	Description
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (d1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Dead time

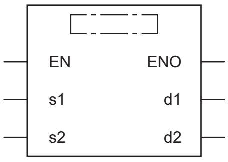
S.DED

RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

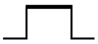
This instruction outputs the input data with the delay by the specified dead time.

Ladder	ST
	ENO:=S_DED(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DED	

Setting data

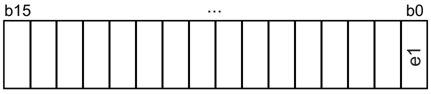
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

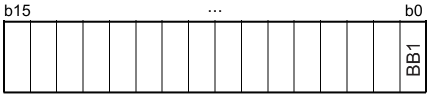
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

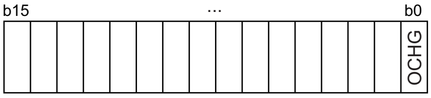
■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User
+2	e1	Operation control signal	 0: Dead time 1: No dead time	16-bit unsigned binary	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB BB1	Data sufficiency bit	 0: Data sufficient 1: Data insufficient	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	ST	Data sampling interval	0 to 999999 Set a value within the following range. $\frac{ST}{\triangle T}$<=32767	Single-precision real number	1.0	User
+2	SN	Sampling count	0 to 48	16-bit unsigned binary	0	User
+3 +4	Ys	Output initial value	-999999 to 999999	Single-precision real number	0.0	User
+5	OCHG	Initial output switching	 0: Output E1 when e1 has changed from 1 to 0 SN times. 1: Output Ys up to SN.	16-bit unsigned binary	0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

Operand: (d2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	—	Last value input (e1')	—	—	—	System
+1		Cycle counter[*1]				
+2		Number of data blocks stored in the dead time table				
+3 +4		Dead time table 1				
+5 +6		Dead time table 2				
⋮		⋮				
+2SN+1 +2SN+2		Dead time table SN				

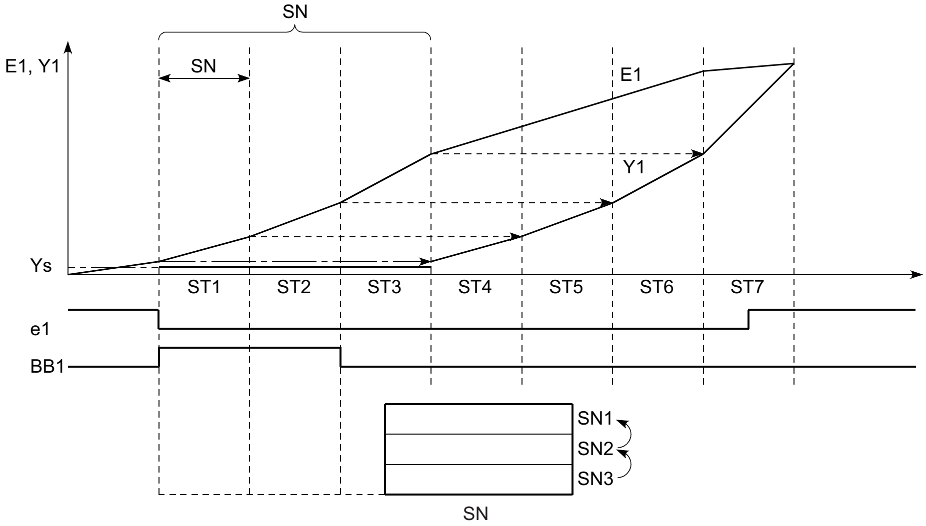
*1 The cycle counter value is rounded off to the nearest whole number.

■Execution cycle (∆T)

Set the execution cycle (∆T) in SD816 and SD817 in single-precision real number.

Processing details

The input value (E1) is output with a delay by the dead time according to the content of the operation control signal (e1).



SN: Dead time table

SN: Sampling count, ST: Data collection interval, E1: Input value, YS: Initial output value

The S.DED instruction performs the following operations.

e1	OCHG	Dead time	BW	
1	0/1	—	E1	
1→0	0	ST×SN	Up to SN	E1 when e1 changed from 1 to 0
			After SN	Least recent data [*1]
	1		Up to SN	YS
			After SN	Least recent data[*1]
0→0	0/1	ST×SN	Least recent data[*1]	

*1 Least recent data is an input value (E1) after SN.

- When the dead time table does not have sufficient data, the data sufficiency bit (BB1) is set to 1.
- When the sampling count (SN) is 0, the data sufficiency bit (BB1) is 0 and the output value (BW) equals the input value (E1).

Operation error

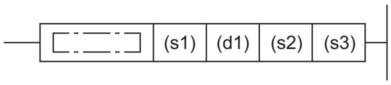
Error code (SD0)	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The execution cycle (ΔT) setting is less than 0.
	The sampling count (SN) is less than 0 or greater than 48.
	The data sampling interval (ST) is less than 0.
	(Data sampling interval (ST)÷execution cycle (ΔT)) is greater than 32767.

High selector

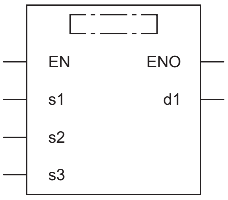
S.HS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction outputs only the maximum value among the input data.

Ladder	ST
	ENO:=S_HS(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.HS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	1 to 16	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	Maximum value in E1 to En	Single-precision real number	—	System
+2	BB	BB1 to BB16	Output selection	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 BB16 BB15 BB14 BB13 BB12 BB11 BB10 BB9 BB8 BB7 BB6 BB5 BB4 BB3 BB2 BB1 0: Corresponding input value is not the maximum value. 1: Corresponding input value is the maximum value.	16-bit unsigned binary	—	System

Processing details

This instruction outputs the maximum value out of input values E1 to En.

■High selector processing

The maximum value out of input values E1 to En is stored in the output value (BW).

In addition, the BB output selection (BB1 to BB16) corresponding to the maximum value is set to 1.

The correspondence between input values 1 (E1) to 16 (E16) and BB output selections (BB1 to BB16) is shown below.

Input value	E16	E15	E14	to	E2	E1
Bit to be set to 1 at the maximum value	BB16	BB15	BB14	to	BB2	BB1

Condition	Processing	
Two or more maximum values exist.	The bits corresponding to the maximum values are all set to 1.	
Only one input	Only input value 1 (E1) is used as the input value.	• The input value 1 (E1) is stored in the output value (BW). • BB output selection BB1 is set to 1. • BB output selections BB2 to BB16 are set to 0.
	Only one of input values 2 (E2) to 16 (E16) is used as the input value.	A value out of input values 2 (E2) to 16 (E16) and the value of input value 1 (E1) are used for data processing.

Operation error

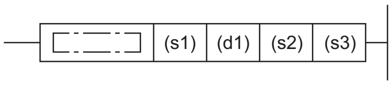
Error code	Error content
3402H	(s1) is a subnormal number or NaN (not a number).
3405H	The number of inputs (n) is less than 1 or greater than 16.

Low selector

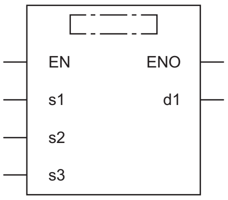
S.LS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

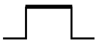
This instruction outputs only the minimum value among the input data.

Ladder	ST
	ENO:=S_LS(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.LS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	n	Number of inputs	1 to 16	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	Minimum value in E1 to En	Single-precision real number	—	System
+2	BB	BB1 to BB16	Output selection	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 BB16 BB15 BB14 BB13 BB12 BB11 BB10 BB9 BB8 BB7 BB6 BB5 BB4 BB3 BB2 BB1 0: Corresponding input value is not the minimum value. 1: Corresponding input value is the minimum value.	16-bit unsigned binary	—	System

Processing details

This instruction outputs the minimum value out of input values E1 to En.

■Low selector processing

The minimum value out of input values E1 to En is stored in the output value (BW).

In addition, the BB output selection (BB1 to BB16) corresponding to the minimum value is set to 1.

The correspondence between input values 1 (E1) to 16 (E16) and BB output selections (BB1 to BB16) is shown below.

Input value	E16	E15	E14	to	E2	E1
Bit to be set to 1 at the minimum value	BB16	BB15	BB14	to	BB2	BB1

Condition	Processing	
Two or more minimum values exist.	The bits corresponding to the minimum values are all set to 1.	
Only one input	Only input value 1 (E1) is used as the input value.	• The input value 1 (E1) is stored in the output value (BW). • BB output selection BB1 is set to 1. • BB output selections BB2 to BB16 are set to 0.
	Only one of input values 2 (E2) to 16 (E16) is used as the input value.	A value out of input values 2 (E2) to 16 (E16) and the value of input value 1 (E1) are used for data processing.

Operation error

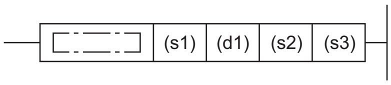
Error code	Error content
3402H	(s1) is a subnormal number or NaN (not a number).
3405H	The number of inputs (n) is less than 1 or greater than 16.

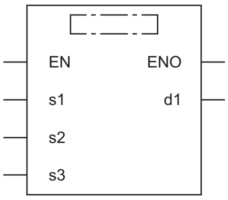
Middle value selector

S.MID

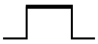
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction outputs the intermediate values between the maximum and minimum values among the input data.

ST	Ladder
	ENO:=S_MID(EN,s1,s2,s3,d1);

FBD/LD


■Execution condition

Instruction	Execution condition
S.MID	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	1 to 16	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	Intermediate value between maximum and minimum values	Single-precision real number	—	System
+2	BB	BB1 to BB16	Output selection	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 BB16 BB15 BB14 BB13 BB12 BB11 BB10 BB9 BB8 BB7 BB6 BB5 BB4 BB3 BB2 BB1 0: Corresponding input value is not the intermediate value. 1: Corresponding input value is the intermediate value.	16-bit unsigned binary	—	System

Processing details

This instruction outputs an intermediate value between the maximum and minimum values among input values E1 to En.

■Middle value selector processing

An intermediate value among input values E1 to En is stored in the output value (BW).

In addition, the BB output selection (BB1 to BB16) corresponding to the intermediate value is set to 1.

The correspondence between input values 1 (E1) to 16 (E16) and BB output selections (BB1 to BB16) is shown below.

Input value	E16	E15	E14	to	E2	E1
Bit to be set to 1 at an intermediate value	BB16	BB15	BB14	to	BB2	BB1

- When the number of inputs is an even number, the smaller value among the intermediate values is stored.
- When two or more intermediate values exist, the bits corresponding to the intermediate values are all set to 1.

■Remarks

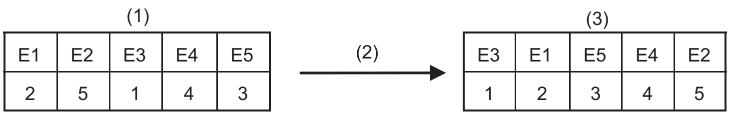
Intermediate values are selected as follows:

- Input values 1 (E1) to n (En) are sorted in ascending order. (When there are same input values, they are sorted in ascending order of their input numbers.)
- The intermediate value in the sorted values is selected.

Ex.

When the input data are 2, 5, 1, 4, and 3, operations are performed as shown below.

In this example, 3 is the intermediate value and accordingly the output select (BB5) is set to 1.



(1) Input data

(2) Rearrangement

(3) Rearranged data

Operation error

Error code	Error content
3402H	(s1) is a subnormal number or NaN (not a number).
3405H	The number of inputs (n) is less than 1 or greater than 16.

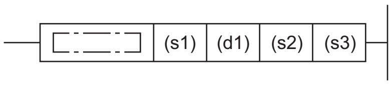
Average value calculation

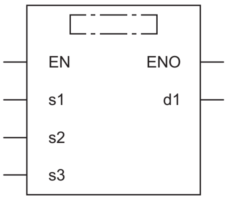
S.AVE

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

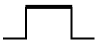
Calculates and outputs the mean value of the input data.

■Execution condition

Ladder	ST
	ENO:=S_AVE(EN,s1,s2,s3,d1);

FBD/LD


■Execution condition

Instruction	Execution condition
S.AVE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	1 to 16	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	Mean value of E1 to En	Single-precision real number	—	System

Processing details

This instruction calculates and outputs the mean value of input values E1 to En.

■Average value calculation

The instruction calculates the mean value of input values E1 to En.

The numerical value specified by the number of inputs (n) is used as the denominator.

$$\mathrm{BW}=\frac{\mathrm{E1}+\mathrm{E2}+\mathrm{E3}\cdots\mathrm{En}}{\mathrm{N}}$$

Operation error

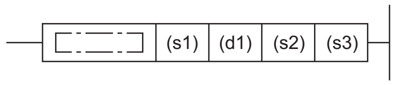
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	(s1) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of inputs (n) is less than 1 or greater than 16.

Upper/lower limiter

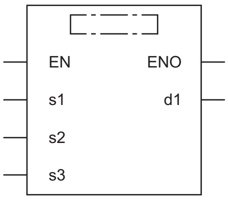
S.LIMT

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

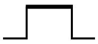
This instruction applies a limiter with hysteresis to the output value.

Ladder	ST
	ENO:=S_LIMT(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.LIMT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

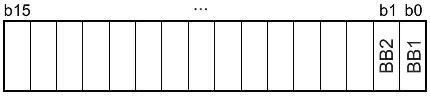
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Upper limit alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Lower limit alarm				

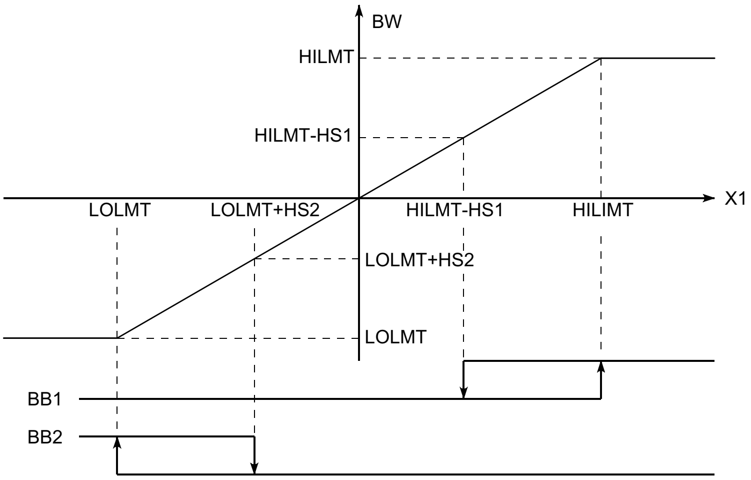
■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	HILMT	Upper limit value[*1]	-999999 to 999999 [%]	Single-precision real number	100.0	User
+2 +3	LOLMT	Lower limit value[*1]	-999999 to 999999 [%]	Single-precision real number	0.0	User
+4 +5	HS1	Upper limit hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User
+6 +7	HS2	Lower limit hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User

[*1] Set the upper limit (HILMT) and lower limit values (LOLMT) in such a way that HILMT equals to or exceeds LOLMT.

Processing details

This instruction applies upper and lower limiters with hysteresis to the output value.



The S.LIMT instruction performs the following operations.

Condition	BW	BB1	BB2
E1≥HILMT	HILMT	1	0
(LOLMT+HS2)<E1<(HILMT-HS1)	E1	0	0
E1≤LOLMT	LOLMT	0	1
Other than the above (hysteresis part)	E1	Last value	Last value

Operation error

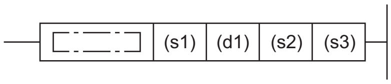
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Upper hysteresis (HS1) is greater than 0 or the lower hysteresis (HS2) is smaller than 0.
	Lower limit value (LOLMT) is greater than upper limit value (HILMT).

Variation rate limiter 1

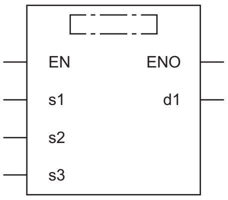
S.VLMT1

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

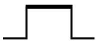
This instruction limits the varying speed and outputs it when the variation rate of input (E1) exceeds the limit.

Ladder	ST
	ENO:=S_VLMT1(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.VLMT1	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

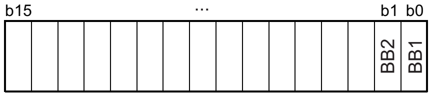
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)

Device	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Positive direction limit alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Negative direction limit alarm				

■Operation constant

Operand: (s2)

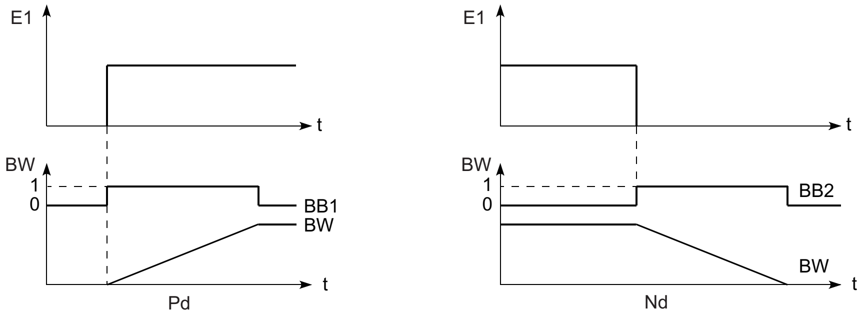
Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	V1	Positive direction limit value	0 to 999999 [%/s]	Single-precision real number	100.0	User
+2 +3	V2	Negative direction limit value	0 to 999999 [%/s]	Single-precision real number	100.0	User
+4 +5	HS1	Positive direction hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User
+6 +7	HS2	Negative direction hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction limits the varying speed and outputs it (BW) when the variation rate of input (E1) exceeds the limit.



Pd: Positive direction
Nd: Negative direction

The S.VLIMT1 instruction performs the following operations.

Condition	Input (E1-BW)	BW	BB1	BB2
Positive direction E1≥BW	(E1-BW)≥(V1×ΔT)	BW=BW+V1×ΔT	1	0
	(E1-BW)<(V1×ΔT-HS1)	BW=E1	0	0
	Others	BW=E1	Last value	Last value
Negative direction E1<BW	(BW-E1)≥(V2×ΔT)	BW=BW-V2×ΔT	0	1
	(BW-E1)<(V2×ΔT-HS2)	BW=E1	0	0
	Others	BW=E1	Last value	Last value

Operation error

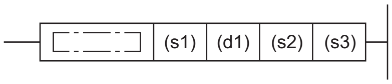
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Positive direction hysteresis (HS1) is less than 0 or the negative direction hysteresis (HS2) is less than 0.

Variation rate limiter 2

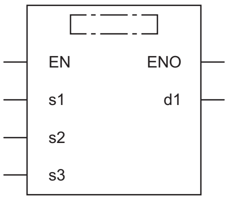
S.VLMT2

✕ RnCPU ✕ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

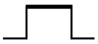
This instruction holds the last value and outputs it when the variation rate of input (E1) exceeds the limit.

Ladder	ST
	ENO:=S_VLMT2(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.VLMT2	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

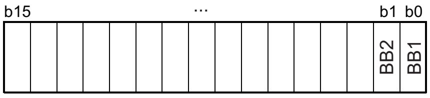
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Positive direction limit alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Negative direction limit alarm				

■Operation constant

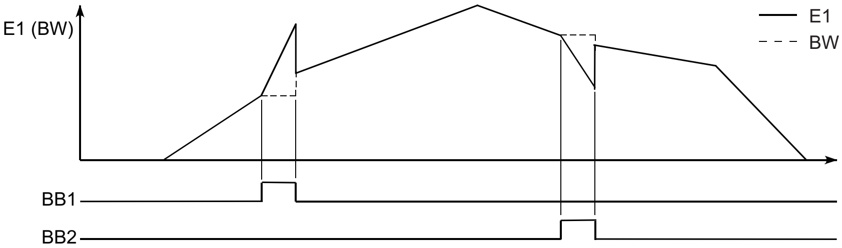
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	V1	Positive direction limit value	0 to 999999 [%/s]	Single-precision real number	100.0	User
+2 +3	V2	Negative direction limit value	0 to 999999 [%/s]	Single-precision real number	100.0	User
+4 +5	HS1	Positive direction hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User
+6 +7	HS2	Negative direction hysteresis	0 to 999999 [%]	Single-precision real number	0.0	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction holds the last value and outputs it when the variation rate of input (E1) exceeds the limit.



E1: Input value, BW: Output value

The S.VLMT2 instruction performs the following operations.

Condition	Input (E1-BW)	BW	BB1	BB2
Positive direction E1≥BW	(E1-BW)≥(V1×ΔT)	BW=BW	1	0
	(E1-BW) < (V1×ΔT-HS1)	BW=E1	0	0
	Others	BW=BW	Last value	Last value
Negative direction E1<BW	(BW-E1)≥(V2×ΔT)	BW=BW	0	1
	(BW-E1)<(V2×ΔT-HS2)	BW=E1	0	0
	Others	BW=BW	Last value	Last value

Operation error

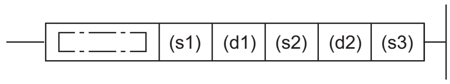
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Positive direction hysteresis (HS1) is less than 0 or the negative direction hysteresis (HS2) is less than 0.

Two-position (on/off) control

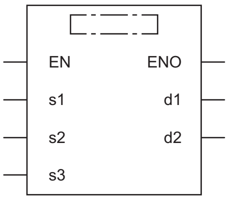
S.ONF2

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

The instruction performs the following steps: SV setting, tracking, MV correction, MV output, and two-position (on/off) control.

Ladder	ST
	ENO:=S_ONF2(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ONF2	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	Set value start device when E2 is used Dummy device when E2 is not used	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

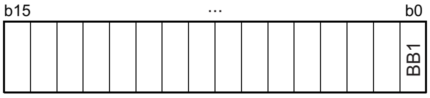
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

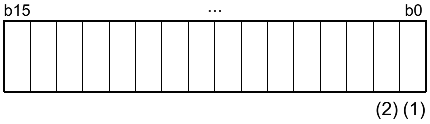
■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User/ system

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB BB1	Operation result	 0: |BW|<50% 1: |BW|≥50%	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User
+1	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+2	SVPTN	Set value pattern	0 to 3  (1) Use of set value Specify whether to use the set value (E2) or not. 0: Used 1: Not used (2) Set value pattern Specify whether to use the upper loop MV as the set value (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	3	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 | | | | | | CSV | CMV | CCB | CAB | CMB | CAS | AUT | MAN | LCC | LCA | LCM |	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b0 | | SPA | | | | | | | | | | | | | | | SPA 0: Loop RUN 1: Loop STOP	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b13 … b0 | | | TRKF | | | | | | | | | | | | | | TRKF 0: Tracking not set 1: Tracking set	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+14 +15	SV	Set value	RL to RH	Single-precision real number	0.0	User
+16 +17	DV	Deviation	-110 to 110 [%]	Single-precision real number	0.0	System
+18 +19	HS0	Hysteresis	0 to 999999	Single-precision real number	0.0	User
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+46 +47	CT	Control cycle	0 to 999999 [s] Set a value within the following range. $\frac{CT}{\triangle T}$<=32767	Single-precision real number	1.0	User

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.
If E2 is not used, specify a dummy device (SD820).

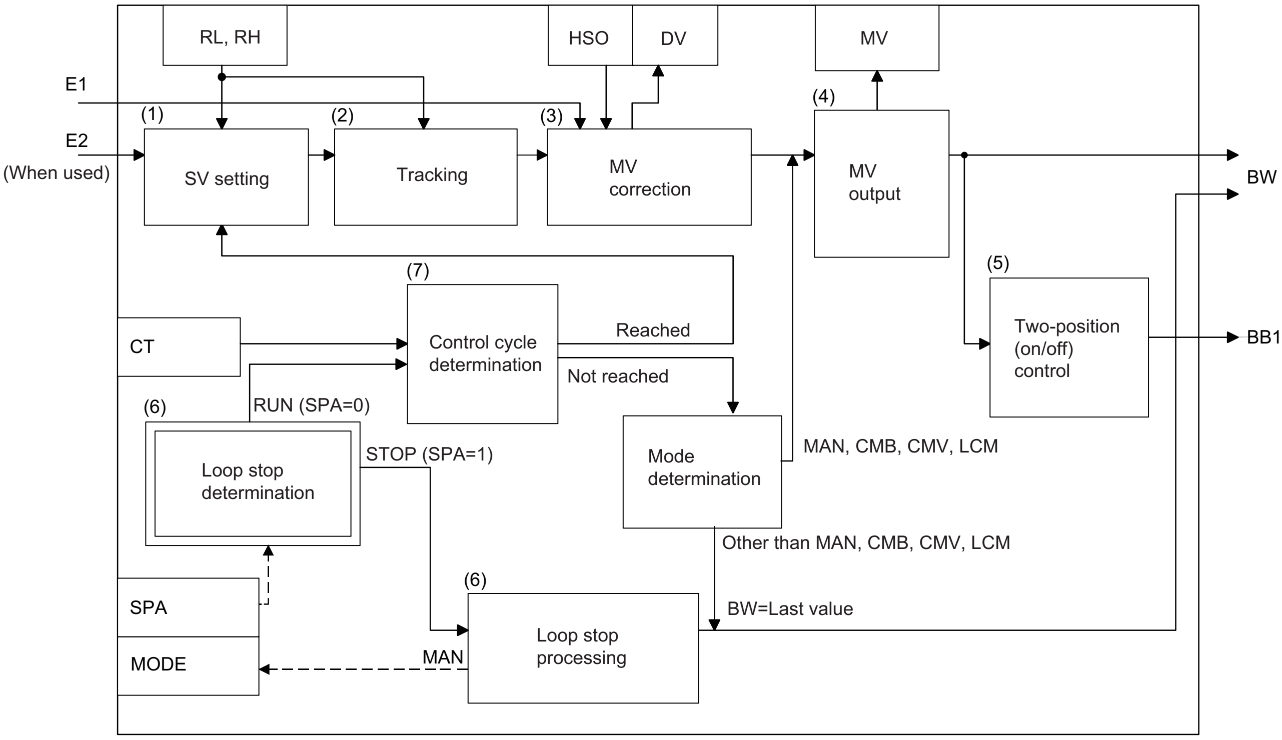
Operand: (s3)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User/ system

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs two-position (on/off) control (1-contact ON/OFF) when the specified control cycle is reached.
The instruction also performs the following steps: SV setting, tracking, MV correction, and MV output processing.
The following is the processing block diagram of the S.ONF2 instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n=\frac{RH-RL}{100}\times E2+RL$ If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

- The set value (SV) is inversely transformed from the engineering value and SVn' is calculated (refer to the following expression).

$$SV_n' = \frac{100}{RH-RL} \times (SV_n - RL)$$

- Tracking processing is performed when all of the following conditions are satisfied.
  - The tracking bit (TRK) is set to 1.
  - The set value (E2) is used.
  - The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

    $E2 = SV_n'$
- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■MV correction (3)

The instruction calculates the deviation (DV) based on the input value (E1) and the set value (SVn') after tracking, and then calculates the MV correction value (MV').

- The deviation (DV) is calculated under the following conditions.

Condition	DV
Direct action (PN = 1)	$E1-SV_n'$
Reserve action (PN = 0)	$SV_n' - E1$

- The MV correction value (MV') is calculated under the following conditions.

Condition	MV'
DV≥HS0	100%
DV≤-HS0	0%
-HS0<DV<HS0	Last value (BW value)

■MV output (4)

The manipulated value (MV (BW)) is calculated under the following conditions.

Condition	BW
CMV, MAN, CMB, LCM	$BW=MV_n$
CSV, CCB, CAB, CAS, AUT, LCC, LCA	BW=MV' $MV_n=BW$

■Two-position (on/off) control (5)

The BB operation result (BB1) is output under the following conditions.

Condition	BB1
$\lvert BW\rvert \geq 50\%$	1
$\lvert BW\rvert < 50\%$	0

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.ONF2 instruction ends. The output value (BW) is held. • The control mode (MODE) is set to MAN. The output result (BB1) value is set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

- If the specified control cycle is not reached, the following processing is performed.

Control mode (MODE)	Processing
CSV, CCB, CAB, CAS, AUT, LCC, or LCA	The output value (BW) is held and the S.ONF2 instruction ends.
MAN, CMB, CMV, or LCM	"Three-position (on/off) control (5) is processed assuming that the output value (BW) equals the manipulated value (MV).

- If the specified control cycle is reached, "SV setting processing (1)" is performed.

Operation error

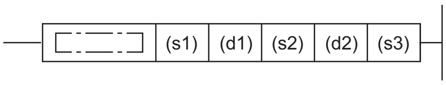
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	Hysteresis (HS0) <- hysteresis (HS0)
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Three-position (on/off) control

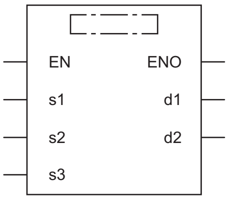
S.ONF3

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

The instruction performs the following steps: SV setting, tracking, MV correction, MV output, and three-position (on/off) control.

Ladder	ST
	ENO:=S_ONF3(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ONF3	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	Set value start device when E2 is used Dummy device when E2 is not used	Refer to "Set value".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

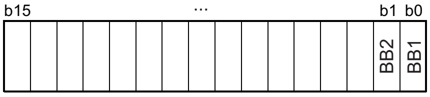
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

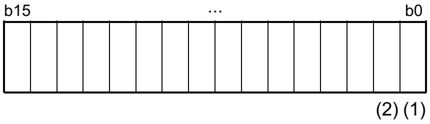
Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

<table>
<tr><th colspan="8">Operand: (d1)</th></tr>
<tr><th>Device</th><th colspan="2">Symbol</th><th>Name</th><th>Recommended range</th><th>Data type</th><th>Standard value</th><th>Set by</th></tr>
<tr><td>+0
+1</td><td colspan="2">BW</td><td>Output value</td><td>-999999 to 999999 [%]</td><td>Single-precision real number</td><td>—</td><td>System</td></tr>
<tr><td rowspan="2">+2</td><td rowspan="2">BB</td><td>BB1</td><td>Operation result</td><td rowspan="2">

<table>
<tr><th>BW</th><th>BB2</th><th>BB1</th></tr>
<tr><td>BW<25%</td><td>1</td><td>0</td></tr>
<tr><td>25%<=BW<75%</td><td>0</td><td>0</td></tr>
<tr><td>BW>=75%</td><td>0</td><td>1</td></tr>
</table></td><td rowspan="2">16-bit unsigned binary</td><td rowspan="2">—</td><td rowspan="2">System</td></tr>
<tr><td>BB2</td><td>Operation result</td></tr>
</table>

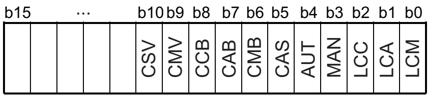
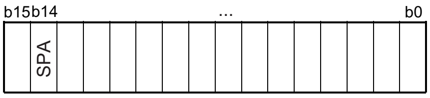
■Operation constant

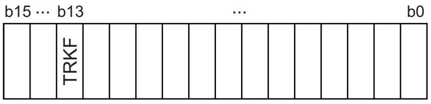
<table>
<tr><th colspan="7">Operand: (s2)</th></tr>
<tr><th>Device</th><th>Symbol</th><th>Name</th><th>Recommended range</th><th>Data type</th><th>Standard value</th><th>Set by</th></tr>
<tr><td>+0</td><td>PN</td><td>Operation mode</td><td>0: Reverse action
1: Direct action</td><td>16-bit unsigned binary</td><td>0</td><td>User</td></tr>
<tr><td>+1</td><td>TRK</td><td>Tracking bit</td><td>0: Tracking not performed
1: Tracking performed</td><td>16-bit unsigned binary</td><td>0</td><td>User</td></tr>
<tr><td>+2</td><td>SVPTN</td><td>Set value pattern</td><td>0 to 3


(1) Use of set value
Specify whether to use the set value (E2) or not.
0: Used
1: Not used
(2) Set value pattern
Specify whether to use the upper loop MV as the set value (E2) or not.
0: E2 is the upper loop MV.
1: E2 is not the upper loop MV.</td><td>16-bit unsigned binary</td><td>3</td><td>User</td></tr>
</table>

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

<table>
<tr><th colspan="7">Operand: (d2)</th></tr>
<tr><th>Position</th><th>Symbol</th><th>Name</th><th>Recommended range</th><th>Data type</th><th>Standard value</th><th>Set by</th></tr>
<tr><td>+1</td><td>MODE</td><td>Control mode</td><td>0 to FFFFH

</td><td>16-bit unsigned binary</td><td>8H</td><td>User/ system</td></tr>
<tr><td>+3</td><td>ALM</td><td>Alarm detection</td><td>0 to FFFFH


SPA
0: Loop RUN
1: Loop STOP</td><td>16-bit unsigned binary</td><td>4000H</td><td>User/ system</td></tr>
<tr><td>+4</td><td>INH</td><td>Disable alarm detection</td><td>0 to FFFFH


TRKF
0: Tracking not set
1: Tracking set</td><td>16-bit unsigned binary</td><td>4000H</td><td>User/ system</td></tr>
<tr><td>+12
+13</td><td>MV</td><td>Manipulated value</td><td>-10 to 110 [%]</td><td>Single-precision real number</td><td>0.0</td><td>User/ system</td></tr>
<tr><td>+14
+15</td><td>SV</td><td>Set value</td><td>RL to RH</td><td>Single-precision real number</td><td>0.0</td><td>User</td></tr>
<tr><td>+16
+17</td><td>DV</td><td>Deviation</td><td>-110 to 110 [%]</td><td>Single-precision real number</td><td>0.0</td><td>System</td></tr>
<tr><td>+18
+19</td><td>HS0</td><td>Hysteresis 0</td><td>0 to 999999</td><td>Single-precision real number</td><td>0.0</td><td>User</td></tr>
<tr><td>+20
+21</td><td>HS1</td><td>Hysteresis 1</td><td>0 to 999999</td><td>Single-precision real number</td><td>0.0</td><td>User</td></tr>
<tr><td>+22
+23</td><td>RH</td><td>Engineering value upper limit</td><td>-999999 to 999999</td><td>Single-precision real number</td><td>100.0</td><td>User</td></tr>
<tr><td>+24
+25</td><td>RL</td><td>Engineering value lower limit</td><td>-999999 to 999999</td><td>Single-precision real number</td><td>0.0</td><td>User</td></tr>
<tr><td>+46
+47</td><td>CT</td><td>Control cycle</td><td>0 to 999999 [s]
Set a value within the following range.
$\frac{CT}{\triangle T}$ <=32767</td><td>Single-precision real number</td><td>1.0</td><td>User</td></tr>
</table>

■Loop tag past value memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

The loop tag past value memory occupies 32 words after the loop tag memory.

Operand: (d2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+96	—	Control cycle counter initialization completion flag	—	—	—	System
+97		Control cycle counter (The value is rounded off to the nearest whole number.)				

■Set value

The set value (E2) is valid only when b0 of the set value pattern (SVPTN) is set to 0 (Used). To use the upper loop MV as the set value (E2), specify the device (offset +12) where the manipulated value (MV) of the upper loop is set.

If E2 is not used, specify a dummy device (SD820).

Operand: (s3)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Set value	-10 to 110 [%]	Single-precision real number	0.0	User/ system

■Execution cycle (ΔT)

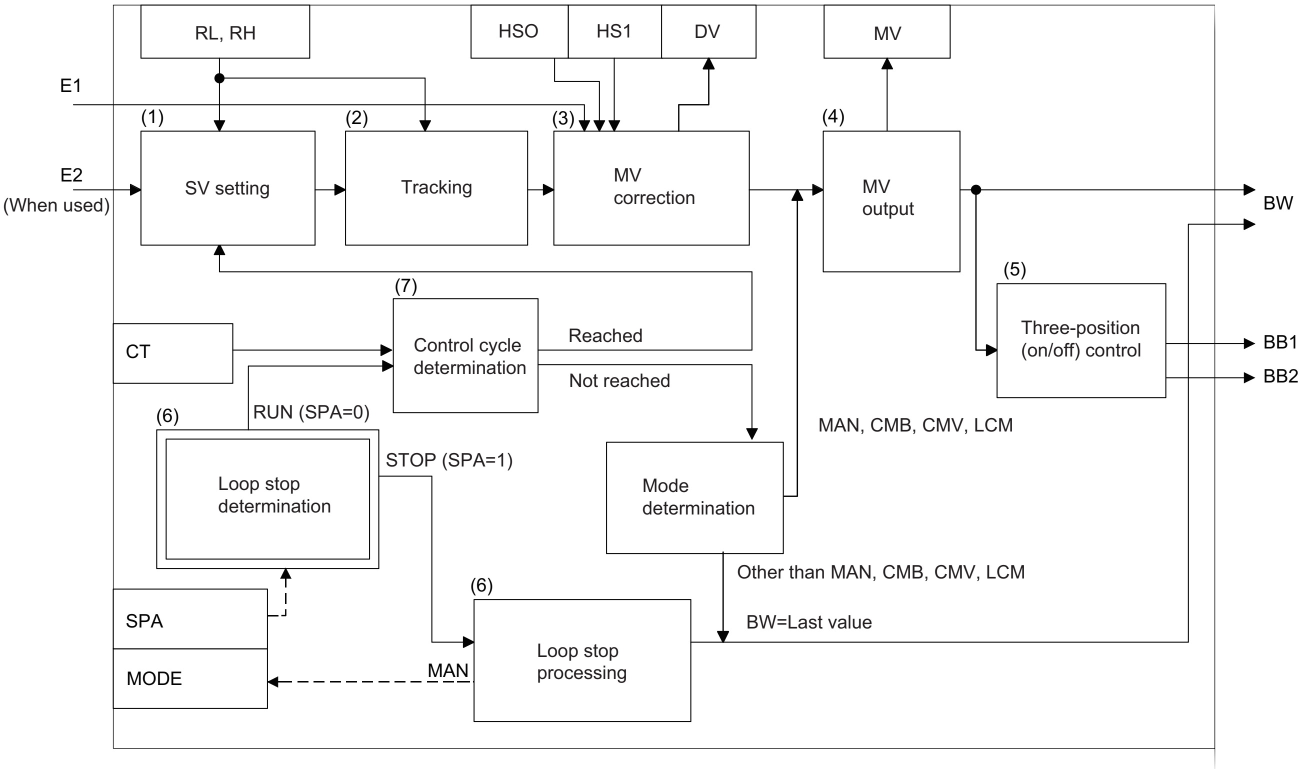
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs two-position (on/off) control (2-contact ON/OFF) every control cycle.

The instruction also performs the following steps: SV setting, tracking, MV correction, and MV output processing.

The following is the processing block diagram of the S.ONF3 instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■SV setting processing (1)

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
CAS, CCB, CSV	If the set value (E2) is specified, engineering value transformation processing (refer to the following expression) is performed, and then "tracking processing (2)" is performed. $SV_n = \frac{RH-RL}{100} \times E2+RL$ If the set value (E2) is not specified, "tracking processing (2)" is performed without performing engineering value transformation processing.
MAN, AUT, CMV, CMB, CAB, LCM, LCA, LCC	"Tracking processing (2)" is performed.

■Tracking processing (2)

The set value (SV) is inversely transformed from the engineering value and SVn' is calculated (refer to the following expression).

$$SV_n' = \frac{100}{RH-RL} \times (SV_n - RL)$$

Tracking processing is performed when all of the following conditions are satisfied.

- The tracking bit (TRK) is set to 1.
- The set value (E2) is used.
- The control mode (MODE) is set to any of the following: MAN, AUT, CMV, CMB, CAB, LCM, LCA, or LCC.

  $E2 = SV_n'$

- If the set value (E2) is the upper loop MV, the TRKF of the disable alarm detection (INH) of the upper loop is set to 1.

■MV correction (3)

The instruction calculates the deviation (DV) based on the input value (E1) and the set value (SVn') after tracking, and then calculates the MV correction value (MV').

- The deviation (DV) is calculated under the following conditions.

Condition	DV
Direct action (PN = 1)	$E1 - SV_n'$
Reserve action (PN = 0)	$SV_n' - E1$

- The MV correction value (MV') is calculated under the following conditions.

Condition	MV'
$DV \geq (HS1+HS0)$	100%
$DV \leq -(HS1+HS0)$	0%
$(-HS1+HS0) < DV < (HS1-HS0)$	50%
Others	Last value (BW value)

■MV output (4)

The manipulated value (MV (BW)) is calculated under the following conditions.

Condition	BW
CMV, MAN, CMB, LCM	$BW=MV_n$
CSV, CCB, CAB, CAS, AUT, LCC, LCA	$BW=MV'$ $MV_n=BW$

■Three-position (on/off) control (5)

The BB operation results (BB1, BB2) are output under the following conditions.

Condition	BB1	BB2
$BW \geq 75\%$	1	0
$25\% \leq BW < 75\%$	0	0
$BW < 25\%$	0	1

■Loop stop processing (6)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.ONF3 instruction ends. • The output value (BW) is held. • The control mode (MODE) is set to MAN. • The output result (BB1, BB2) values are set to 0.
0	The loop runs and "control cycle determination processing (7)" is performed.

■Control cycle determination (7)

- If the specified control cycle is not reached, the following processing is performed.

Control mode (MODE)	Processing
CSV, CCB, CAB, CAS, AUT, LCC, or LCA	The output value (BW) is held and the S.ONF3 instruction ends.
MAN, CMB, CMV, or LCM	"Three-position (on/off) control (5) is processed assuming that the output value (BW) equals the manipulated value (MV).

- If the control cycle (CT) is reached, "SV setting processing (1)" is performed.

Operation error

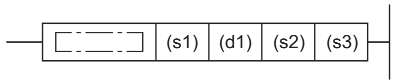
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Hysteresis 0 (HS0) is less than 0.
	The control cycle (CT) setting is less than 0.
	The execution cycle (ΔT) setting is less than 0.
	(Hysteresis 1 (HS1) + hysteresis 0 (HS0)) is less than 0.
	Hysteresis 1 (HS1) is less than 0.
	The value divided the control cycle (CT) by the execution cycle (ΔT) exceeds 32767.

Dead band

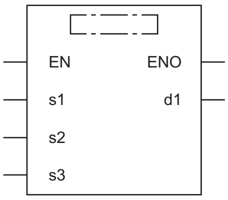
S.DBND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

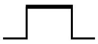
This instruction provides a dead band and performs output processing.

Ladder	ST
	ENO:=S_DBND(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DBND	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

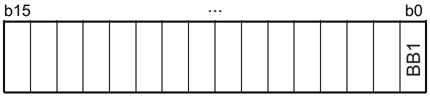
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

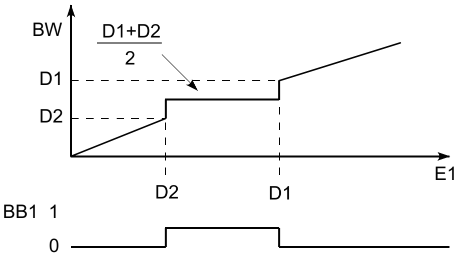
Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Dead band operation	 0: Out of dead band range 1: Within dead band range	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	D1	Dead band upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	D2	Dead band lower limit	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction provides a dead band and performs output processing.



D1: Dead band upper limit, D2: Dead band lower limit

The S.DBND instruction performs the following operations.

Condition	BW	BB1
D2≤E1≤D1	$\frac{D2+D1}{2}$	1
(E1<D2) or (E1>D1)	E1	0

Operation error

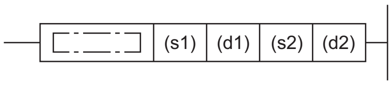
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Program setter

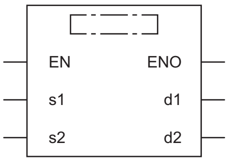
S.PGS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

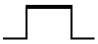
This instruction provides control output according to the SV and MV patterns.

Ladder	ST
	ENO:=S_PGS(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PGS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	—	—	—	—	—	—	—	—	○	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999 [%]	Single-precision real number	—	System
+2	BB	BB1	Alarm	b15 … b2 b1 b0 BB3 BB2 BB1 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Output upper limit alarm				
		BB3	Output lower limit alarm				

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH b15 b14 … b1 b0 SPA MHA MLA SPA 0: Loop RUN 1: Loop STOP MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH b15 … b1 b0 ERRI MHI MLI 0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+10	PTNO	Number of operation constant break points	0 to 16	16-bit unsigned binary	0	User
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+14 +15	SV	Set value	0 to 999999 [s]	Single-precision real number	0.0	User/ system
+16	TYPE	Operation type	Control mode AUT or CAB 0: Hold type operation 1: Return type operation	16-bit unsigned binary	0	User
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+22 +23	SV1	Setting time 1	0 to 999999 [s]	Single-precision real number	0.0	User
⋮	⋮	⋮				
+52 +53	SV16	Setting time 16				

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+54 +55	MV1	Set output 1	-10 to 110 [%]	Single-precision real number	0.0	User
⋮	⋮	⋮				
+84 +85	MV16	Set output 16				

■Execution cycle (ΔT)

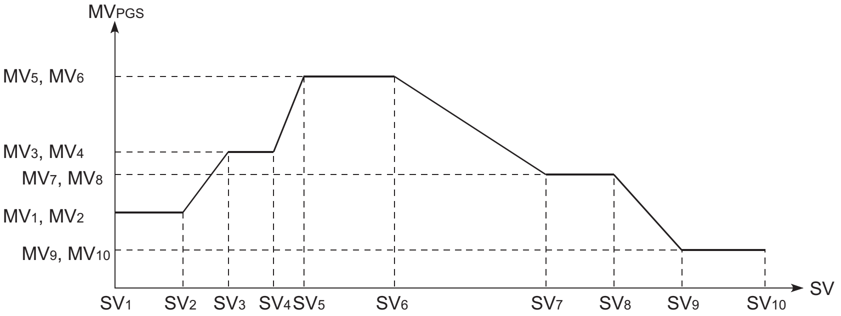
Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

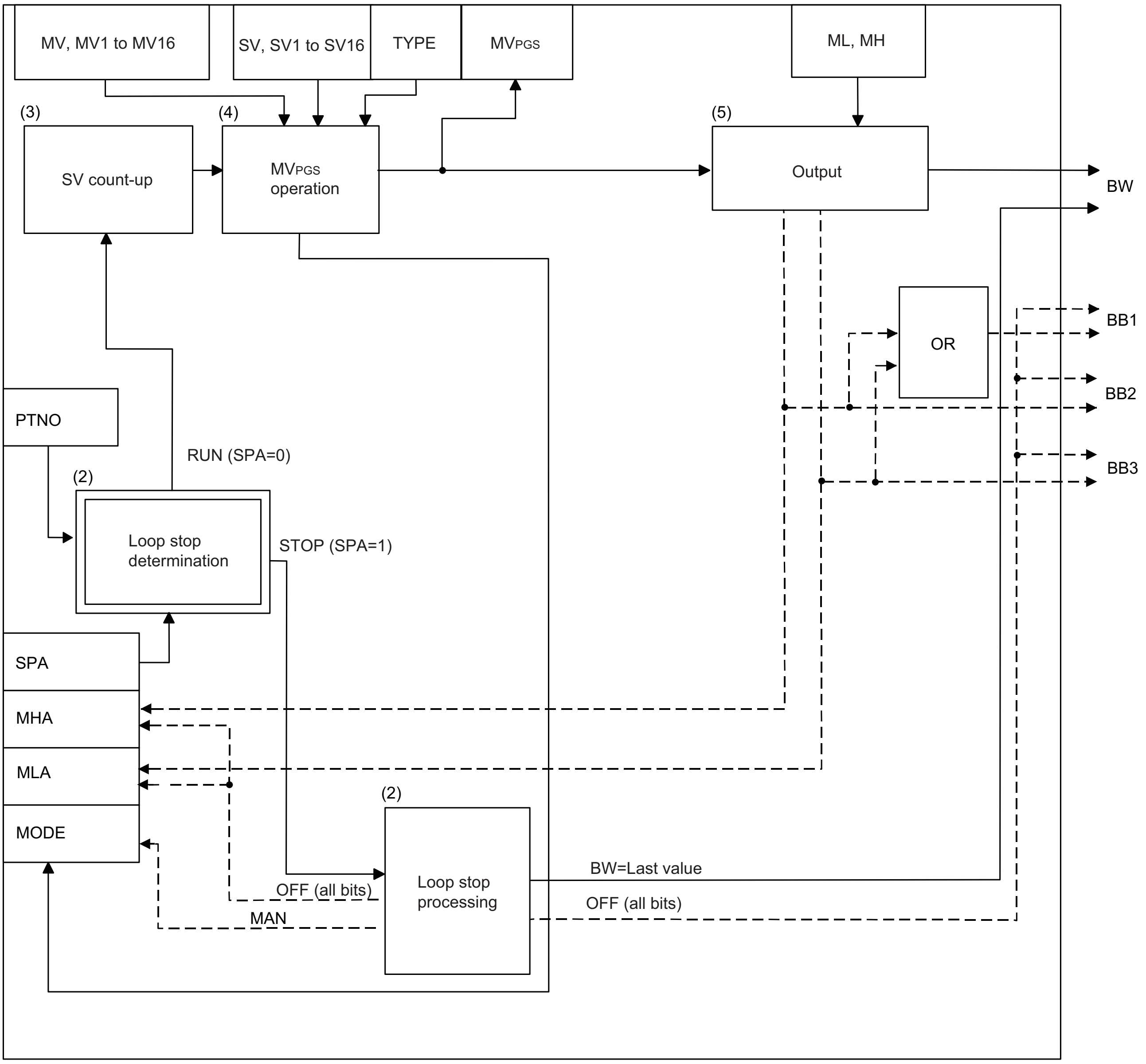
This instruction provides control output according to the SV and MV patterns.

The S.PGS instruction has the following three output types.

Output type	Description
Hold type	Outputs data while holding the value of setting time 10 (SV10).
Return type	Sets the set value (SV) to 0 and outputs the last value of the manipulated value (MV).
Cyclic type	Processes based on the setting time 1 (SV1) to 10 (SV10) and then restarts processing from setting time 1 (SV1).



The following is the processing block diagram of the S.PGS instruction. (The numbers (1) to (5) in the diagram indicate the order of the processing.)



■Output type (1)

The output type is determined by the combination of control mode (MODE) and operation type (TYPE) as follows.

Control mode (MODE)	Operation type (TYPE)	Operation
MAN, CMB, CMV, LCM, LCA, LCC	—	Operation stop with the current SV and MV
AUT, CAB	0	Hold type operation
	1	Return type operation
CAS, CCB, CSV	—	Cyclic type operation

■Loop stop processing (2)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops or the number of operation constant break points is 0, the following operations are performed and the S.PGS instruction ends. • The output value (BW) is held. • All of the output alarm bits (BB1, BB2, and BB3) are set to 0. • The MHA and MLA of alarm detection (ALM) are set to 0. • The control mode (MODE) is set to MAN.
0	The loop runs. When the loop runs, "SV count-up processing (3)" is performed.

■SV count-up processing (3)

SV count-up is performed every execution cycle (ΔT) according to the following expression.

$SV'=SV+\Delta T$

■MVPGS operation (4)

MV_{PGS} operation is shown below.

Output type		Hold type	Return type	Cyclic type
Control mode (MODE)		AUT, CAB		CAS, CCB, CSV
MV_{PGS} operation	$SV<SV1$	MV1		
	$SV_{n-1}\leq SV<SV_n$	$\frac{MV_n-MV_{n-1}}{SV_n-SV_{n-1}}\times(SV'-SV_{n-1})+MV_{n-1}$		
Processing at $SV'>SV_n$	0: Mode shift	MAN	MAN	No mode shift
	SV	Last value	0	0
	MV	Last value	Last value	MV1
	Restart method	After SV setting, change to the MAN→AUT mode.	Change to the MAN→AUT mode.	Automatic restart

■Output processing (5)

The output processing conditions are shown below.

Condition	Manual			Automatic		
	MAN, CMB, CMV, LCM, LCA, LCC			AUT, CAB, CAS, CCB, CSV		
	BW	BB2, MHA	BB3, MLA	BW	BB2, MHA	BB3, MLA
$MV_{PGS}>MH$	MV_n	0	0	$MV_n=MH$	1 [*1]	0
$MV_{PGS}<ML$	MV_n	0	0	$MV_n=ML$	0	1 [*2]
Others	MV_n	0	0	$MV_n=MV_{PGS}$	0	0

[*1] If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB2) and the MHA of the alarm detection (ALM) are set to 0.

[*2] If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB3) and the MLA of the alarm detection (ALM) are set to 0.

Operation error

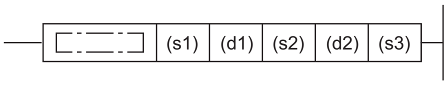
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of operation constant break points (PTNO) is less than 0 or greater than 16.

Loop selector

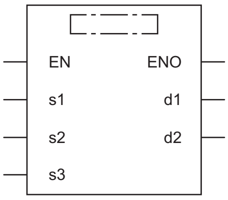
S.SEL

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

In automatic mode, outputs the value selected by the selection signal from the input data, and in manual mode, outputs the manipulated value (MV) in the loop tag memory.

Ladder	ST
	ENO:=S_SEL(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SEL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data 1 start device	Refer to "Input data 1".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(s3)	Input data 2 start device	Refer to "Input data 2".	Single-precision real number
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—

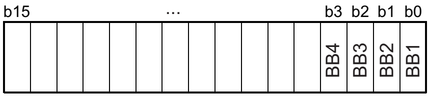
■Input data 1

Operand: (s1)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value 1	-999999 to 999999 [%]	Single-precision real number	—	User

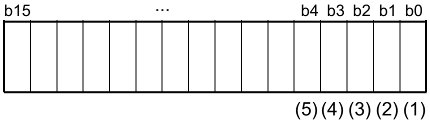
■Block memory

Operand: (d1)

Device	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB	BB1	Alarm	 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Output upper limit alarm				
		BB3	Output lower limit alarm				
		BB4	Output variation rate alarm				

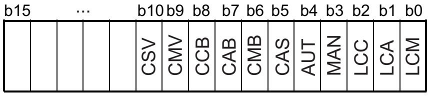
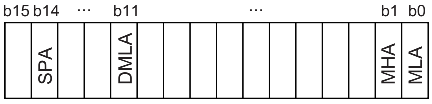
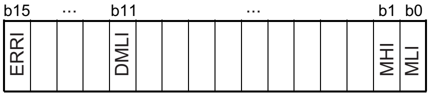
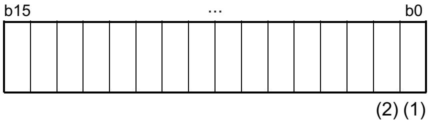
■Operation constant

Operand: (s2)

Position	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	NMAX	Output conversion upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	NMIN	Output conversion lower limit	-999999 to 999999	Single-precision real number	0.0	User
+4	TRK	Tracking bit	0: Tracking not performed 1: Tracking performed	16-bit unsigned binary	0	User
+5	SVPTN	Set value pattern	 (1) Input value selection (e1) Specify whether to use E1 or E2 for the input value. 0: E1 1: E2 (2) Use of input value 1 (E1) Specify whether to use the input value 1 (E1) or not. 0: Use 1: Not use (3) Use of input value 2 (E2) Specify whether to use the input value 2 (E2) or not. 0: Use 1: Not use (4) Input value 1 (E1) pattern Specify whether to use the upper loop MV as the input value 1 (E1) or not. 0: E1 is the upper loop MV. 1: E1 is not the upper loop MV. (5) Input value 2 (E2) pattern Specify whether to use the upper loop MV as the input value 2 (E2) or not. 0: E2 is the upper loop MV. 1: E2 is not the upper loop MV.	16-bit unsigned binary	1EH	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH  	16-bit unsigned binary	8H	User/ system
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP DMLA, MHA, MLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+4	INH	Disable alarm detection	0 to FFFFH  0: Alarm detection enabled 1: Alarm detection disabled	16-bit unsigned binary	4000H	User/ system
+10 +11	PV	Selecting a Value	RL to RH	Single-precision real number	0.0	System
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+14 +15	PV1	Process value 1	RL to RH	Single-precision real number	0.0	System
+16 +17	PV2	Process value 2	RL to RH	Single-precision real number	0.0	System
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+22 +23	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+24 +25	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User
+26	SLNO	Selected No.	 (1) Selection of input value 1 (E1) 0: Not selected 1: Selected (2) Selection of input value 2 (E2) 0: Not selected 1: Selected	16-bit unsigned binary	0	System
+48 +49	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	100.0	User

■Input data 2

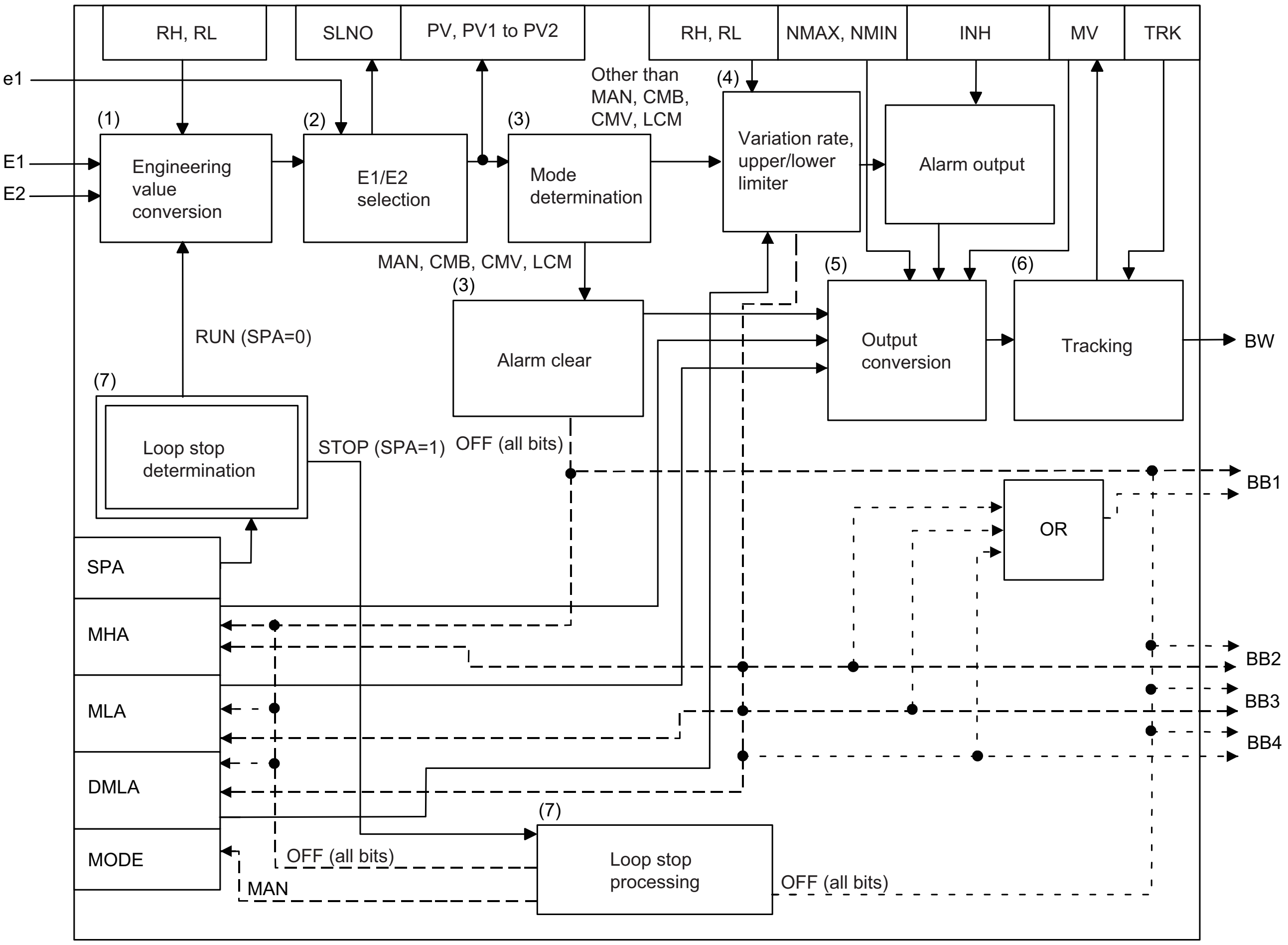
Operand: (s3)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E2	Input value 2	-999999 to 999999 [%]	Single-precision real number	0.0	User

Processing details

This instruction outputs data in the specified mode (automatic or manual mode).

- In automatic mode, the instruction outputs the value selected by the selection signal (e1) from the input values 1 (E1) and 2 (E2).
- In manual mode, it outputs the manipulated value (MV).

The following is the processing block diagram of the S.SEL instruction. (The numbers (1) to (7) in the diagram indicate the order of the processing.)



■Engineering value transformation (1)

The instruction performs engineering value transformation using the following expression.

$$PV_n = \frac{RH-RL}{100} \times E_n + RL$$

■Input value 1 (E1), 2 (E2) selection processing (2)

Specify the input value selection (e1) of the set value pattern (SVPTN) to specify which input value is to be used, 1 (E1) or 2 (E2).

- Input value selection (e1) = 0: Input value 1 (E1) is used. Selected value (PV) = process value 1 (PV1)
- Input value selection (e1) = 1: Input value 2 (E2) is used. Selected value (PV) = process value 2 (PV2)

Selected No. (SLNO): The bit corresponding to the input value E1 or E2 is set to 1.

■Mode check (3)

The following processing is performed depending on the control mode (MODE).

Control mode (MODE)	Processing details
MAN, CMB, CMV, LCM	"Output conversion processing (5)" is performed. The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. The alarm bits (BB1, BB2, BB3, and BB4) are set to 0.
AUT, CAB, CAS, CCB, CSV, LCA, LCC	The instruction performs engineering value inverse transformation using the following expression. $T=\frac{100}{RH-RL}\times(PV-RL)$ "Variation rate & upper/lower limiter processing (4)" is performed.

■Variation rate & upper/lower limiter (4)

The instruction checks the variation rate and upper/lower limit values for the input value 1 (E1) or 2 (E2).

The variation rate limiter conditions are shown below.

Condition	T'	BB4, DMLA
$\lvert T-MV_n\rvert \leq DML$	T'=T	0
$(T-MV_n)>DML$	$T'=MV_n+DML$	1[1]
$(T-MV_n)<-DML$	$T'=MV_n-DML$	1[1]

[1] If the DMLI or ERRI of the disable alarm detection (INH) is set to 1, the output variation rate alarm (BB4) and the DMLA of the alarm detection (ALM) are set to 0.

The upper/lower limiter conditions are shown below.

Condition	MV	BB2, MHA	BB3, MLA
T'>MH	$MV_n=MH$	1[2]	0
T'<ML	$MV_n=ML$	0	1[3]
ML≤T'≤MH	$MV_n=T'$	0	0

[2] If the MHI or ERRI of the disable alarm detection (INH) is set to 1, the output upper limit alarm (BB2) and the MHA of the alarm detection (ALM) are set to 0.

[3] If the MLI or ERRI of the disable alarm detection (INH) is set to 1, the output lower limit alarm (BB3) and the MLA of the alarm detection (ALM) are set to 0.

■Output conversion processing (5)

The instruction performs engineering value transformation using the following expression.

$$BW=\frac{NMAX-NMIN}{100}\times MV_n+NMIN$$

■Tracking processing (6)

Processing is performed when the following conditions are satisfied.

Condition	Processing
When all of the following conditions are satisfied: • The control mode (MODE) is set to any of the following: MAN, CMB, CMV, or LCM. • Tracking bit (TRK) is 1. $E_n=MV_n$	Operation result is output to the input value 1 (E1) or 2 (E2).
When all of the following conditions are satisfied: • The control mode (MODE) is set to any of the following: AUT, CAS, CAB, CCB, CSV, LCA, or LCC. • Tracking bit (TRK) is 1. • BB alarm (BB1) = 1 $E_n=MV_n$	

■Loop stop processing (7)

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.SEL instruction ends. • The output value (BW) is held. • The alarm bits (BB1, BB2, BB3, and BB4) are set to 0. • The MHA, MLA, and DMLA of the alarm detection (ALM) are set to 0. • The control mode (MODE) is set to MAN.
0	Processing from "Engineering value transformation (1)" through to "Tracking processing (6)" is performed.

Operation error

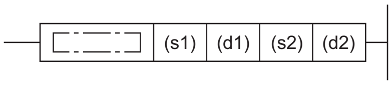
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), (d2), or (s3) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Bumpless transfer

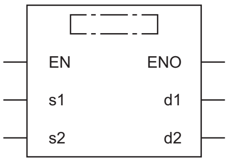
S.BUMP

RnCPU ✕ RnENCPU ✕ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety) ✕

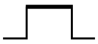
This instruction gradually brings the output value (BW) closer to the output set value (E1) from the output control value (E2) when the mode switching signal (e1) changes from manual to automatic.

Ladder	ST
	ENO:=S_BUMP(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.BUMP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand (s1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Output set value	-999999 to 999999 [%]	Single-precision real number	—	User
+2 +3	E2	Output control value	-999999 to 999999 [%]	Single-precision real number	—	User
+4	e1	Mode switching signal	0: Manual mode 1: Automatic mode	16-bit unsigned binary	—	User

■Block memory

Operand (d1)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999 [%]	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	T	Lag time	0 to 999999 [s]	Single-precision real number	1.0	User
+2 +3	a	Lag band	0 to 999999 [%]	Single-precision real number	1.0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	Xq	Initial deviation value	—	Single-precision real number	1.0	System
+2 +3	Xp	Deviation				

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction gradually brings the output value (BW) closer to the output set value (E1) from the output control value (E2) when the control mode changes from manual to automatic.

When the output value (BW) enters the range specified by the lag band (a) on the basis of the output set value (E1), the instruction brings it closer to the output set value (E1) with a primary lag.

In manual mode (mode switching signal (e1) = 0), the instruction calculates the output value (BW), initial deviation value (Xq), and deviation (Xp) from the following expressions.

- Output value (BW) = output control value (E2)
- Initial deviation value (Xq) = output control value (E2) - output set value (E1)
- Deviation (Xp) = output control value (E2) - output set value (E1)

In automatic mode (mode switching signal (e1) = 1), the instruction calculates the output value (BW) from the following expressions.

Condition	|Xp|>a	|Xp|≤a
Xp	$Xp=Xp'-\frac{\triangle T}{T}Xq$	$Xp=\frac{T}{T+\triangle T}Xp'$
BW	BW=E1+Xp $\lvert Xp\rvert <= \frac{\triangle T}{T}\lvert Xq\rvert$ On the condition above, • BW=E1 • Xp=Xp'	BW=E1+Xp provided that $\lvert Xp\rvert \leq 10^{-4}$ • BW=E1 • Xp=Xp'

However, in automatic mode in which lag time (T)≤execution cycle (ΔT), output value (BW) = output set value (E1), deviation (Xp) = Xp'.

Operation error

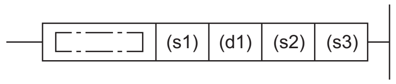
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Analog memory

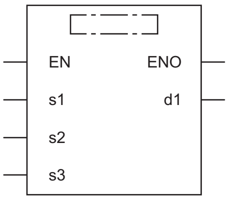
S.AMR

✕ RnCPU ✕ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

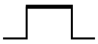
This instruction increases or decreases the output value (BW) at a fixed rate.

Ladder	ST
	ENO:=S_AMR(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.AMR	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Output addition value	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Output subtraction value	-999999 to 999999	Single-precision real number	—	User
+4 +5	E3	Output set value	-999999 to 999999	Single-precision real number	—	User
+6	e1	Operation output signal	 e1 0: Manual mode 1: Automatic mode e2 0: Do not add 1: Add e3 0: Do not subtract 1: Subtract	16-bit unsigned binary	—	User
	e2	Output addition signal				
	e3	Output subtraction signal				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

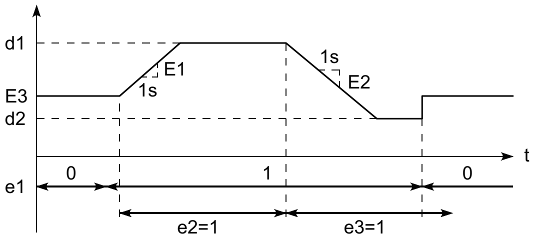
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	d1	Output upper limit value	0 to 999999	Single-precision real number	1.0	User
+2 +3	d2	Output lower limit value	-999999 to 999999	Single-precision real number	1.0	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction increases or decreases the output value (BW) at a fixed rate.



The instruction performs the following processing as specified by operation output signal (e1), output addition signal (e2), and output subtraction signal (e3).

- In manual mode (operation output signal (e1) = 0), the output value (BW) equals the output set value (E3).
- In automatic mode (operation output signal (e1) = 1), the instruction performs the following operations as specified by the output addition signal (e2) and output subtraction signal (e3).

e2	e3	BW
1	0	$BW=BW+\lvert E1\rvert\times\Delta T$ provided that when d1≤BW, BW equals d1.
0	1	$BW=BW-\lvert E2\rvert\times\Delta T$ provided that when BW≤d2, BW equals d2.
1	1	BW=BW
0	0	

Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

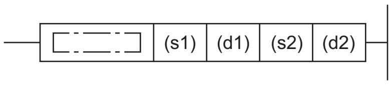
11.4 Correction Operation Instructions

Function generator

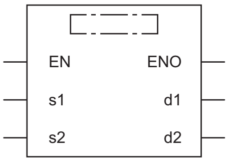
S.FG

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

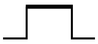
This instruction outputs the input data values following the specified function generator pattern.

Ladder	ST
	ENO:=S_FG(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.FG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	16-bit unsigned binary
(d2)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

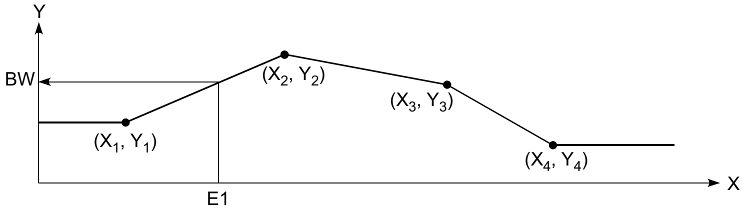
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	SN	Number of break points	0 to 48	16-bit unsigned binary	0	User

■Local work memory

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	X1	Break point coordinates	-999999 to 999999	Single-precision real number	—	User
+2 +3	Y1	Break point coordinates				
+4 +5	X2	Break point coordinates				
+6 +7	Y2	Break point coordinates				
⋮	⋮	⋮				
+4SN-4 +4SN-3	Xn	Break point coordinates				
+4SN-2 +4SN-1	Yn	Break point coordinates				

Processing details

This instruction outputs values according to the function generator pattern consisting of n break points as specified by (s2) with regard to the input value (E1).



The S.FG instruction performs the following operations.

Condition	Output value (BW)
$E1 \leq X1$	$BW=Y1$
$X_{i-1}<E1 \leq X_i(i=2 \text{ to } n)$	$BW=\frac{Y_i-Y_{i-1}}{X_i-X_{i-1}} \times (E1-X_{i-1})+Y_{i-1}$
$X_n<E1$	$BW=Y_n$

- If the value specified in (n) is 0, no processing is performed.
- If $X_{i-1} > X_i$, processing is stopped when n = i - 1. (The subsequent data is ignored.)
- When there are two or more X_i for the same Y_i, the Y with smaller i is selected.

Operation error

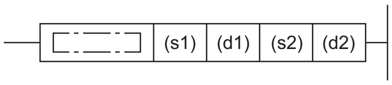
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of break points (SN) is less than 0 or greater than 48.

Inverse function generator

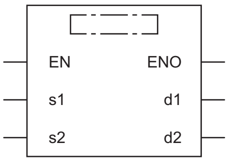
S.IFG

RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

This instruction outputs the input data values following the specified inverse function generator pattern.

Ladder	ST
	ENO:=S_IFG(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.IFG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	16-bit unsigned binary
(d2)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

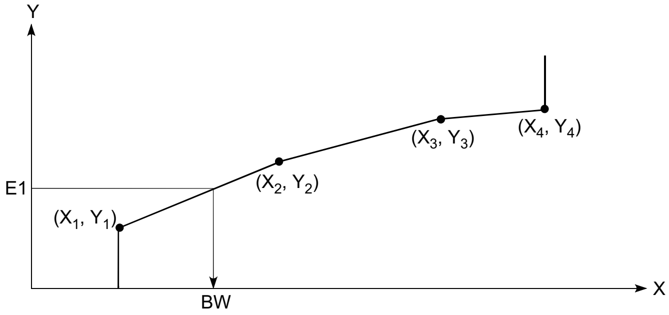
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	SN	Number of break points	0 to 48	16-bit unsigned binary	0	User

■Local work memory

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	X1	Break point coordinates	-999999 to 999999	Single-precision real number	—	User
+2 +3	Y1	Break point coordinates				
+4 +5	X2	Break point coordinates				
+6 +7	Y2	Break point coordinates				
⋮	⋮	⋮				
+4SN-4 +4SN-3	Xn	Break point coordinates				
+4SN-2 +4SN-1	Yn	Break point coordinates				

Processing details

This instruction outputs values according to the inverse function generator pattern consisting of n break points as specified by (s2) with regard to the input value (E1).



The S.IFG instruction performs the following operations.

Condition	Output value (BW)
E1≤Y1	BW=X1
$Y_{i-1}<E1\leq Y_i$(i=2 to n)	$BW=\frac{X_i-X_{i-1}}{Y_i-Y_{i-1}}\times(E1-Y_{i-1})+X_{i-1}$
$Y_n<E1$	$BW=X_n$

- If the value specified in (n) is 0, no processing is performed.
- If $X_{i-1} > X_i$, processing is stopped when n = i - 1. (The subsequent data is ignored.)
- When there are two or more X_i for the same Y_i, the X with smaller i is selected.

Operation error

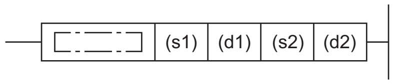
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of break points (SN) is less than 0 or greater than 48.

Standard filter

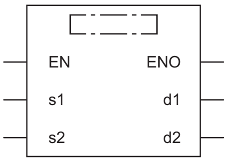
S.FLT

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This function outputs the mean value of the n pieces of data sampled at the specified data collection intervals (ST).

Ladder	ST
	ENO:=S_FLT(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.FLT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(d2)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

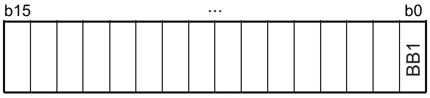
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	-999999 to 999999	Single-precision real number	—	System
+2	BB	BB1	Data sufficiency bit	 0: Data sufficient 1: Data insufficient	Unsigned 16-bit binary data	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	ST	Data sampling interval	0 to 999999 [s] Set a value within the following range. $\frac{ST}{\triangle T}$<=32767	Single-precision real number	1.0	User
+2	SN	Sampling count	0 to 48	16-bit unsigned binary	0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

Operand: (d2)						
Position	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	ST'	Last data sampling interval	—	Single-precision real number	—	System
+2	SN'	Last sampling count		16-bit unsigned binary		System
+3	i	Cycle counter[1]		16-bit unsigned binary		System
+4	n1	Number of data blocks stored		16-bit unsigned binary		System
+5	n2	Storage address		16-bit unsigned binary		System
+6 +7	—	—	—	—	—	—
+8 +9	1	Dead time table 1	—	Single-precision real number	—	System
+10 +11	2	Dead time table 2				
⋮	⋮	⋮				
+2SN+6 +2SN+7	SN	Dead time table SN				

[1] The cycle counter value is rounded off to the nearest whole number.

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction samples input values (E1) at data sampling intervals (ST) by the number of samples (SN), stores them in the dead time table, averages the number of collected data blocks (SN), and outputs the result.

- The data refreshing cycle is "data sampling interval (ST)÷execution cycle (ΔT)". (The resultant value is rounded off to the nearest whole number.)
- When the dead time table is filled with the number of sampled data blocks (SN), the data sufficiency bit (BB1) is set to 0. When the dead time table does not have sufficient data, BB1 is set to 1.

Point

- Until the dead time table is filled with data, the past data is averaged and output.
- Processing is performed by ST=n×ΔT (n is an integer).

Operation error

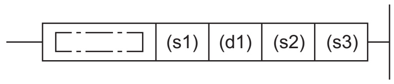
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (s2), or (d2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The sampling count (SN) is less than 0 or greater than 48.
	The execution cycle (ΔT) setting is less than 0.
	The data sampling interval (ST) is less than 0.
	(Data sampling interval (ST)÷execution cycle (ΔT)) is greater than 32767.

Integration

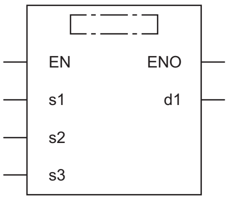
S.SUM

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction integrates and outputs the input data.

Ladder	ST
	ENO:=S_SUM(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SUM	⎍

Setting data

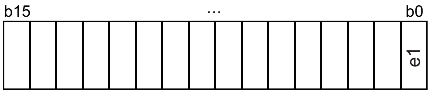
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1		Input value	-999999 to 999999	Single-precision real number	—	User
+2	e	e1	Integration start signal	 0: Integration operation not performed 1: Integration operation not performed	16-bit unsigned binary	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	ILC	Input low-cut value	-999999 to 999999	Single-precision real number	0.0	User
+2 +3	A	Initial value	-999999 to 999999	Single-precision real number	0.0	User
+4	RANGE	Input range	1: /second 2: /minute 3: /hour	16-bit unsigned binary	1	User

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

When the integration start signal (e1) changes from 0 to 1, this instruction integrates the input values (E1) and outputs the result.

The S.SUM instruction performs the following operations.

e1	E1	Output (BW)
0	—	Outputs the initial value (A) of the operation constant.
1	E1≤ILC	The last value is output as is.
	E1>ILC	$BW=E1\times\frac{\triangle T}{T}$+Last value

The value of T used for operation varies depending on the setting of the input range (RANGE).

- Input range (RANGE) = 1: T = 1
- Input range (RANGE) = 2: T = 60
- Input range (RANGE) = 3: T = 3600

Operation error

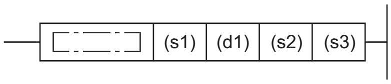
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1), (d1), or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Input range (RANGE) is less than 1 or greater than 3.

Temperature/pressure correction

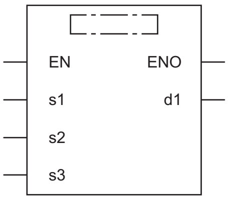
S.TPC

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

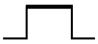
This instruction outputs input data after temperature/pressure correction.

Ladder	ST
	ENO:=S_TPC(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.TPC	

Setting data

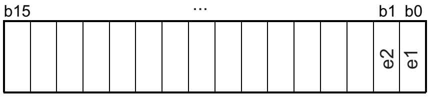
■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1		Differential pressure	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2		Measured temperature	-999999 to 999999	Single-precision real number	—	User
+4 +5	E3		Measured pressure	-999999 to 999999	Single-precision real number	—	User
+6	e	e1	E2 use flag	 0: Not used 1: Used	16-bit unsigned binary	—	User
		e2	E3 use flag				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	TEMP	Designed temperature T' (engineering value)	-999999 to 999999 [℃]	Single-precision real number	0.0	User
+2 +3	B1	Bias (temperature)	-999999 to 999999 [℃]	Single-precision real number	273.15	User
+4 +5	PRES	Designed pressure P' (engineering value)	-999999 to 999999	Single-precision real number	0.0	User
+6 +7	B2	Bias (pressure)	-999999 to 999999	Single-precision real number	10332.0	User

Processing details

This instruction outputs the input value (E1) after performing temperature or pressure correction over it.

The S.TPC instruction calculates the temperature/pressure correction value using the following expression.

$BW=E1\times A1\times A2$

The values of A1 and A2 are calculated by the following expressions.

Input		**A1**	**A2**
e1	**e2**		
Used	Used	$\frac{T'+B1}{E2+B1}$	$\frac{E3+B2}{P'+B2}$
Not used	Used	1.0	$\frac{E3+B2}{P'+B2}$
Used	Not used	$\frac{T'+B1}{E2+B1}$	1.0
Not used	Not used	1.0	1.0

Operation error

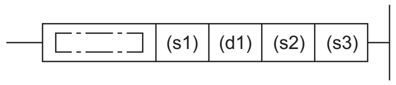
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Engineering value transformation

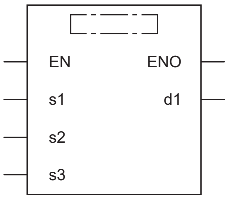
S.ENG

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

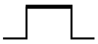
This instruction performs engineering value inverse transformation of the input data.

Ladder	ST
	ENO:=S_ENG(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ENG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User

■Block memory

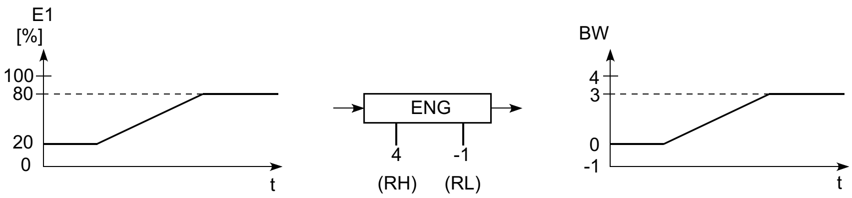
Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction performs engineering value transformation of the input data (E1) and outputs it.



The S.ENG instruction performs the following operations.

$$BW=\frac{RH-RL}{100}\times E1+RL \quad (E1=0\ \text{to}\ 100\%)$$

Operation error

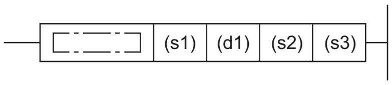
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Engineering value inverse transformation

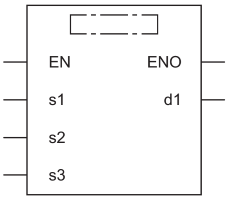
S.IENG

✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

The instruction performs engineering value inverse transformation processing to the input data.

Ladder	ST
	ENO:=S_IENG(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.IENG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

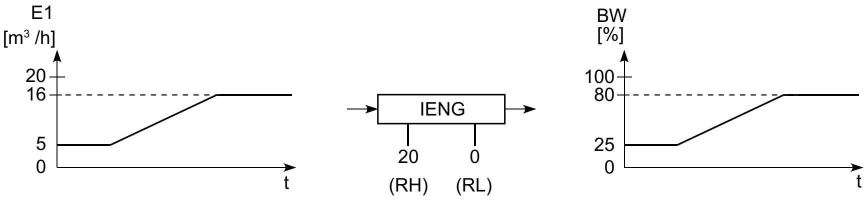
Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999 [%]	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	100.0	User
+2 +3	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction converts the input value (E1) to the % value and outputs it.



The S.IENG instruction performs the following operations.

$$BW=\frac{100}{RH-RL}\times(E1-RL)\ [\%]$$

Set the engineering value upper limit (RH) and lower limit (RL) so that RH is greater than RL.

Even when RH equals or less than RL, processing is executed accordingly but does not result in engineering value inverse transformation.

When RH equals RL, the output value (BW) becomes 0.

Operation error

Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

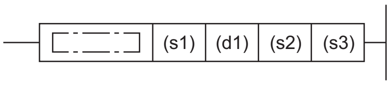
11.5 Arithmetic Operation Instructions

Addition

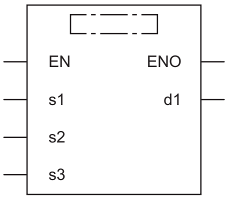
S.ADD

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction adds input data with a coefficient.

Ladder	ST
	ENO:=S_ADD(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ADD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	0 to 5	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of coefficients	0 to 5	16-bit unsigned binary	0	User
+1 +2	K1	Coefficient 1	-999999 to 999999	Single-precision real number	1.0	User
+3 +4	K2	Coefficient 2				
⋮	⋮	⋮				
+2n-1 +2n	Kn	Coefficient n				
+2n+1 +2n+2	B	Bias	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction adds the data of input values (E1 to En) with a coefficient.

The S.ADD instruction performs the following operations.

$BW=(K1\times E1)+(K2\times E2)...+(Kn\times En)+B$

When the number of inputs (n) is 0, the output value (BW) becomes bias (B).

Operation error

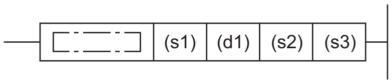
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of inputs (n) is less than 0 or greater than 5.
	The number of coefficients (n) is less than 0 or greater than 5.

Subtraction

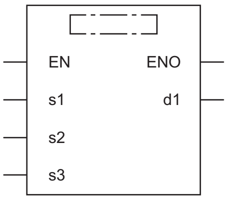
S.SUB

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

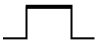
This instruction subtracts input data with a coefficient.

Ladder	ST
	ENO:=S_SUB(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SUB	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	0 to 5	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of coefficients	0 to 5	16-bit unsigned binary	0	User
+1 +2	K1	Coefficient 1	-999999 to 999999	Single-precision real number	1.0	User
+3 +4	K2	Coefficient 2				
⋮	⋮	⋮				
+2n-1 +2n	Kn	Coefficient n				
+2n+1 +2n+2	B	Bias	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction performs subtraction of the data of input values (E1 to En) with a coefficient.

The S.SUB instruction performs the following operations.

$BW=(K1\times E1)-(K2\times E2)...-(Kn\times En)+B$

When the number of inputs (n) is 0, the output value (BW) becomes bias (B).

Operation error

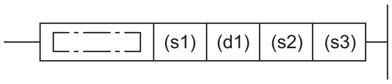
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of inputs (n) is less than 0 or greater than 5.
	The number of coefficients (n) is less than 0 or greater than 5.

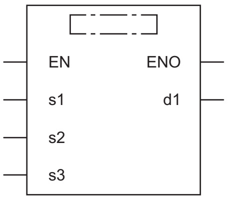
Multiplication

S.MUL

✕RnCPU ✕RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

This instruction multiplies input data with a coefficient.

Ladder	ST
	ENO:=S_MUL(EN,s1,s2,s3,d1);

FBD/LD


■Execution condition

Instruction	Execution condition
S.MUL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of inputs	0 to 5	16-bit unsigned binary	—	User
+1 +2	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+3 +4	E2	Input value 2				
⋮	⋮	⋮				
+2n-1 +2n	En	Input value n				

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	n	Number of coefficients	0 to 5	16-bit unsigned binary	0	User
+1 +2	K1	Coefficient 1	-999999 to 999999	Single-precision real number	1.0	User
+3 +4	K2	Coefficient 2				
⋮	⋮	⋮				
+2n-1 +2n	Kn	Coefficient n				
+2n+1 +2n+2	B	Bias	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction performs multiplication of the data of input values (E1 to En) with a coefficient.

The S.MUL instruction performs the following operations.

$BW=(K1\times E1)\times(K2\times E2)...\times(Kn\times En)+B$

When the number of inputs (n) is 0, the output value (BW) becomes bias (B).

Operation error

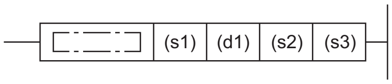
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	The number of inputs (n) is less than 0 or greater than 5.
	The number of coefficients (n) is less than 0 or greater than 5.

Division

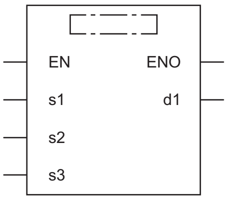
S.DIV

RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

This instruction performs division of the input data with a coefficient.

Ladder	ST
	ENO:=S_DIV(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DIV	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value 1 (numerator)	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2 (denominator)	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	-999999 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	A	Coefficient 1	-999999 to 999999	Single-precision real number	1.0	User
+2 +3	K1	Coefficient 2	-999999 to 999999	Single-precision real number	1.0	User
+4 +5	K2	Coefficient 3	-999999 to 999999	Single-precision real number	1.0	User
+6 +7	B1	Bias 1	-999999 to 999999	Single-precision real number	0.0	User
+8 +9	B2	Bias 2	-999999 to 999999	Single-precision real number	0.0	User
+10 +11	B3	Bias 3	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction divides the input value 1 (E1) by the input value 2 (E2).

The S.DIV instruction performs the following operations.

$$BW=A\times\frac{K1\times E1+B1}{K2\times E2+B2}+B3$$

When the denominator (efficient 2 (K2)×input value 2 (numerator) (E2) + bias 2 (B2)) is 0, the output value (BW) becomes bias 3 (B3).

Operation error

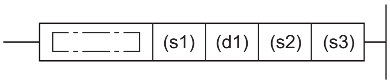
Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

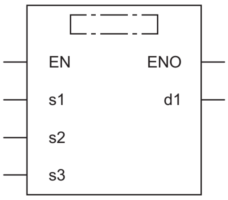
Square root

S.SQR

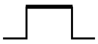
✗ RnCPU ✗ RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

This instruction outputs the square root (√) of input data.

Ladder	ST
	ENO:=S_SQR(EN,s1,s2,s3,d1);

FBD/LD


■Execution condition

Instruction	Execution condition
S.SQR	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Single-precision real number
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	0 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW	Output value	0 to 999999	Single-precision real number	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	OLC	Output low-cut value	0 to 999999	Single-precision real number	0.0	User
+2 +3	K	Coefficient	0 to 999999	Single-precision real number	1.0	User

Processing details

This instruction outputs √ of the input value (E1). When the input value (E1) is less than 0, 0 is output.

The S.SQR instruction performs the following operations.

$\text{BW} = \text{K} \times \sqrt{(\text{E1})}$

In the following case, however, the output value (BW) becomes 0.

$\text{Coefficient (K)} \times \sqrt{\text{(Input value (E1))}} <= \text{Output low-cut value (OLC)}$

Operation error

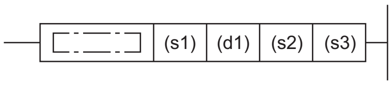
Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

Absolute value

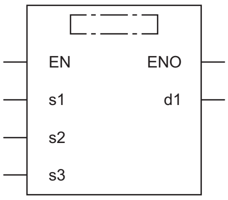
S.ABS

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

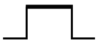
This instruction outputs the absolute value of input data.

Ladder	ST
	ENO:=S_ABS(EN,s1,s2,s3,d1);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ABS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Single-precision real number
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

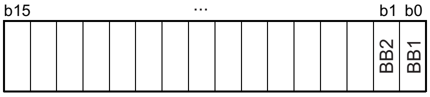
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

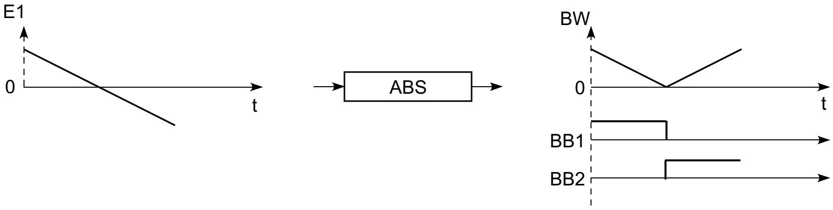
Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	0 to 999999	Single-precision real number	—	System
+2	BB	BB1	Sign determination of Input value (E1)	 E1 > 0: BB1 = 1 E1 < 0: BB2 = 1 E1 = 0: BB1 = BB2 = 0	16-bit unsigned binary	—	System
		BB2					

Processing details

This instruction outputs the absolute value of the input value (E1).



The S.ABS instruction performs the following operations.

BW=|E1|

The instruction determines the sign of the input value (E1) and outputs the result to the sign determination bits (BB1 and BB2) of the input value (E1).

E1 status	BB1	BB2
E1>0	1	0
E1 < 0	0	1
E1=0	0	0

Operation error

Error code	Error content
3402H	(s1) is a subnormal number or NaN (not a number).

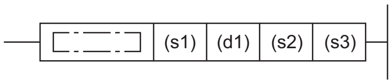
11.6 Comparison Operation Instructions

Comparing data

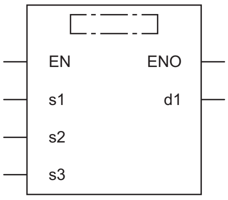
S.>

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction compares input data, and outputs the comparison result.

Ladder	ST
	ENO:=S_GT(EN,s1,s2,s3,d1);

FBD/LD



(□ is to be replaced by S_GT.)

■Execution condition

Instruction	Execution condition
S.>	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

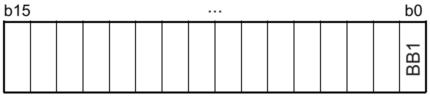
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	The same value as the input value 1 (E1) is stored.	Single-precision real number	—	System
+2	BB	BB1	Comparison output	 The comparison result of E1 and E2 is stored.	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	K	Set value	-999999 to 999999	Single-precision real number	0.0	User
+2 +3	HS	Hysteresis	0 to 999999	Single-precision real number	0.0	User

Processing details

This instruction compares input values 1 (E1) and 2 (E2) and outputs the result (d1) to the comparison output (BB1).

Condition	BB1
E1>(E2+K)	1
E1≤(E2+K-HS)	0
(E2+K-HS)<E1≤(E2+K)	Last value is output.

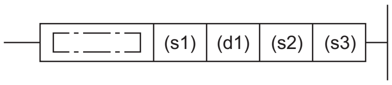
Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Hysteresis (HS) is less than 0.

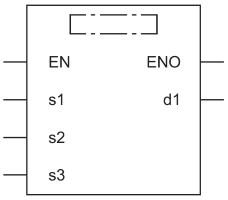
S.<

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction compares input data, and outputs the comparison result.

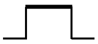
Ladder	ST
	ENO:=S_LT(EN,s1,s2,s3,d1);

FBD/LD



(□ is to be replaced by S_LT.)

■Execution condition

Instruction	Execution condition
S.<	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

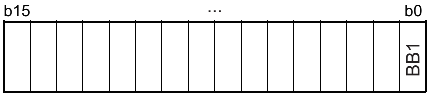
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	The same value as the input value 1 (E1) is stored.	Single-precision real number	—	System
+2	BB	BB1	Comparison output	 The comparison result of E1 and E2 is stored.	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	K	Set value	-999999 to 999999	Single-precision real number	0.0	User
+2 +3	HS	Hysteresis	0 to 999999	Single-precision real number	0.0	User

Processing details

This instruction compares input values 1 (E1) and 2 (E2) and outputs the result (d1) to the comparison output (BB1).

Condition	BB1
E1<(E2+K)	1
E1≥(E2+K+HS)	0
(E2+K)≤E1<(E2+K+HS)	Last value is output.

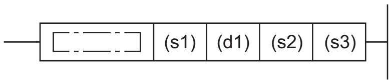
Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Hysteresis (HS) is less than 0.

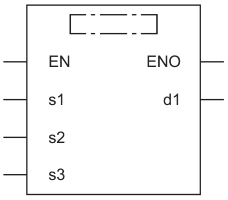
S.=

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction compares input data, and outputs the comparison result.

Ladder	ST
	ENO:=S_EQ(EN,s1,s2,s3,d1);

FBD/LD



(□ is to be replaced by S_EQ.)

■Execution condition

Instruction	Execution condition
S.=	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Single-precision real number
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

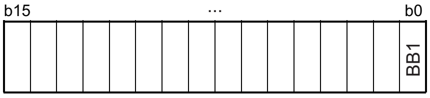
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)

Device	Symbol		Name	Recommended range	Data type	Standard value	Set by
+0 +1	BW		Output value	The same value as the input value 1 (E1) is stored.	Single-precision real number	—	System
+2	BB	BB1	Comparison output	 The comparison result of E1 and E2 is stored.	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	K	Set value	-999999 to 999999	Single-precision real number	0.0	User

Processing details

This instruction compares input values 1 (E1) and 2 (E2) and outputs the result (d1) to the comparison output (BB1).

Condition	BB1
E1=(E2+K)	1
E1≠(E2+K)	0

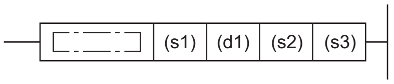
Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.

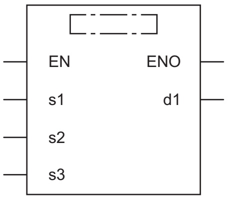
S.>=

~~RnCPU~~ ~~RnENCPU~~ RnPCPU (Process) RnPCPU (Redundant) ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

This instruction compares input data, and outputs the comparison result.

Ladder	ST
	ENO:=S_GE(EN,s1,s2,s3,d1);

FBD/LD



(□ is to be replaced by S_GE.)

■Execution condition

Instruction	Execution condition
S.>=	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

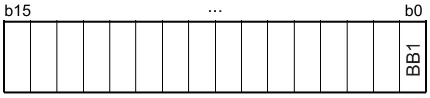
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	The same value as the input value 1 (E1) is stored.	Single-precision real number	—	System
+2	BB	BB1	Comparison output	 The comparison result of E1 and E2 is stored.	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	K	Set value	-999999 to 999999	Single-precision real number	0.0	User
+2 +3	HS	Hysteresis	0 to 999999	Single-precision real number	0.0	User

Processing details

This instruction compares input values 1 (E1) and 2 (E2) and outputs the result (d1) to the comparison output (BB1).

Condition	BB1
E1≥(E2+K)	1
E1<(E2+K-HS)	0
(E2+K-HS)≤E1<(E2+K)	Last value is output.

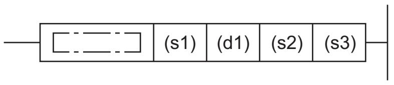
Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Hysteresis (HS) is less than 0.

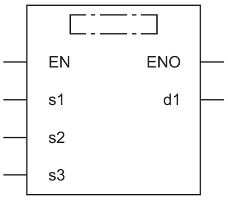
S.<=

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction compares input data, and outputs the comparison result.

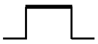
Ladder	ST
	ENO:=S_LE(EN,s1,s2,s3,d1);

FBD/LD



(□ is to be replaced by S_LE.)

■Execution condition

Instruction	Execution condition
S.<=	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	Word
(s2)	Operation constant start device	Refer to "Operation constant".	Word
(s3)	Empty string specification (Specify "" in ladder. Specify '' in ST or FBD/LD.)	—	String
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

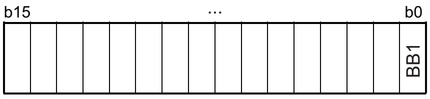
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	—	—	—	—	—	—	—	—	○	—

■Input data

Operand: (s1)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0 +1	E1	Input value 1	-999999 to 999999	Single-precision real number	—	User
+2 +3	E2	Input value 2	-999999 to 999999	Single-precision real number	—	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	BW		Output value	The same value as the input value 1 (E1) is stored.	Single-precision real number	—	System
+2	BB	BB1	Comparison output	 The comparison result of E1 and E2 is stored.	16-bit unsigned binary	—	System

■Operation constant

Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	K	Set value	-999999 to 999999	Single-precision real number	0.0	User
+2 +3	HS	Hysteresis	0 to 999999	Single-precision real number	0.0	User

Processing details

This instruction compares input values 1 (E1) and 2 (E2) and outputs the result (d1) to the comparison output (BB1).

Condition	BB1
E1≤(E2+K)	1
E1>(E2+K+HS)	0
(E2+K)<E1≤(E2+K+HS)	Last value is output.

Operation error

Error code	Error content
3402H	The value specified by (s1) or (s2) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	Hysteresis (HS) is less than 0.

11.7 Auto Tuning

Auto tuning is designed to make initial setting of PID constants.
Auto tuning can be used for processes that can be approximated with the "primary lag plus dead time" represented by the following expression.

Ex.
Process with relatively slow response such as temperature control

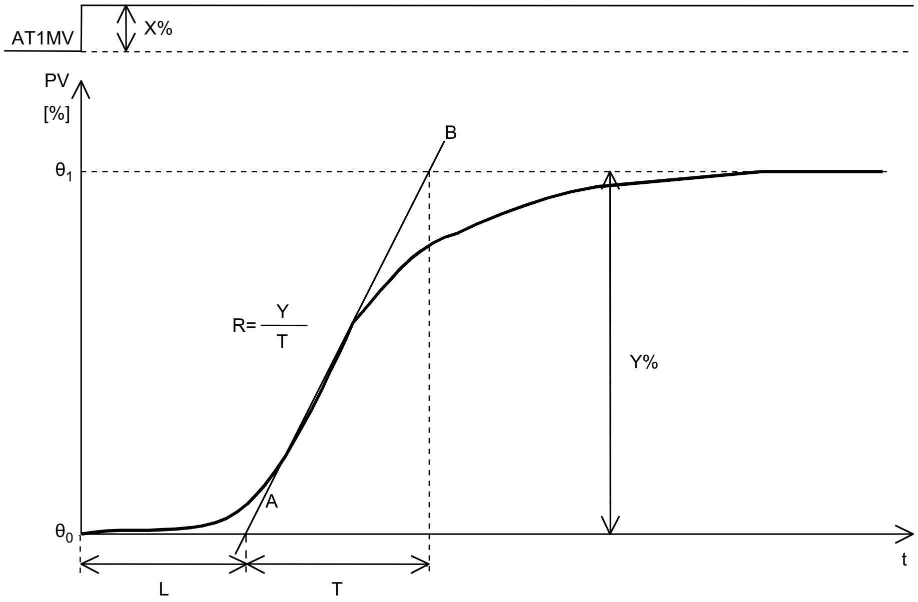
$$\frac{K}{1+Ts}e^{-Ls}$$

K: Gain, T: Time constant, L: Dead time, s: Laplace operator
Auto tuning can be used for the loop that uses S.PID or S.2PID instruction.
Auto tuning is performed in the ZN method: step response method of Ziegler and Nichols.

Outline of step response method

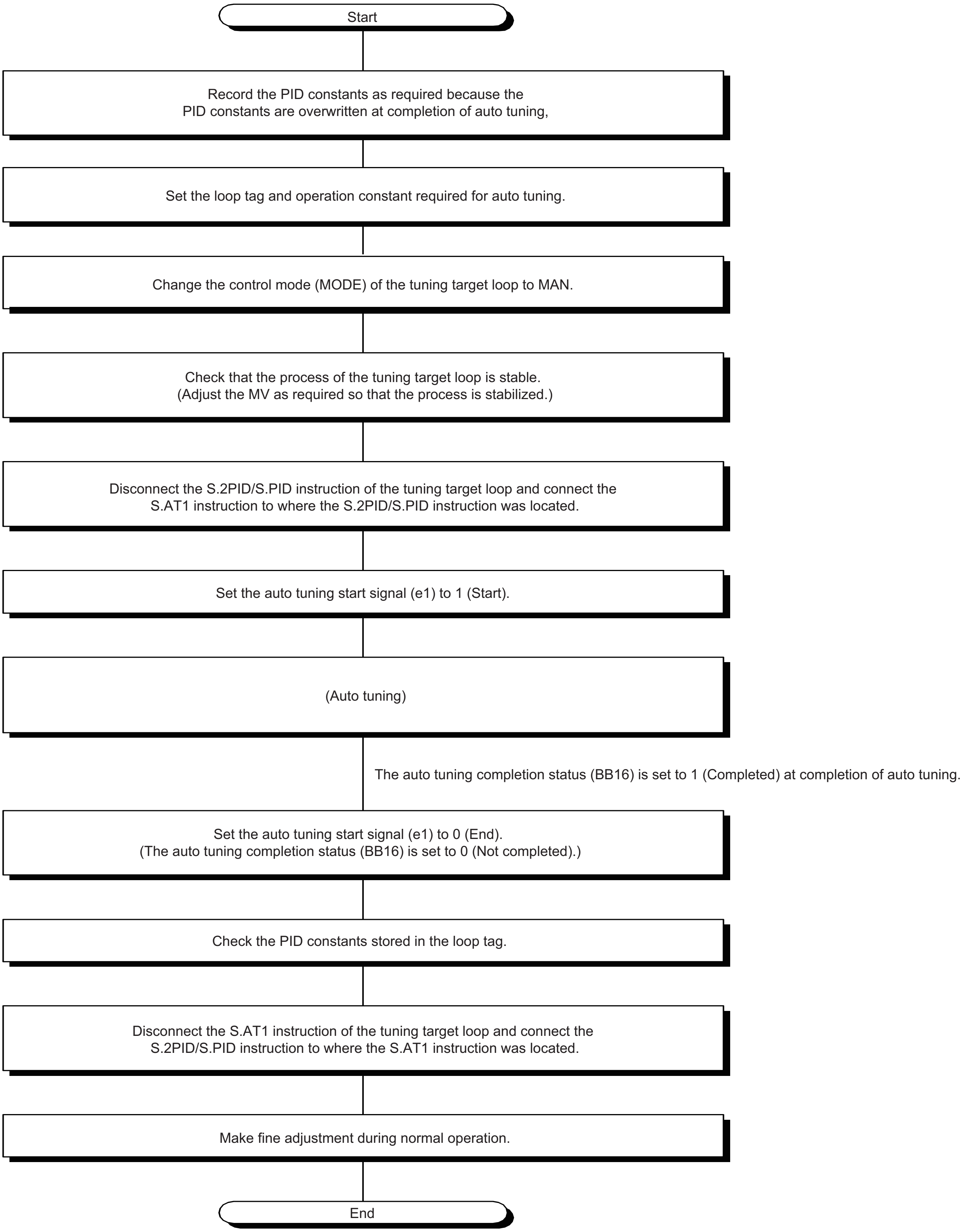
With no control operation being performed, change the manipulated value (MV) in a stepwise manner and look how the process value (PV) varies.

- When MV is changed in a stepwise manner, PV begins to change slowly. Soon, the change speed becomes faster and then becomes slow again, and finally is settled as a fixed value.
- Draw a tangent line at the place where PV varies fastest, and find the points of intersection A and B where the tangent line crosses the horizontal axis corresponding to the first process value (θ_0) and last process value (θ_1). This obtains the equivalent dead time (L) and equivalent time constant (T) as shown below.
- Determine maximum ramp (response speed) R = Y/T from the equivalent time constant (T) and maximum process value width (Y). Apply the equivalent dead time (L) and maximum ramp (R) to the Ziegler and Nichols' adjustment rule to calculate the proportional gain KP(P), integral constant TI(I), and derivative constant TD(D).



Auto tuning procedure

The following shows the auto tuning procedure.



Start

Record the PID constants as required because the PID constants are overwritten at completion of auto tuning,

Set the loop tag and operation constant required for auto tuning.

Change the control mode (MODE) of the tuning target loop to MAN.

Check that the process of the tuning target loop is stable.
(Adjust the MV as required so that the process is stabilized.)

Disconnect the S.2PID/S.PID instruction of the tuning target loop and connect the S.AT1 instruction to where the S.2PID/S.PID instruction was located.

Set the auto tuning start signal (e1) to 1 (Start).

(Auto tuning)

The auto tuning completion status (BB16) is set to 1 (Completed) at completion of auto tuning.

Set the auto tuning start signal (e1) to 0 (End).
(The auto tuning completion status (BB16) is set to 0 (Not completed).)

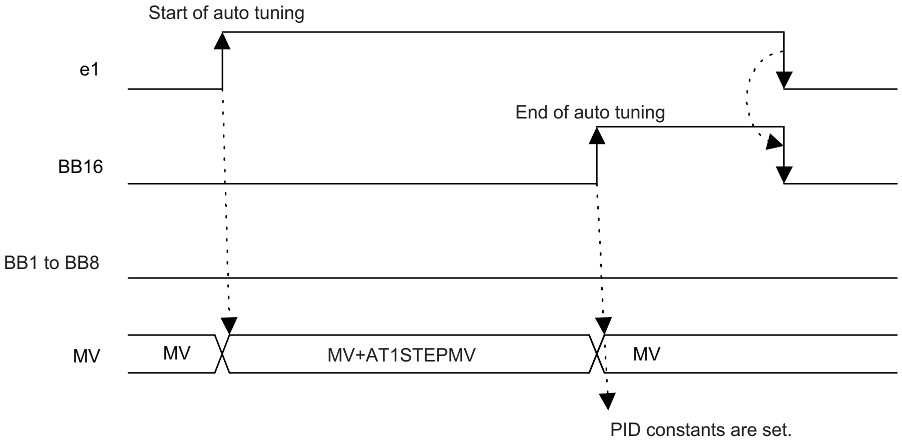
Check the PID constants stored in the loop tag.

Disconnect the S.AT1 instruction of the tuning target loop and connect the S.2PID/S.PID instruction to where the S.AT1 instruction was located.

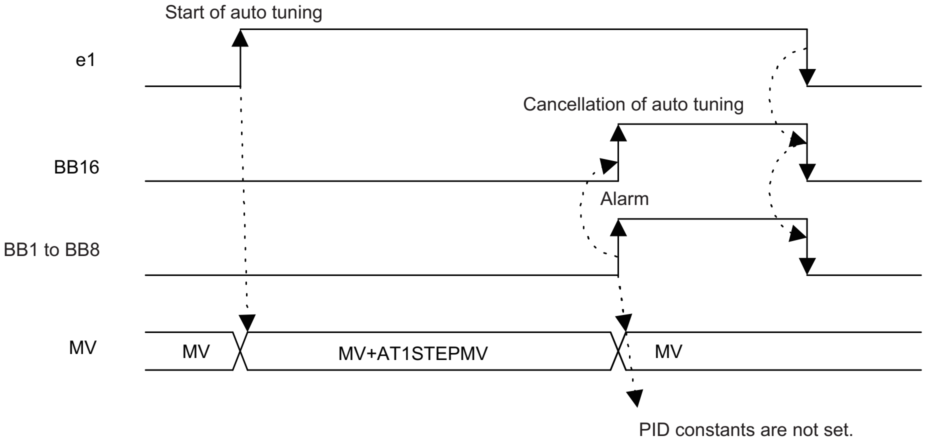
Make fine adjustment during normal operation.

End

- Time chart from auto tuning start till normal completion



- Time chart from auto tuning start till stop due to alarm occurrence

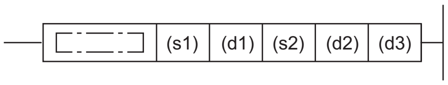


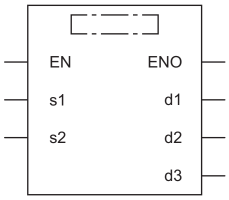
Auto tuning instructions

S.AT1

✗RnCPU ✗RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

This instruction performs auto tuning to make initial setting of PID constants.

Ladder	ST
	ENO:=S_AT1(EN,s1,s2,d1,d2,d3);

FBD/LD


■Execution condition

Instruction	Execution condition
S.AT1	

Setting data

■Description, range, data type

Operand	Description	Range	Data type
(s1)	Input data start device	Refer to "Input data".	Word
(d1)	Block memory start device	Refer to "Block memory".	16-bit unsigned binary
(s2)	Operation constant start device	Refer to "Operation constant".	16-bit unsigned binary
(d2)	Loop tag memory start device	Refer to "Loop tag memory".	Word
(d3)	Local work memory start device	Refer to "Local work memory".	Word
EN	Execution condition	—	Bit
ENO	Execution result	—	Bit

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	—	—	○	—	—	—	—	○	—	—	—	—

■Input data

Operand: (s1)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0 +1	E1	Input value	-999999 to 999999 [%]	Single-precision real number	—	User
+2	e1	Auto tuning start signal	b15 … b0 e1 (b0) 0: Stop/End 1: Start	16-bit unsigned binary	0	User

■Block memory

Operand: (d1)							
Device	**Symbol**		**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	BB	BB1	Alarm	b15 … b7 b6 b5 b4 b3 b2 b1 b0 BB16 … BB8 BB7 BB6 BB5 BB4 BB3 BB2 BB1 BB16 0: Not completed 1: Completed BB1 to BB8 0: No alarm 1: Alarm	16-bit unsigned binary	—	System
		BB2	Input upper limit alarm				
		BB3	Input lower limit alarm				
		BB4	Output upper limit alarm				
		BB5	Output lower limit alarm				
		BB6	Timeout alarm				
		BB7	Control mode alarm				
		BB8	Identification alarm				
		BB16	Auto tuning completion				

■Operation constant

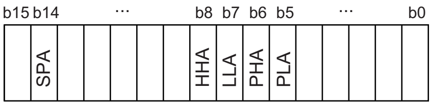
Operand: (s2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+0	PN	Operation mode	0: Reverse action 1: Direct action	16-bit unsigned binary	0	User

■Loop tag memory

The loop tag memory occupies 96 words from the specified start device.

Operand: (d2)						
Device	**Symbol**	**Name**	**Recommended range**	**Data type**	**Standard value**	**Set by**
+1	MODE	Control mode	0 to FFFFH b15 … b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CSV CMV CCB CAB CMB CAS AUT MAN LCC LCA LCM (b10 to b0)	16-bit unsigned binary	8H	User/ system

Operand: (d2)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+3	ALM	Alarm detection	0 to FFFFH  SPA 0: Loop RUN 1: Loop STOP HHA, LLA, PHA, PLA 0: No alarm 1: Alarm	16-bit unsigned binary	4000H	User/ system
+12 +13	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	0.0	User/ system
+18 +19	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	100.0	User
+20 +21	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	0.0	User
+52 +53	P	Gain	0 to 999999	Single-precision real number	1.0	User/ system
+54 +55	I	Integral constant	0 to 999999 [s]	Single-precision real number	10.0	User/ system
+56 +57	D	Derivative constant	0 to 999999 [s]	Single-precision real number	0.0	User/ system
+70 +71	AT1STEP MV	AT1 step manipulated value	-100 to 100 [%]	Single-precision real number	0.0	User
+72 +73	AT1ST	AT1 sampling cycle	0 to 999999 [s] Set a value within the following range. $\frac{\text{AT1ST}}{\triangle\text{T}}<=32767$	Single-precision real number	1.0	User
+74 +75	AT1TOUT 1	AT1 timeout time	0 to 999999 [s] Set a value within the following range. $\frac{\text{AT1TOUT1}}{\triangle\text{T}}<=32767$	Single-precision real number	100.0	User
+76 +77	AT1TOUT 2	Timeout time after maximum AT1 ramp	0 to 999999 [s] Set a value within the following range. $\frac{\text{AT1TOUT2}}{\triangle\text{T}}<=32767$	Single-precision real number	10.0	User

■Local work memory

The system uses this area as a work area.

To start the control from the initial status, clear data by using a sequence program.

Operand: (d3)

Device	Symbol	Name	Recommended range	Data type	Standard value	Set by
+0	—	Sampling period counter initialization completion flag	—	—	—	System
+1		Sampling period counter[*1]				
+2		Timeout time counter initialization completion flag				
+3		Timeout time counter[*1]				
+4		Timeout time (after maximum ramp) counter initialization completion flag				
+5		Timeout time (after maximum ramp) counter[*1]				
+6		Step manipulated value set completion flag				
+7		Auto tuning counter				
+8 +9		Auto tuning start PV0				
+10 +11	PV_{n-1}	Last process value				
+12 +13	—	Maximum ramp value				
+14 +15		Counter at maximum ramp				
+16 +17		PV at maximum ramp				
+18 +19	R	Response speed				
+20 +21	L	Equivalent dead time				

[*1] The sampling period counter, timeout time counter, and timeout (after maximum ramp) counter are each rounded off to the nearest whole number.

■Execution cycle (ΔT)

Set the execution cycle (ΔT) in SD816 and SD817 in single-precision real number.

Processing details

This instruction performs auto tuning to make initial setting of PID constants.

■Start signal determination processing

The instruction performs the following processing according to the auto tuning start signal (e1) and auto tuning completion (BB16).

e1	BB16	Processing
0	0	• BB alarms from BB1 to identification alarm BB8 are set to 0. • When the step manipulated value set completion flag is 1, the following processing is performed. MV=MV-AT1STEPMV • The S.AT1 instruction is terminated.
1	0	• "Loop stop processing" is performed.
0	1	• BB auto tuning completion (BB16) is set to 0. • The S.AT1 instruction is terminated.
1	1	• The S.AT1 instruction is terminated.

■Loop stop processing

The following processing is performed according to the SPA status of the alarm detection (ALM).

SPA status	Processing details
1	The loop stops. When the loop stops, the following operations are performed and the S.AT1 instruction ends. • Auto tuning completion (BB16) is set to 1. • When the step manipulated value set completion flag is 1, the following processing is performed. MV=MV-AT1STEPMV
0	The loop runs, and "mode determination processing" is performed.

■Mode determination processing

The following processing is performed depending on the control mode (MODE) setting.

Control mode (MODE) setting	Processing details
AUT, CAB, CAS, CCB, CSV, LCA, LCC	The following processing is performed and the S.AT1 instruction is terminated. • The control mode alarm (BB7) is set to 1. • Auto tuning completion (BB16) is set to 1. • When the step manipulated value set completion flag is 1, the following processing is performed. MV=MV-AT1STEPMV
MAN, CMB, CMV, LCM	"Input check processing" is performed.

■Input check processing

The following processing is performed according to the alarm detection (ALM).

Alarm Detection (ALM)	Processing details
PHA or HHA is 1.	The following processing is performed and the S.AT1 instruction is terminated. • The input upper limit alarm (BB2) is set to 1. • Auto tuning completion (BB16) is set to 1.
PLA or LLA is 1.	The following processing is performed and the S.AT1 instruction is terminated. • The input upper limit alarm (BB3) is set to 1. • Auto tuning completion (BB16) is set to 1.

■Timeout determination processing

This function determines whether the auto tuning processing has reached the AT1 timeout time (AT1TOUT1).

Auto tuning processing	Processing details
If the AT1 timeout time (AT1TOUT1) is reached	The following processing is performed and the S.AT1 instruction is terminated. • The input upper limit alarm (BB6) is set to 1. • Auto tuning completion (BB16) is set to 1.
If the AT1 timeout time (AT1TOUT1) has not been reached	"Timeout (after maximum ramp) determination processing" is performed.

■Timeout (after maximum ramp) determination processing

This function determines whether the auto tuning processing has reached the AT1 timeout time after maximum ramp (AT1TOUT2).

However, if the timeout time (after maximum ramp) counter initialization completion flag is 0, "step manipulated value set processing" is performed.

Auto tuning processing	Processing details
If the AT1 timeout time after maximum ramp (AT1TOUT2) is reached	"Identification processing" is performed.
If the AT1 timeout time after maximum ramp (AT1TOUT2) has not been reached	"Step manipulated value set processing" is performed.

■Step manipulated value set processing

This function checks the step manipulated value set completion flag to determine whether the step manipulated value has been set (1) or not set (0).

- When the step manipulated value set completion flag is 0, the following processing is performed.

The AT1 step manipulated value (AT1SETPMV) is added to the manipulated value (MV).

T1=MV+AT1STEPMV

The upper/lower limiter function performs the following operations, and outputs the result to the output upper limit alarm (BB4) and output lower limit alarm (BB5).

Condition	Result				
	BB4	BB5	BB16	MV	Processing after upper/lower limiter
T1 > MH	1	0	1	Original MV remains unchanged.	The S.AT1 instruction is terminated.
T1 < ML	0	1	1		
ML ≤ T1 ≤ MH	0	0	0	T1	The following processing is performed. • The step manipulated value set completion flag is set to 1. • The auto tuning counter is set to 0. • The input value (E1) is stored in auto tuning start PV0. • The input value (E1) is stored in the last measurement value (PV_{n-1}). • The maximum slope value, maximum ramp counter, maximum ramp PV, response speed (R), and equivalent dead time (L) are set to 0.

- When the step manipulated value set completion flag is 1, "sampling cycle determination processing" is performed.

■Sampling cycle determination processing

This function checks the AT1 sampling period (AT1ST) to determine whether the sampling period has been reached.

- If the sampling period has not been reached, the S.AT1 instruction is terminated.
- If the sampling period has been reached, "response waveform observation processing" is performed.

■Response waveform observation processing

The following processing is performed for the input value (E1).

Item	Processing		
Response waveform observation	The auto tuning counter is incremented.		
	The following processing is performed according to the input value (E1) and last measurement value (PV_{n-1}). $\text{T2}=\text{E1}-\text{PV}_{n-1}$ • Reserve action (PN = 0) • Direct action (PN = 1)		
	The input value (E1) is stored in the last measurement value (PV_{n-1}).		
Maximum ramp value	The following processing is performed according to the ramp (T2) and the S.AT1 instruction is terminated.		
	• Reverse action (PN=0) and AT1 step manipulated value (AT1STEPMV) ≥ 0 • Direct action (PN=1) and AT1 step manipulated value (AT1STEPMV) < 0	Maximum ramp value ≤ Ramp (T2)	• Maximum ramp value = Ramp (T2) • Maximum ramp counter = Counter from the start of auto tuning • Maximum ramp PV = Input value (E1) • Reset the timeout time (after maximum ramp) counter, and start counting again.
		Maximum ramp value > Ramp (T2)	The last maximum ramp value remains unchanged.
	• Direct action (PN=1) and AT1 step manipulated value (AT1STEPMV) ≥ 0 • Reverse action (PN=0) and AT1 step manipulated value (AT1STEPMV) < 0	Maximum ramp value ≥ Ramp (T2)	• Maximum ramp value = Ramp (T2) • Maximum ramp counter = Counter from the start of auto tuning • Maximum ramp PV = Input value (E1) • Reset the timeout time (after maximum ramp) counter, and start counting again.
		Maximum ramp value < Ramp (T2)	The last maximum ramp value remains unchanged.

■Identification processing

The following processing is performed from the maximum ramp value.

Processing	Item
Response speed	The response speed for calculation (R') and response speed (R) are calculated from the following expressions. $R'=\frac{\text{Maximum ramp value (\%)}}{\text{AT1ST}}$, $R=\frac{\lvert R'\rvert}{100}$ (/s)
	When the response speed (R) is 0, the following processing is performed and the S.AT1 instruction is terminated. • Identification alarm BB8 is set to 1. • Auto tuning completion (BB16) is set to 1. • When the step manipulated value set completion flag is 1, the following processing is performed. MV=MV-AT1STEPMV
Equivalent dead time	The Y-axis intercept (b) when tangent is drawn by the response speed for calculation (R') and the equivalent dead time (L) are calculated from the following expressions. b = (PV at maximum ramp) - R'×(counter at maximum ramp)×AT1ST $L=\frac{\text{(Auto tuning start-time PV0)-b}}{R'}$
	When the equivalent dead time (L) is equal to or less than 0, the following processing is performed and the S.AT1 instruction is terminated. • Identification alarm BB8 is set to 1. • Auto tuning completion (BB16) is set to 1. • When the step manipulated value set completion flag is 1, the following processing is performed. MV=MV-AT1STEPMV

■PID constants calculation processing

The response speed (R), equivalent dead time (L), and AT1 step manipulated value (AT1STEPMV) are assigned to the adjustment rule to calculate PID constants.

- Control method

The control method is selected according to the integral constant TI (I) and derivative constant TD (D).

Integral constant T_I(I)	Derivative constant T_D(D)	Control method
$T_I \leq 0$	—	Proportional control (P action) only
$T_I > 0$	$T_D \leq 0$	PI control (PI action)
	$T_D > 0$	PID control (PID action)

- Adjustment rule

N method: The adjustment rule using the step response of Ziegler and Nichols is used.

Control method	Proportional gain K_P(P)	Integral constant T_I(I)	Derivative constant T_D(D)
P	$\frac{1}{R \times L} \times \frac{\lvert AT1STEPMV \rvert}{100}$	0	0
PI	$\frac{0.9}{R \times L} \times \frac{\lvert AT1STEPMV \rvert}{100}$	3.33×L	0
PID	$\frac{1.2}{R \times L} \times \frac{\lvert AT1STEPMV \rvert}{100}$	2×L	0.5×L

- PID constants storing

The following processing is performed and the S.AT1 instruction is terminated.

- PID constants are stored in the gain (P), integral constants (I), and derivative constants (D).
- Auto tuning completion (BB16) is set to 1.
- The AT1 step manipulated value (AT1SETPMV) is subtracted from the manipulated value (MV) and the result is stored in the manipulated value (MV).

MV=MV-AT1STEPMV

Operation error

Error code	Error content
3400H	An invalid operation (such as division by zero) is performed.
3402H	Input data (s1) is a subnormal number or NaN (not a number).
3403H	An overflow has occurred.
3405H	AT1 sampling period (AT1ST) < 0
	AT1 timeout time (AT1TOUT1) < 0
	AT1 timeout (after maximum ramp) time (AT1TOUT2) < 0
	Execution cycle (ΔT) < 0
	(AT1 sampling period (AT1ST)÷execution cycle (ΔT)) > 32767
	(AT1 timeout time (AT1TOUT1)÷execution cycle (ΔT)) > 32767
	(AT1 timeout (after maximum ramp) time (AT1TOUT2)÷Execution cycle (ΔT)) > 32767

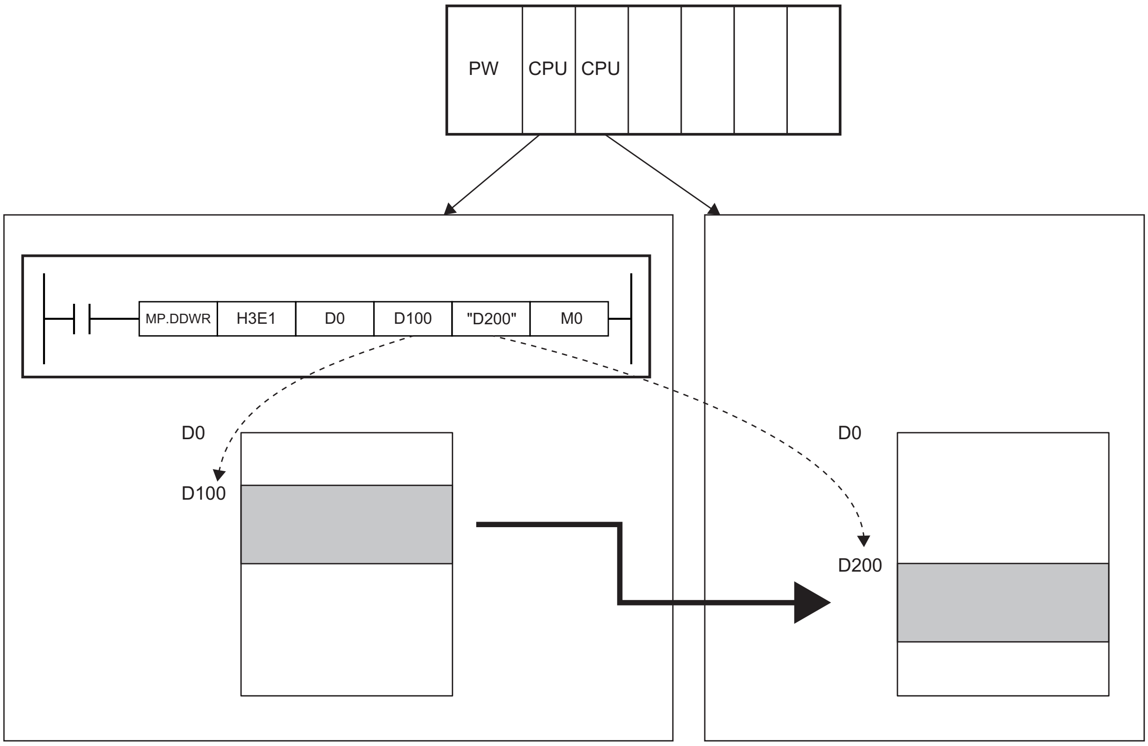
12 MULTIPLE CPU DEDICATED INSTRUCTIONS

12.1 Another CPU Module Access Instructions

Overview

The host CPU module read or write device data from or to another CPU module by using another CPU module access instructions.

The following figure shows the operation for writing data from CPU No.1 to CPU No.2 by using another CPU module access instruction.



The following table lists another CPU module access instructions.

Instruction symbol	Description	Application
D(P).DDRD	Loads the device data of another CPU module to the device of the host CPU module.	Use these instructions to read or write data at the timing set by the fixed scan communication function.
D(P).DDWR	Writes the device data of the host CPU module to the device of another CPU module.	
M(P).DDRD	Loads the device data of another CPU module to the device of the host CPU module.	Use these instructions to read or write data at the timing of each CPU module.
M(P).DDWR	Writes the device data of the host CPU module to the device of another CPU module.	

Setting parameters

To use the D(P).DDRD or D(P).DDWR instruction, the fixed scan communication function of the system parameters needs to be set.

Readable/writable devices

The following table lists the devices that can be read from or written to another CPU module by using another CPU module access instructions.

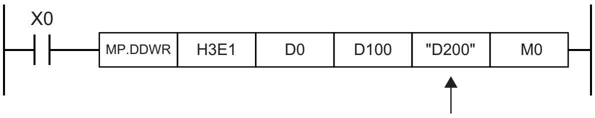
Classification	Type	Device name[2]	Target device setting	Condition
User device System device	Bit device	X, Y, M, L, B, F, SB, SM	Available	Satisfy the following conditions. • Four digits are specified by 16 bits. • The start bit device is a multiple of 16 (10H).
	Word device	T, ST, C, D, W, SW, SD	Available	None
File register	Word device	R, ZR	Available	None
Indirect specification[1]		—	Not available	None

[1] When an indirect specification is used for the target device, the device to be written or read is determined from the indirect address stored in the device of the host CPU module.

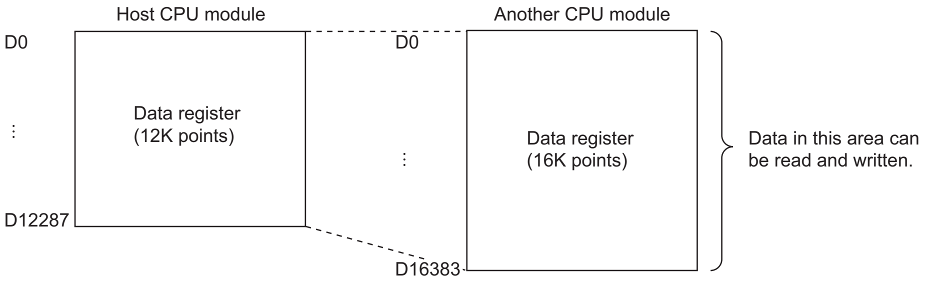
[2] Index modification (Z, ZZ representation) using the index register by the string specification can be performed. Another CPU module is accessed with a value which is index-modified by the value of the index register of the host CPU module.
For example, "K4M0Z0" with Z0=16 causes M0+16=M16, causing K4M16 to access another CPU module.
Similarly, "ZR0ZZ0" with Z0, Z1=100000 causes ZR0+10000=ZR100000, causing ZR100000 to access another CPU module.

Device specification method and readable/writable ranges

Specify the device of another CPU module with character strings.
Directly specify "D200", the write-target device number in another CPU module.



The string specification enables writing to or reading from every range of the device in another CPU module. For example, when the data register of the host CPU has 12K points while the data register of another CPU module has 16K points, 16K points of data can be written to or read from the head of the data register of another CPU module.



Point

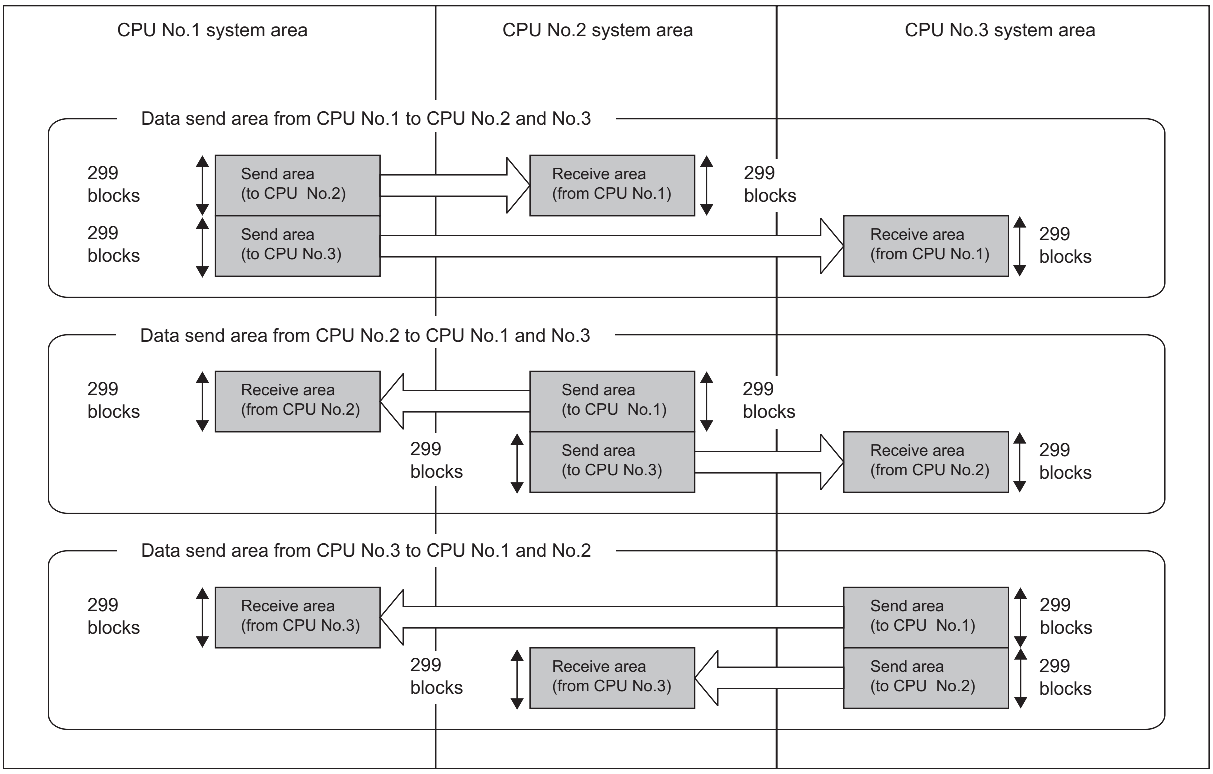
- Even if "0" is added to higher places of a device number, the device is processed the same as when it is not added. For example, "D1" and "D0001" are each processed as D1.
- Device numbers are not case-sensitive in terms of processing. For example, "D1" and "d1" are each processed as D1.
- Note that if a device not existing in another CPU module is specified by a character string, the instruction will be completed with an error.

Number of available blocks

Another CPU module access instructions use the system area in minimum units of blocks, each consisting of 16 words.
The following table lists the numbers of blocks available for another CPU module access instructions.

Number of CPU modules	Maximum number of blocks
2	599
3	299
4	199

The following figure shows how blocks are used in a multiple CPU system consisting of three CPU modules.



Maximum number of data points that can be read or written

The maximum number of data points that can be read or written by an instruction depends on the number of CPU modules in a multiple CPU system configuration.

Number of CPU modules	Maximum number of data points that can be read	Maximum number of data points that can be written
2 modules	8192 point	8192 point
3 modules	4096 point	4096 point
4 modules	2048 point	2048 point

Number of blocks used by instructions

The number of blocks used by instructions depends on the number of read/write data points. The following table lists the numbers of blocks used by instructions.

Reading/ writing	Number of blocks	Example
Read	Number of blocks used by instructions = (21 + number of read data points)÷16	• When the number of read data points is 100 Number of blocks used by instructions = (21 + 100)÷16 = 7 [blocks]
Write	Number of blocks used by instructions = (19 + number of write data points)÷16	• When the number of write data points is 100 Number of blocks used by instructions = (19 + 100)÷16 = 7 [blocks]

Simultaneous execution of another CPU module access Instructions

Another CPU module access instructions can be executed simultaneously within the range of the following expression.

Conditions under which another CPU module access instructions can be executed simultaneously
[Number of blocks available for each CPU module]≥[total number of blocks used by concurrently executed instructions]

If executing another CPU module access instruction causes the number of blocks used by the CPU module access instructions to exceed the total number of blocks in the system area, the instruction is not executed (no processing) in the relevant scan and is executed in the next scan.

Note, however, that this instruction is completed with an error if the number of empty block in the system area is less than the value specified in SD796 to SD799 (maximum number of blocks for multiple CPU dedicated instructions) when the instruction is executed.

The table below shows whether another CPU module access instruction can be executed when the number of empty blocks in the system area is less than the number of blocks used by another CPU module access instructions or the value set in SD796 to SD799.

Size relationship between the value set in SD[*3]and number of empty blocks[*2]	Size relationship between the number of blocks used by instruction[*1] and number of empty blocks	
	Number of blocks used by instruction≤ and number of empty blocks	Number of blocks used by instruction > number of empty blocks
Value set in SD ≤ number of empty blocks	Executed	Not executed (non-processing)
Value set in SD > number of empty blocks	Completed with an error	

*1 Number of blocks used by another CPU module access instructions

*2 Number of empty blocks in the system area

*3 Value set in SD796 to SD799

Interlock applied when another CPU module access instructions are used

Special relay SM796 to SM799 is used for interlocking among another CPU module access instructions.

When executing multiple another CPU module access instructions concurrently, use SM796 to SM799 for interlocking among these instructions.

Point

When using SM796 to SM799, specify the maximum numbers of blocks of the instructions used by individual CPU modules in SD796 to SD799. For example, when the maximum number of blocks used by another CPU module access instructions executed for CPU module No. 3 is 5, specify 5 in SD798.

When the number of blocks specified in any of SD796 to SD799 is exceeded, the relevant special relay (SM796 to SM799) turns on.

Precautions

- Execute the D(P).DDWR, M(P).DDWR, D(P).DDRD, or M(P).DDRD instruction while the read/write target CPU module is on. If the instruction is executed while the target CPU is not on, the instruction performs no processing.
- After the D(P).DDWR, M(P).DDWR, D(P).DDRD, or M(P).DDRD instruction is executed, do not change the device range specified in the setting data before the completion device is turned on; otherwise, the completion status and completion device data can no longer be stored in the system.
- SB/SW and SM/SD include the system information area. When writing data with the D(P)DDWR or M(P).DDWR instruction, be careful not to overwrite the system information area.
- If the number of blocks used by the instruction to be executed is greater than the value set in SD796 to SD799, the instruction may not be executed (terminated abnormally) even if it is interlocked with SD796 to SD799.
- Set SD796 to SD799 before executing the instruction for the corresponding CPU module. (It is recommended to set them in the first scan after the CPU module runs.)

Reading device data from another CPU module

D(P).DDRD, M(P).DDRD

- D(P).DDRD

RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

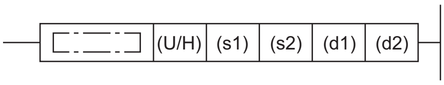
The RnENCPU with firmware version "25" or later supports these instructions. (Use an engineering tool with version "1.032J" or later.)

- M(P).DDRD

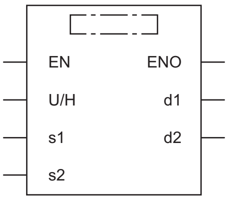
RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

The RnENCPU with firmware version "25" or later supports these instructions. (Use an engineering tool with version "1.032J" or later.)

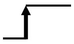
These instructions read device data from another CPU module in a multiple CPU system.

Ladder	ST
	ENO:=D_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=DP_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=M_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=MP_DDRD(EN,U/H,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
D.DDRD M.DDRD	
DP.DDRD MP.DDRD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U/H)	Start I/O number (first three digits in four-digit hexadecimal representation) of another CPU module	3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)	Start device of host CPU module where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device of another CPU module where the data to be read is stored	—	String	ANYSTRING_SINGLE
(d1)	Start device of host CPU module for storing the data that has been read	—	Word	ANY16[*1]
(d2)	Completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U/H)	○	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

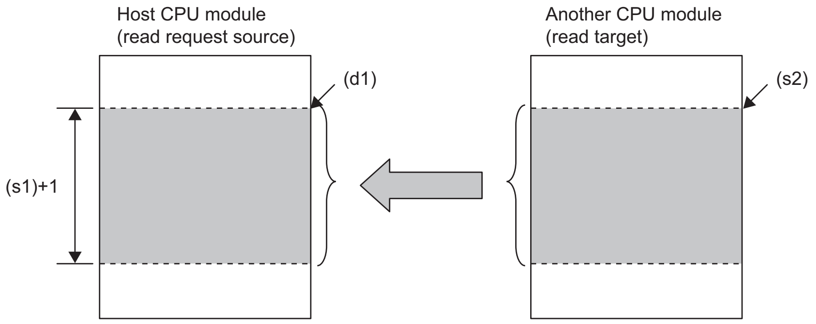
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Completion status	The completion status is stored. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+1	Number of read data points	Specify the number of read data points in units of words.	1 to 8192[*1]	User

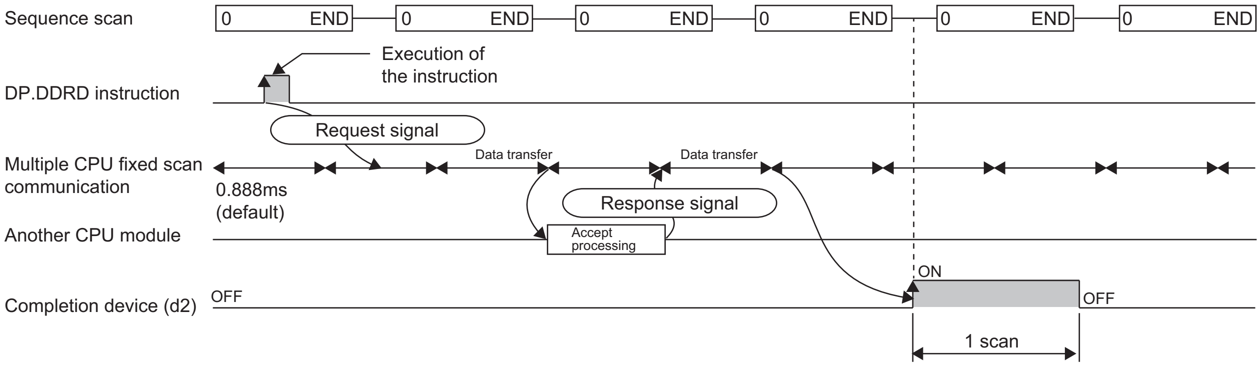
[*1] This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be read varies depending on the system configuration. (☞ Page 1401 Another CPU Module Access Instructions)

Processing details

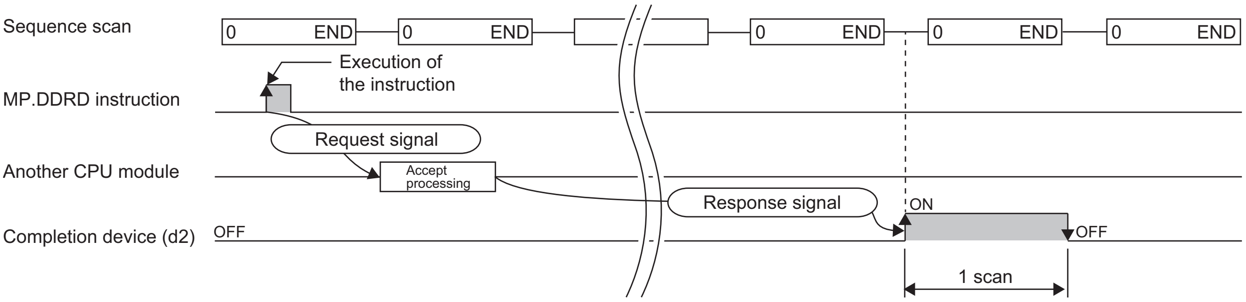
- In a multiple CPU system, these instructions read the data in the device specified by (d1) in the host CPU module, by the number of read data points specified by (s1)+1, and store it in the device specified by (d1) and later in another CPU module (U/H).



- The following figure shows an outline of operation of the D(P).DDRD and M(P).DDRD instructions.
  - Outline of operation of the D(P).DDRD instructions



- Outline of operation of the M(P).DDRD instructions



- The execution of the D(P).DDRD or M(P).DDRD instruction and whether it has been completed normally or with an error can be checked with the completion device (d2) or completion status indication device (d2)+1.
  - Completion device (d2)

The completion device turns on in END processing of the scan performed upon completion of the D(P).DDRD or M(P).DDRD instruction and turns off in the next END processing.

- Completion status indication device (d2)+1

The completion device turns on or off depending on the completion status of the D(P).DDRD or M(P).DDRD instruction.
When completed normally: Unchanged from off.
When completed with an error: Turns on in END processing of the scan performed upon completion of the D(P).DDRD or M(P).DDRD instruction and turns off in the next END processing.
When completed with an error, an error code is stored in the device (completion status) specified by (s1)+0.

- The number of blocks used by instructions depends on the number of read data points. For the number of blocks used by instructions, refer to the following.

☞ Page 1401 Another CPU Module Access Instructions

- For the specifiable target devices in the read target CPU module, refer to the following.

☞ Page 1401 Another CPU Module Access Instructions

- If an instruction is executed while the system area has no empty block, it is completed with an error. Completion with an error can be prevented by setting the number of blocks used by instructions in SD796 to SD799 and using SM796 to SM799 as interlocks.

Operation error

Error code (SD0)	Description
2800H	The start I/O number (first three digits in four-digit hexadecimal representation) of the specified CPU module is out of the range, 3E0H to 3E3H.
2801H	An invalid another CPU module is specified. • A reserved CPU module is specified. • An unmounted CPU module is specified.
2802H	Another CPU module does not support the D(P).DDRD and M(P).DDRD instructions.
2803H	The host CPU module is specified as another CPU module.
2810H	A CPU module which cannot execute the instruction is specified as another CPU module.
3404H	An invalid character string is used to specify a device.
3405H	The number of read data points specified by (s1)+1 is out of the range from 0 to 8192.[*2]
3440H	The D(P).DDRD instruction is executed with the inter-CPU fixed-scan communication disabled.
3441H	The specified number of data points exceeds the size of the system area that can be used by each CPU module.

[*2] This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be read varies depending on the system configuration. (☞ Page 1401 Another CPU Module Access Instructions)

Error code ((s1)+0)	Description
0010H	The instruction request to the target CPU module exceeds the allowable value. (There is not empty block in the system area.)
1001H	The device of another CPU module specified by (s2) cannot be used by another CPU module. Alternatively, it is out of the device range.
1080H	The number of read data points that has been set by the D(P).DDRD or M(P).DDRD instruction is 0.

Writing device data to another CPU module

D(P).DDWR, M(P).DDWR

- D(P).DDWR

RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

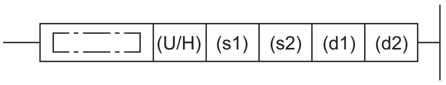
The RnENCPU with firmware version "25" or later supports these instructions. (Use an engineering tool with version "1.032J" or later.)

- M(P).DDWR

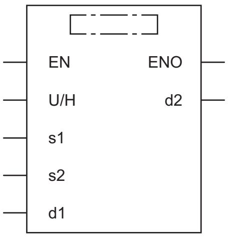
RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

The RnENCPU with firmware version "25" or later supports these instructions. (Use an engineering tool with version "1.032J" or later.)

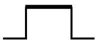
These instructions write device data to another CPU module in a multiple CPU system.

Ladder	ST
	ENO:=D_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=DP_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=M_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=MP_DDWR(EN,U/H,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
D.DDWR M.DDWR	
DP.DDWR MP.DDWR	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U/H)	Start I/O number (first three digits in four-digit hexadecimal representation) of another CPU module	3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)	Start device of host CPU module where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device of host CPU module where the write data is stored	—	Word	ANY16[1]
(d1)	Start device of another CPU module for storing the written data	—	String	ANYSTRING_SINGLE
(d2)	Completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U/H)	○	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

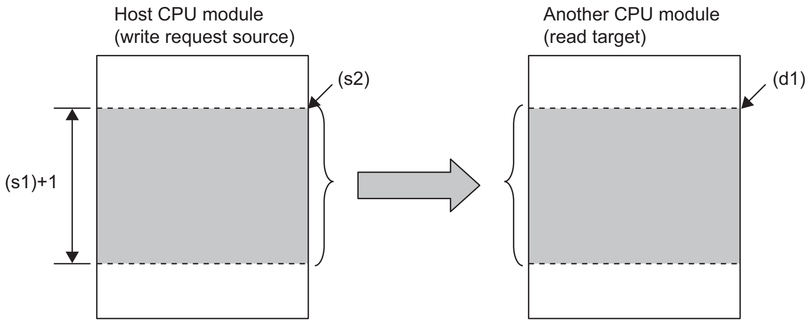
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Completion status	The completion status is stored. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+1	Number of write data points	Specify the number of write data points in units of words.	1 to 8192[*1]	User

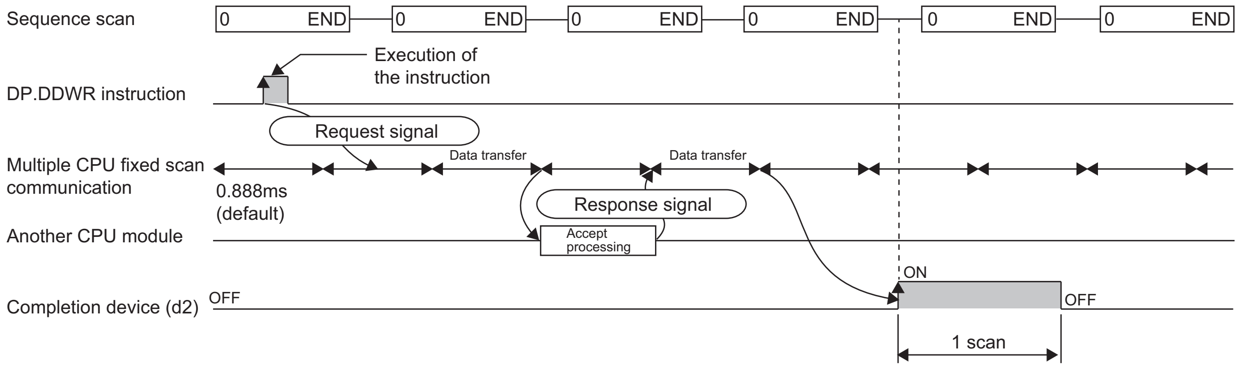
[*1] This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be written varies depending on the system configuration. (☞ Page 1401 Another CPU Module Access Instructions)

Processing details

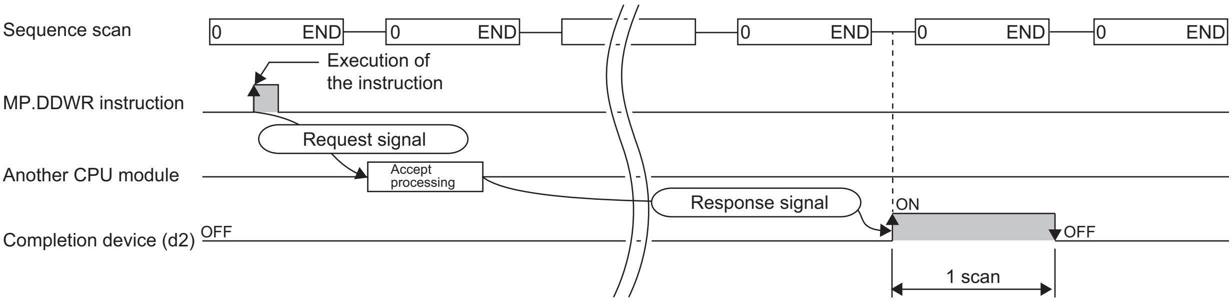
- In a multiple CPU system, these instructions read the data in the device specified by (s2) in the host CPU module, by the number of write data points specified by (s1)+1, and store it in the device specified by (d1) and later in another CPU module (U/H).



- The following figure shows an outline of operation of the D(P).DDWR and M(P).DDWR instructions.
  - Outline of operation of the D(P).DDWR instructions



- Outline of operation of the M(P).DDWR instructions



- The execution of the D(P).DDWR or M(P).DDWR instruction and whether it has been completed normally or with an error can be checked with the completion device (d2) or completion status indication device (d2)+1.

- Completion device (d2)

The completion device turns on in END processing of the scan performed upon completion of the D(P).DDWR or M(P).DDWR instruction and turns off in the next END processing.

- Completion status indication device (d2)+1

The completion device turns on or off depending on the completion status of the D(P).DDWR or M(P).DDWR instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on in END processing of the scan performed upon completion of the D(P).DDWR or M(P).DDWR instruction and turns off in the next END processing.

When completed with an error, an error code is stored in the device (completion status) specified by (s1)+0.

- The number of blocks used by instructions depends on the number of write data points. For the number of blocks used by instructions, refer to the following.

☞ Page 1401 Another CPU Module Access Instructions

- For the specifiable target devices in the write target CPU module, refer to the following.

☞ Page 1401 Another CPU Module Access Instructions

- If an instruction is executed while the system area has no empty block, it is completed with an error. Completion with an error can be prevented by setting the number of blocks used by instructions in SD796 to SD799 and using SM796 to SM799 as interlocks.

Operation error

Error code (SD0)	Description
2800H	The start I/O number (first three digits in four-digit hexadecimal representation) of the specified CPU module is out of the range, 3E0H to 3E3H.
2801H	An invalid another CPU module is specified. • A reserved CPU module is specified. • An unmounted CPU module is specified.
2802H	Another CPU module does not support the D(P).DDWR and M(P).DDWR instructions.
2803H	The host CPU module is specified as another CPU module.
2810H	A CPU module which cannot execute the instruction is specified as another CPU module.
3404H	An invalid character string is used to specify a device.
3405H	The number of write data points specified by (s1)+1 is out of the range from 0 to 8192.[*2]
3440H	The D(P).DDWR instruction is executed with the inter-CPU fixed-scan communication disabled.
3441H	The specified number of data points exceeds the size of the system area that can be used by each CPU module.

[*2] This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be written varies depending on the system configuration. (☞ Page 1401 Another CPU Module Access Instructions)

Error code ((s1)+0)	Description
0010H	The instruction request to the target CPU module exceeds the allowable value. (There is not empty block in the system area.)
1001H	The device of another CPU module specified by (d1) cannot be used by another CPU module. Alternatively, it is out of the device range.
1080H	The number of write data points that has been set by the D(P).DDWR or M(P).DDWR instruction is 0.

13 SFC PROGRAM INSTRUCTIONS

Point

This chapter describes the instructions used in SFC programs. For details on SFC programs, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

13.1 SFC Control Instructions

Checking the status of a step

LD, LDI, AND, ANI, OR, ORI [S□/BL□\S□]

RnCPU RnENCPU RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "12" or later support these instructions. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

- LD: Normally open contact, LDI: Normally closed contact

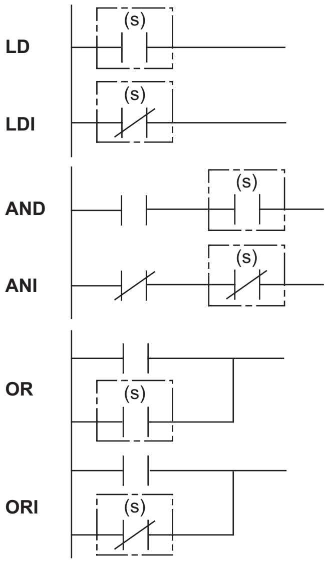
These instructions output the status (active or inactive) of the specified step as the operation result.

- AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result.

- OR: Single normally open contact parallel connection, ORI: Single normally closed contact parallel connection

These instructions perform an OR operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
LD LDI AND ANI OR ORI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, S, 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	$	(BL□\S□)
(s)	○[1]	—	—	—	—	—	—	—	—	—	—	○

[1] Only S can be used.

Processing details

- These instructions check whether the specified step in the specified block is active or not.
- The status (on or off) of each contact will be as follows depending on the status (active or inactive) of the specified step.

Status of the specified step	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
Active	On	Off
Inactive	Off	On

- The following table summarizes specification methods of steps.

Program		Specification method
SFC program	Specifying a step in current block	Use S□.
	Specifying a step in another block	Use BL□\S□
Sequence program		Use BL□\S□ When a block No. is not specified, specify a block No. by using the BRSET instruction. If a block No. is not specified by using the BRSET instruction, the target block will be block 0.

- Specify the block No. within the range of 0 to 319, and the step No. within the range of 0 to 511. If the block No. or the step No. specified is out of range, both of the normally open contact and normally closed contact turn off.
- Execute the instruction only when an SFC program exists (SM320 is on) and SM321 is on.

Precautions

- The status (on or off) of the contact of a step specified by S□, which is specified without a block No., on the sequence program, cannot be monitored on the engineering tool. However, the operation is performed. If the contact is on in the CPU module, coil output turns on.
- When no SFC program exists (SM320 is off) or SM321 is off, the status (on or off) of the contact will be as follows:

Specified block No.	Specified step No.	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
0 to 319	0 to 511	Off	On
	Less than 0, 512 or more (out of range)	Off	Off
Less than 0, 320 or more (out of range)	0 to 511	Off	Off
	Less than 0, 512 or more (out of range)	Off	Off

Operation error

There is no operation error.

Checking the status of a block

LD, LDI, AND, ANI, OR, ORI [BL□]

RnCPU RnENCPU RnPCPU (Process) ✗ RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "12" or later support these instructions. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

- LD: Normally open contact, LDI: Normally closed contact

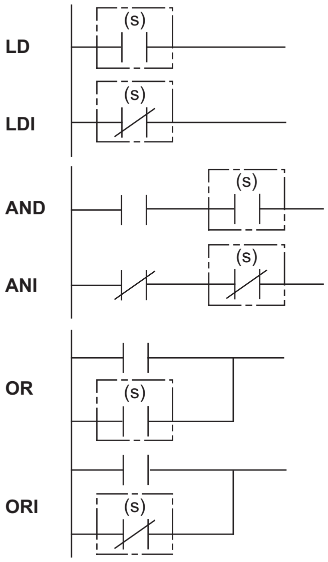
These instructions output the status (active or inactive) of the specified block as the operation result.

- AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result.

- OR: Single normally open contact parallel connection, ORI: Single normally closed contact parallel connection

These instructions perform an OR operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
LD LDI AND ANI OR ORI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

- These instructions check whether the specified block is active or not.
- The status (on or off) of each contact will be as follows depending on the status (active or inactive) of the specified block.

Status of the specified block	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
Active	On	Off
Inactive	Off	On

- Specify the block No. within the range of 0 to 319. If the block No. specified is out of range, both of the normally open contact and normally closed contact turn off.
- Execute the instruction only when an SFC program exists (SM320 is on) and SM321 is on.

Precautions

- When no SFC program exists (SM320 is off) or SM321 is off, the status (on or off) of the contact will be as follows:

Specified block No.	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
0 to 319	Off	On
Less than 0, 320 or more (out of range)	Off	Off

Operation error

There is no operation error.

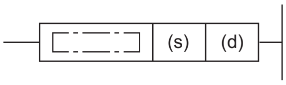
Batch-reading the status of steps

MOV(P) [K4S□/BL□\K4S□]

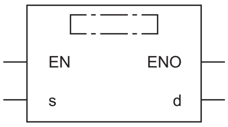
RnCPU RnENCPU RnPCPU (Process) ✕RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

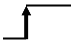
These instructions batch-read (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block, and store the read data in the specified device.

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

FBD/LD



■Execution condition

Instruction	Execution condition
MOV	
MOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device number where the transfer source data is stored	—	16-bit signed binary	ANY16
(d)	Transfer destination device number	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

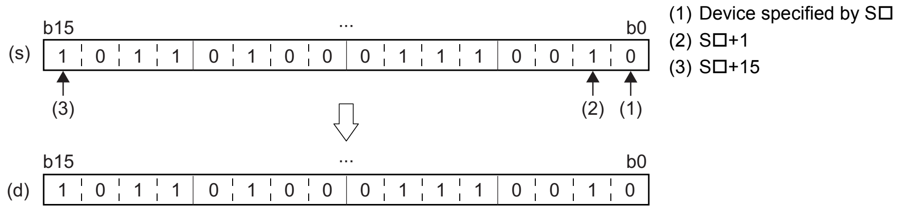
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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)	○[*1]	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	○	—	—	○	—	—	—	—

[*1] Only S can be used.

Processing details

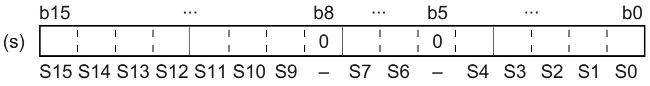
- These instructions batch-read (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d). When the step is active, 1 is stored. When the step is inactive, 0 is stored.



- When there is a missing step No., 0 is stored in the corresponding bit.

Ex.

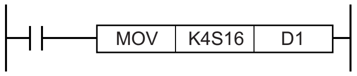
When the step No.5 and No.8 are missing in the specified block (The status of each step is stored in other bits.)

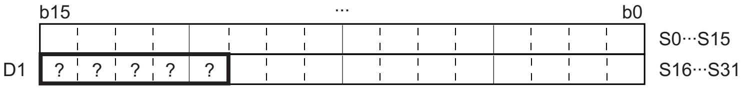


- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

When the last step No. in the block is S26 and the status of steps (S16 to S31) are read to D1. (The status of each step is stored in other bits.)

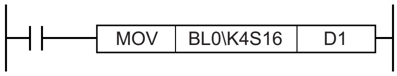


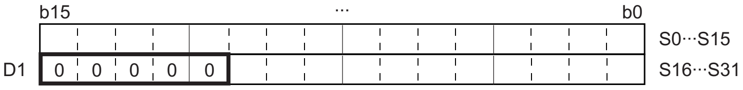


- If the read target range exceeds the number of steps in the specified block, 0 is stored in the bits exceeding the existing step No.

Ex.

When the last step No. in the block is 26 and the status of steps (No.16 to No.31) are read to D1 (The status of each step is stored in other bits.)





- If the block No. that does not exist or does not include the read target data is specified, or if the specified block No. is correct but the non-existent step is specified, 0 is read and stored in all bits.
- If the instruction is executed while no SFC program exists, 0 is read and stored in all bits.

Operation error

Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range, 0 to 319.
	When a block No. is specified, the specified step No. is out of the range, 0 to 511.

Point

Use a digit specification to specify a step.

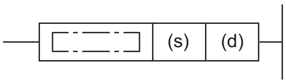
- To specify a step in the current block of an SFC program, use K4S□.
- To specify a step in another block of an SFC program, use BL□\K4S□.
- To specify a step of a sequence program, use BL□\K4S□.

DMOV(P) [K8S□/BL□\K8S□]

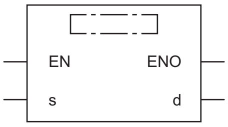
RnCPU **RnENCPU** **RnPCPU (Process)** ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

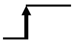
These instructions batch-read (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block, and store the read data in the specified device.

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

FBD/LD



■Execution condition

Instruction	Execution condition
DMOV	
DMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device number where the transfer source data is stored	—	32-bit signed binary	ANY32
(d)	Transfer destination device number	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

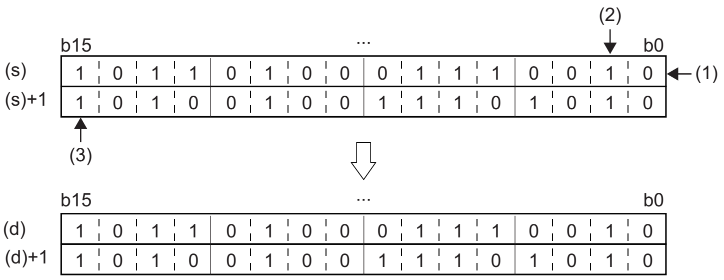
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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)	○*1	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	○	○	○	○	—	—	—	—

*1 Only S can be used.

Processing details

- These instructions batch-read (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d). When the step is active, 1 is stored. When the step is inactive, 0 is stored.

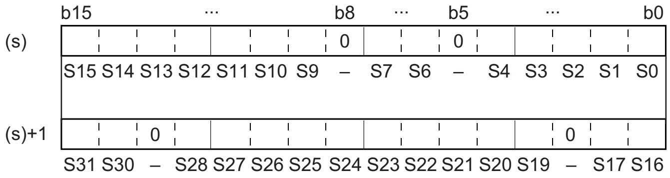


(1) Device specified by S□
(2) S□+1
(3) S□+31

- When there is a missing step No., 0 is stored in the corresponding bit.

Ex.

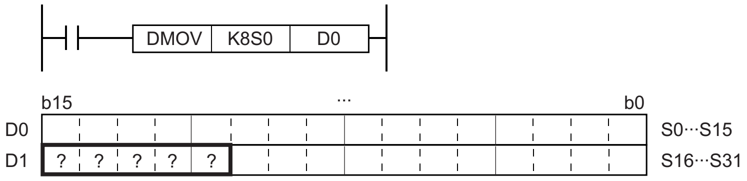
When the step No.5, 8, 18, and 29 are missing in the specified block (The status of each step is stored in other bits.)



- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

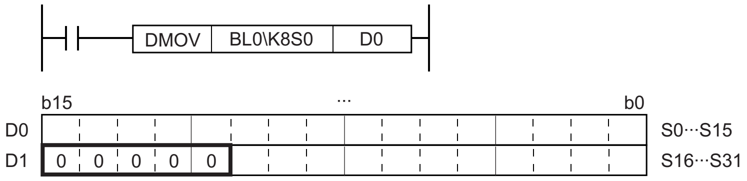
When the last step No. in the block is 26 and the status of the steps (No.0 to No.31) are read to D0 and D1 (The status of each step is stored in other bits.)



- If the read target range exceeds the number of steps in the specified block, 0 is stored in the bits exceeding the existing step No.

Ex.

When the last step No. in the block is 26 (The status of each step is stored in other bits.)



- If the block No. that does not exist or does not include the read target data is specified, or if the specified block No. is correct but the non-existent step is specified, 0 is read and stored in all bits.
- If the instruction is executed while no SFC program exists, 0 is read and stored in all bits.

Operation error

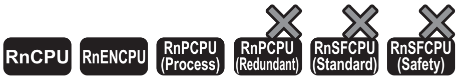
Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range, 0 to 319.
	When a block No. is specified, the specified step No. is out of the range, 0 to 511.

Point

Use a digit specification to specify a step.

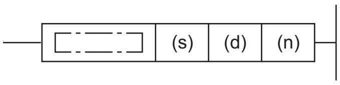
- To specify a step in the current block of an SFC program, use K8S□.
- To specify a step in another block of an SFC program, use BL□\K8S□.
- To specify a step of a sequence program, use BL□\K8S□.

BMOV(P) [K4S□/BL□\K4S□]

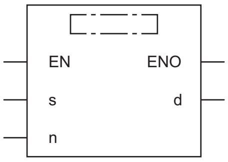


- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

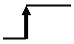
These instructions batch-read (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.

Ladder	ST
	ENO:=BMOV(EN,s,n,d); ENO:=BMOVP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BMOV	
BMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start number of a device where the transfer target data is stored	—	16-bit signed binary	ANY16
(d)	Transfer destination device start number	—	16-bit signed binary	ANY16
(n)	Number of transfer data points	0 to 65535	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

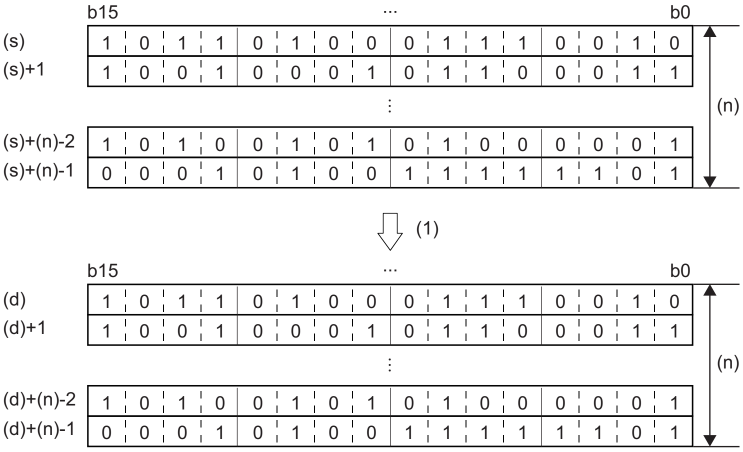
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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)	○*1	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only S can be used.

Processing details

- These instructions batch-read (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d).



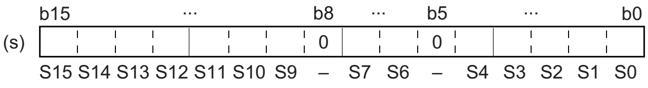
	b15							···								b0
(s)	1	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0
(s)+1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1
⋮																
(s)+(n)-2	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	1
(s)+(n)-1	0	0	0	1	0	1	0	0	1	1	1	1	1	1	0	1

	b15							···								b0
(d)	1	0	1	1	0	1	0	0	0	1	1	1	0	0	1	0
(d)+1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1
⋮																
(d)+(n)-2	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	1
(d)+(n)-1	0	0	0	1	0	1	0	0	1	1	1	1	1	1	0	1

- When there is a missing step No., 0 is stored in the corresponding bit.

Ex.

When the step No.5 and No.8 are missing in the specified block (The status of each step is stored in other bits.)

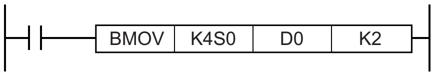


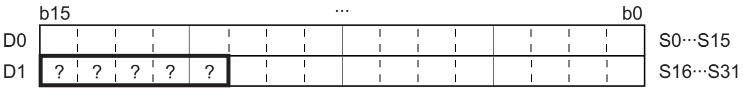
	b15			···				b8	···		b5		···			b0
(s)								0			0					
	S15	S14	S13	S12	S11	S10	S9	–	S7	S6	–	S4	S3	S2	S1	S0

- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

When the last step No. in the block is 26 and the status of the steps (two words from step No.0) is read to D0 and D1 (The status of each step is stored in other bits.)



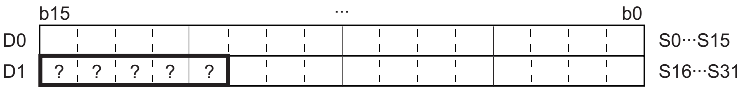


	b15							···								b0	
D0																	S0···S15
D1	?	?	?	?	?												S16···S31

- If the read target range exceeds the number of steps in the specified block, or if the non-existent step is specified as a start step, undefined values are stored.

Ex.

When the last step No. in the block is 26 and the status of the steps (two words from step No.0) is read to D0 and D1 (The status of each step is stored in other bits.)



	b15							···								b0	
D0																	S0···S15
D1	?	?	?	?	?												S16···S31

- If the instruction is executed while no SFC program exists, or if the block No. that does not exist or does not include the read target data is specified, 0 is read and stored in all bits.

Operation error

Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range, 0 to 319.
	When a block No. is specified, the specified step No. is out of the range, 0 to 511.

Point

Use a digit specification to specify a step.

- To specify a step in the current block of an SFC program, use K4S□.
- To specify a step in another block of an SFC program, use BL□\K4S□.
- To specify a step of a sequence program, use BL□\K4S□.

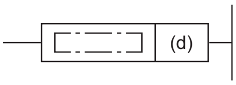
Starting a block

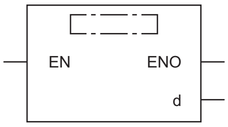
SET [BL□]

RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction activates the specified block, and executes a step sequence starting from an initial step.

Ladder	ST
	ENO:=SET(EN,d);

FBD/LD


■Execution condition

Instruction	Execution condition
SET	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. to be activated (Set (on) target bit device number)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○[*1]

*1 The BL can be index modified.

Processing details

- This instruction activates the specified block, and executes a step sequence starting from an initial step. When there are several initial steps, all the initial steps are activated.
- If the block start/end bit of the SFC information device is set, the corresponding bit device turns on.
- If the instruction is executed to an active block, the instruction is ignored and processing will continue.
- If the instruction is executed to an inactive block for which online change is being performed, the instruction is ignored and processing will continue.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range, 0 to 319.

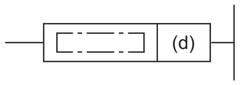
Ending a block

RST [BL□]

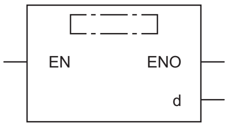
RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction deactivates the specified block.

Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. to be deactivated (Reset (off) target bit device number)	—	Bit	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○[*1]

[*1] The BL can be index modified.

Processing details

- This instruction deactivates the specified block independently.
- All the active steps are deactivated and coil outputs are turned off.
- If the block start/end bit of the SFC information device is set, the corresponding bit device turns off.
- If the instruction is executed to an inactive block, the instruction is ignored.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range, 0 to 319.

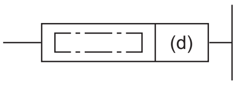
Pausing a block

PAUSE [BL□]

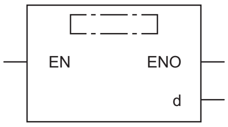
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction temporarily stops the step sequence in the specified block.

Ladder	ST
	ENO:=PAUSE(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PAUSE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. where the sequence is temporarily stopped	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○*1

*1 The BL can be index modified.

Processing details

- This instruction temporarily stops the step sequence in the specified block.
- If the block pause/restart bit of the SFC information device is set, the corresponding bit device turns on.
- Stop timing of each step after execution of the instruction differs depending on the setting (immediate stop or stop after transition) of the block stop mode bit of the SFC information device.

Setting of the block stop mode bit	Stop timing
Immediate stop	All the steps become inactive immediately after execution of the instruction.
Stop after transition	Steps continue operation even after the execution. After transition condition is satisfied, the operation stops at next step.

- The coil output status by using the OUT instruction at the temporary stop depends on the status of SM325.

Status of SM325	Coil output status by using the OUT instruction
On	The coil output remains the state immediately before stop.
Off	The coil output turns off at stop.

Precautions

If the sequence is stopped while SM325 is off, coil HOLD steps become inactive. The sequence cannot be restarted with the hold status. If the sequence is stopped while SM325 is on, the sequence can be restarted with the hold status.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range, 0 to 319.

Point

Operation of the PAUSE and RESTART instructions depends on the combination of the SM325 status, block stop mode bit setting of SFC information device, and step hold status. For details, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

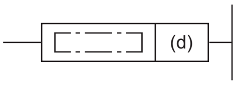
Restarting a block

RSTART [BL□]

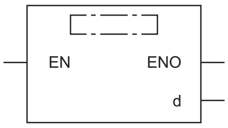
RnCPU RnENCPU RnPCPU (Process) ✕ RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

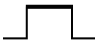
This instruction releases the temporary stop, and restarts the sequence from the step where the sequence was stopped in the specified block.

Ladder	ST
	ENO:=RSTART(EN, d);

FBD/LD



■Execution condition

Instruction	Execution condition
RSTART	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. where the temporary stop is released	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○[1]

[1] The BL can be index modified.

Processing details

- This instruction restarts the sequence from the step where the sequence was stopped in the specified block. Operation HOLD steps (with or without transition check), which have been stopped holding the operation status, restart operation with the hold status.
- If the block pause/restart bit of the SFC information device is set, the corresponding bit device turns off.
- After the step sequence is restarted, the operation of the PLS instruction and the instructions which is executed only on the rising edge depends on the status of SM325.

Status of SM325	Operation of the PLS instruction and the instructions executed on the rising edge
On (Coil output is held.)	The instruction is not executed.
Off (Coil output is off.)	The instruction is executed again.

Precautions

If the sequence is stopped while SM325 is off, coil HOLD steps become inactive. The sequence cannot be restarted with the hold status. If the sequence is stopped while SM325 is on, the sequence can be restarted with the hold status.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range, 0 to 319.

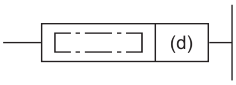
Activating a step

SET [S□/BL□\S□]

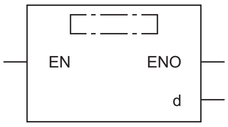
RnCPU RnENCPU RnPCPU (Process) ✗RnPCPU (Redundant) ✗RnSFCPU (Standard) ✗RnSFCPU (Safety)

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction activates the specified step.

Ladder	ST
 (d)	ENO:=SET(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SET	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Step No. to be activated (Set (on) target bit device number)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	○*1

*1 Only S, BL□\S□ can be used. The devices can be index modified.

Processing details

- This instruction activates the specified step in the specified block. The operation of the specified block will be as follows depending on the status (active or inactive) of the specified block.

Status of the specified block	Operation
Inactive	The specified block is activated at execution of the instruction, and starts the processing from the specified step. If the block start/end bit of the SFC information device is set, the corresponding bit device turns on.
Active	The specified step is newly activated while the step which has already been active continues processing.

- If the instruction is executed to an active step, the instruction is ignored and processing will continue. Note that if the specified step is holding the operating status, the hold status is reset. The step becomes a normal step, and executes the action and transition.
- If the instruction is executed to an inactive block for which online change is being performed, the instruction is ignored and processing will continue.
- When there are several initial steps, any of them can be selected and activated.
- If no block is specified, the following block is targeted depending on the execution program type.
- Sequence program: Block 0
- SFC program (within the action): Block where the instruction is executed (current block)

Precautions

- In a simultaneous sequence, specify and activate all the steps by using the SET instruction (Activating a step). If there is any inactive step left, a convergence is not performed. Likewise, if the RST instruction (Deactivating a step) is executed to a single step in a simultaneous sequence, a convergence condition is not satisfied.
- The current step No. cannot be specified within the action of the SFC program. If specified, an error occurs.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist. (SET/RST BL□\S□)
31A2H	When a block No. is specified, the specified block No. is out of the range, 0 to 319.
31B1H	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
	No specified step exists in the specified block. (SET/RST BL□\S□/S□)
31B2H	The specified step No. is out of the range, 0 to 511.
31B5H	The current step is specified within the action.

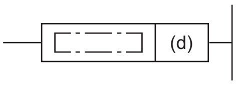
Deactivating a step

RST [S□/BL□\S□]

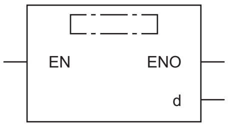
RnCPU **RnENCPU** **RnPCPU (Process)** ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction deactivates the specified step.

Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Step No. to be deactivated (Reset (off) target bit device number)	—	Bit	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, S, 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	$	(BL□\S□)
(d)	○*1	—	—	—	—	—	—	—	—	—	—	○*1

*1 Only S, BL□\S□ can be used. The devices can be index modified.

Processing details

- This instruction deactivates the specified step in the specified block.
- When the number of active steps in the specified becomes 0, the specified block will be inactive. If the block start/end bit of the SFC information device is set, the corresponding bit device turns off.
- If the instruction is executed to an inactive step, the instruction is ignored.
- If no block is specified, the following block is targeted depending on the execution program type.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)

Precautions

- In a simultaneous sequence, specify and activate all the steps by using the SET instruction (Activating a step). If there is any inactive step left, a convergence is not performed. Likewise, if the RST instruction (Deactivating a step) is executed to a single step in a simultaneous sequence, a convergence condition is not satisfied.
- The current step No. cannot be specified within the action of the SFC program. If specified, an error occurs.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist. (SET/RST BL□\S□)
31A2H	When a block No. is specified, the specified block No. is out of the range, 0 to 319.
31B1H	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
	No specified step exists in the specified block. (SET/RST BL□\S□/S□)
31B2H	The specified step No. is out of the range, 0 to 511.
31B5H	The current step is specified within the action.

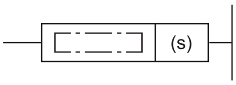
Switching a target block

BRSET

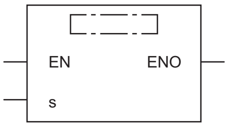
RnCPU RnENCPU RnPCPU (Process) ~~RnPCPU (Redundant)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction specifies an SFC control instruction target block No.

Ladder	ST
	ENO:=BRSET(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
BRSET	

Setting data

■Description, range, data type

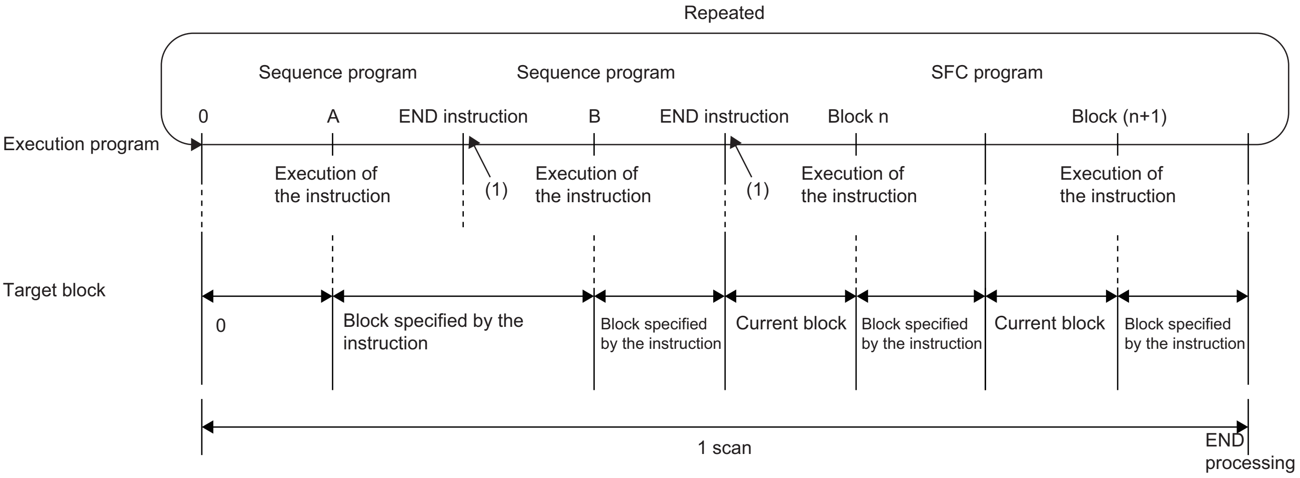
Operand	Description	Range	Data type	Data type (label)
(s)	Target block No.	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, S, 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)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- This instruction switches the SFC control instruction target block No. of the step (specified by S□) to the block No. specified by (s).
- The effective range of the instruction is as follows depending on the execution program type.
  - Sequence program: The block is switched from the point where the instruction is executed to the start of the SFC program. In the next scan, the target block will be the block 0 (default) until the instruction is executed again.
  - SFC program: The block is switched only within the step being executed. Even when the same step is targeted, the instruction must be executed for each block where the instruction specified by S□ is used. Moreover, even within a single step, the block is switched only from the point where the instruction is executed to the END processing of the step. In the next program, the target block will be the current block (default) until the instruction is executed again.



(1) END processing is not performed.

- If the block No. is specified by BL□\S□, the block No. is switched with or without execution of the instruction.
- The instruction is valid only for the target step. If multiple steps are active in such as a simultaneous sequence, the instruction needs to be executed for each step.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist, or the SFC program is in a standby state.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range, 0 to 319.

13.2 SFC Dedicated Instruction

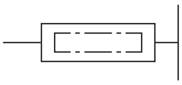
Creating a dummy transition condition

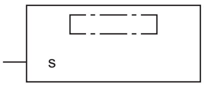
TRAN

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety) ✗

- The RnCPU and RnENCPU with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.015R" or later.)
- The RnPCPU (process) with firmware version "03" or later supports these instructions. (Use an engineering tool with version "1.020W" or later.)

This instruction is a dummy output which satisfies a transition condition.

Ladder	ST
	TRAN(s);

FBD/LD


Point

For details on transitions, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

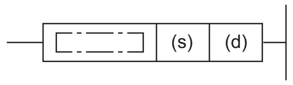
14 REDUNDANT SYSTEM INSTRUCTIONS

14.1 System Switching

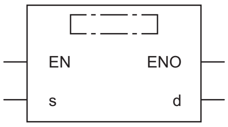
SP.CONTSW

✕RnCPU ✕RnENCPU ✕RnPCPU (Process) RnPCPU (Redundant) ✕RnSFCPU (Standard) ✕RnSFCPU (Safety)

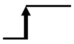
This instruction switches the systems (control system and standby system) during END processing of the scan where the instruction is executed.

Ladder	ST
	ENO:=SP_CONTSW(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.CONTSW	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Instruction ID number: A value used to identify a system switching request	-32768 to 32767	16-bit signed binary	ANY16
(d)	Error completion device: The device turns on when the system switching operation failed.	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

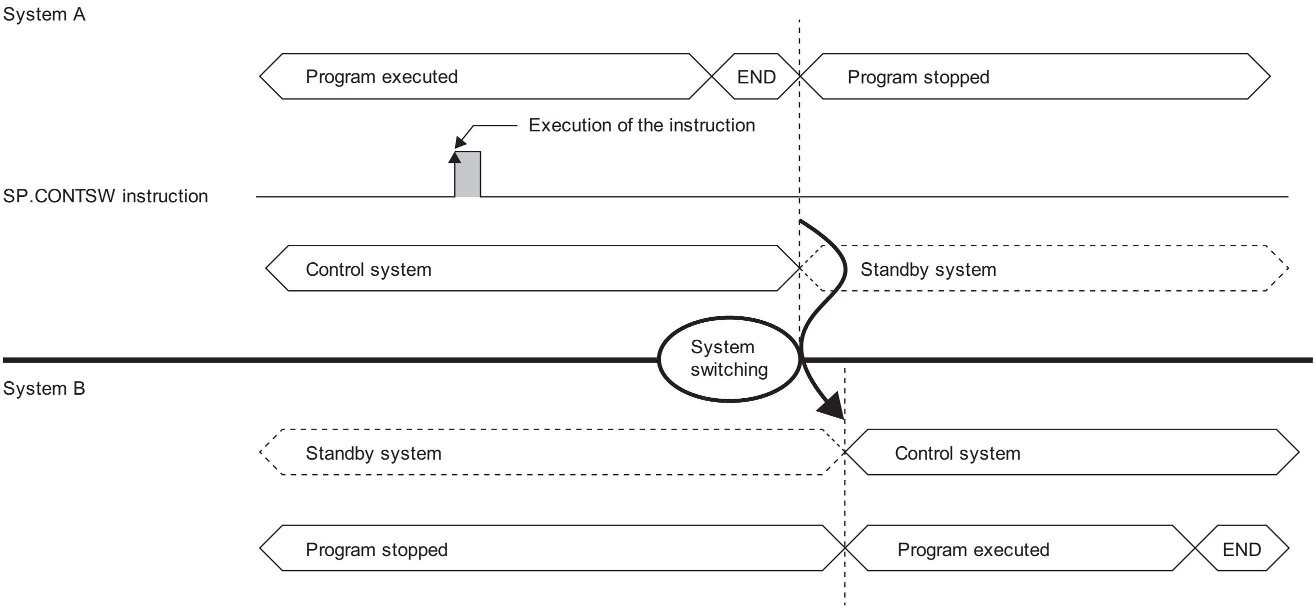
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction switches the systems (control system and standby system) during END processing of the scan where the instruction is executed. The instruction must be executed in the control system. If it is executed in the standby system, no processing is performed.

The following figure shows the system switching operation by using the instruction.

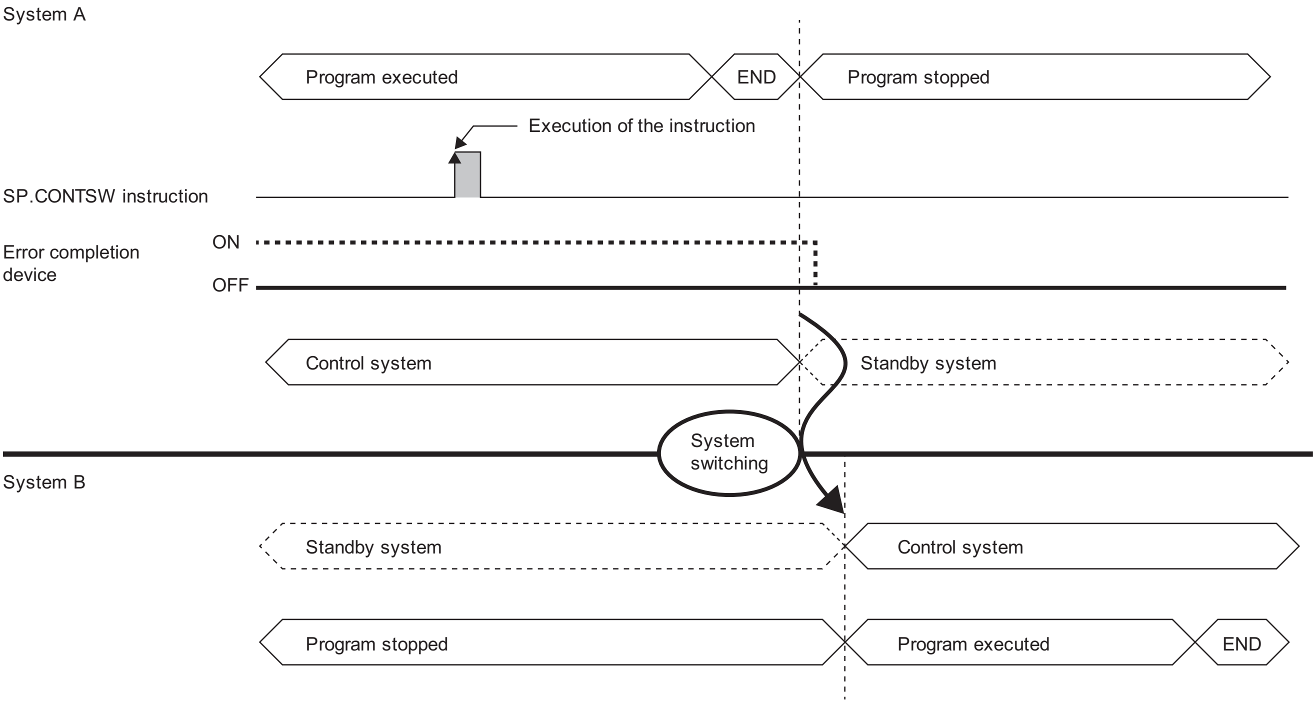


- To switch the systems by using the instruction, turn on SM1646 (System switching by a user) in advance. If the system switching operation is disabled by using the DCONTSW instruction, execute the ECONTSW instruction in the standby system before executing the SP.CONTSW instruction.
- The value specified by (s) is stored in SD1650 (System switching instruction ID number) of both CPU modules when the systems are switched successfully. When multiple SP.CONTSW instructions are executed in a program, the instruction used can be identified by reading data stored in SD1650. If two or more SP.CONTSW instructions are executed in one scan, only the data (argument) of the first instruction is stored.

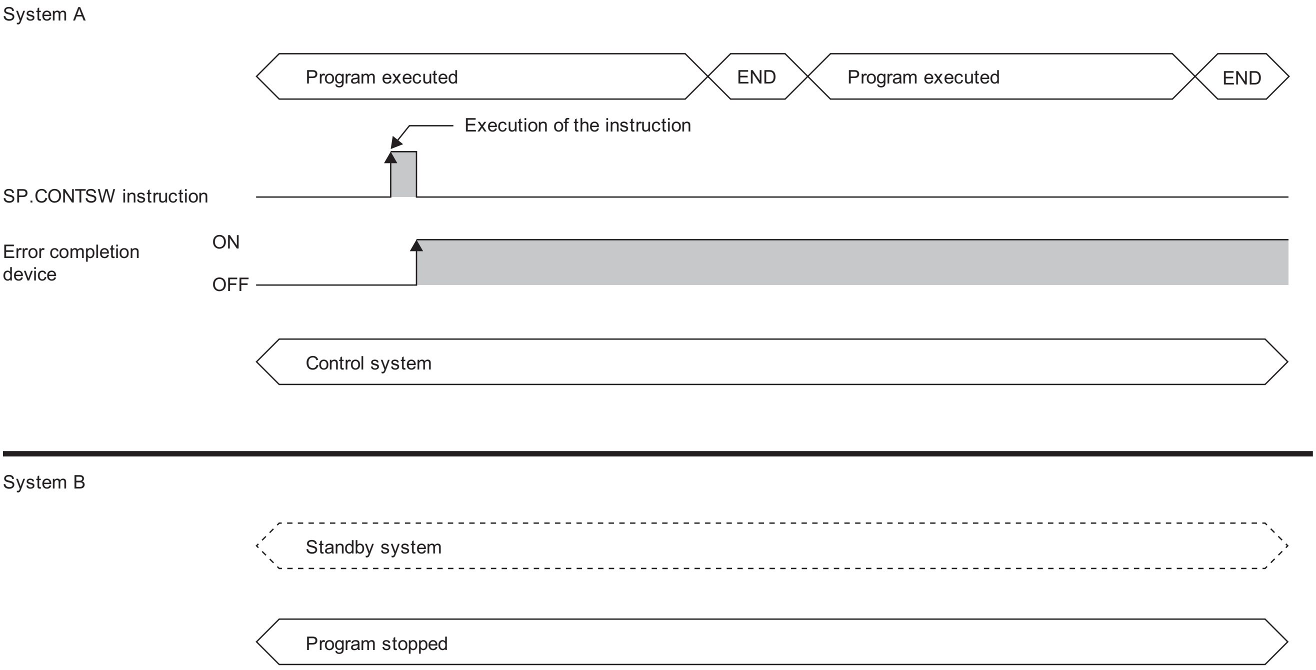
- The error completion device specified by (d) turns off when the systems were switched successfully, and turns on when the system switching operation failed. When there is a switching disable cause, 16 (System switching request by using the SP.CONTSW instruction) is stored in SD1643 of the CPU module in the control system. The corresponding disable cause number is stored in SD1644. For the disable cause number stored in SD1644, refer to the following.

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

- When the systems are switched by using the instruction



- When SM1646 is off at execution of the instruction



Precautions

- To execute the instruction, turn on (enabled) SM1646. If SM1646 is off (disabled), the systems cannot be switched.
- Even if SM1646 is turned off after execution of the instruction and before the END processing that performs the system switching operation, the systems can be switched.

Operation error

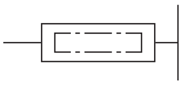
Error code (SD0)	Description
1BD0H	There is a switching disable cause and the systems are not switched during END processing.
1BD1H	When the instruction is executed, SM1646 is off (disabled).

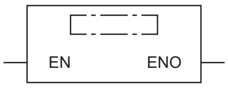
14.2 Disabling/Enabling System Switching

DCONTSW, ECONTSW

✗ RnCPU ✗ RnENCPU ✗ RnPCPU (Process) RnPCPU (Redundant) ✗ RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- DCONTSW: This instruction disables manual system switching.
- ECONTSW: This instruction enables manual system switching.

Ladder	ST
	ENO:=DCONTSW(EN); ENO:=ECONTSW(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
DCONTSW	
ECONTSW	

Processing details

- When the DCONTSW instruction is executed in the standby system, the system switching operation in the control system is disabled.
- When the ECONTSW instruction is executed in the standby system, the system switching operation in the control system is enabled.
- The system switching operation in the control system is enabled by default.
- These instructions can only be executed in the standby system. Even if these instructions are executed in the control system, no processing is performed.

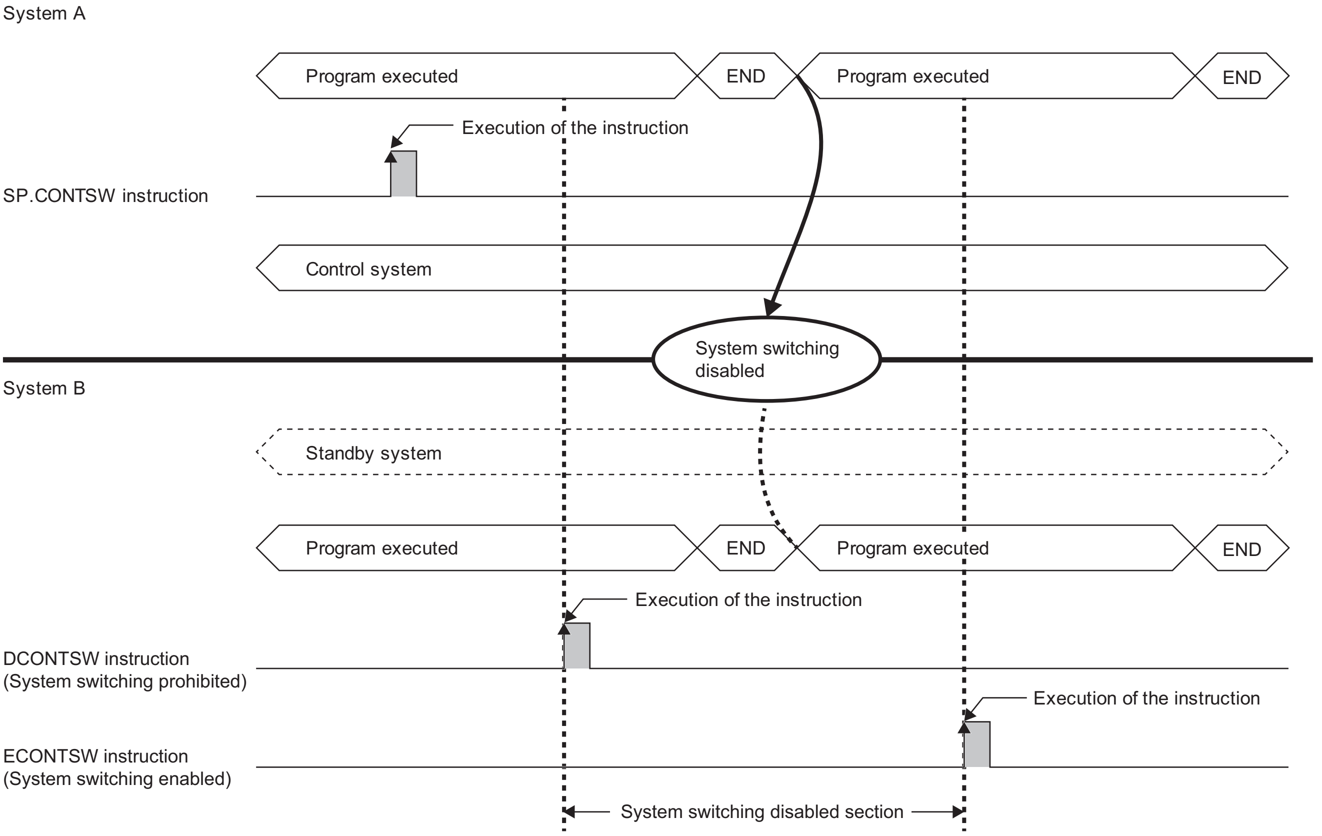
Point

To execute these instructions in the standby system in backup mode, write the instructions in a program executed in both systems or in a POU called by a program executed in both systems. For details on executing a program in both systems, refer to the following.

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

Ex.

When the SP.CONTSW instruction is executed while the system switching operation is disabled by the DCONTSW instruction



- The following operations enables system switching. To disable system switching, execute the DCONTSW instruction again.
  - Powering off and on the standby system
  - Resetting the CPU module in the standby system
  - Changing the operating status of the CPU module in the standby system from RUN to STOP
  - A stop error in the CPU module of the standby system
  - Switching the operation mode from backup mode to separate mode
  - Automatic system switching

Operation error

There is no operation error.

MEMO

PART 4

MODULE DEDICATED INSTRUCTIONS

This part consists of the following chapters.

15 NETWORK COMMON INSTRUCTIONS

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. When using MELSEC-Q series modules, refer to the manual for each module used and create programs.

For precautions when using modules, refer to the following.

📖 MELSEC iQ-R Module Configuration Manual

Availability of each module

The following table summarizes the availability of each module for the instructions explained in this chapter.

Instruction symbol	Availability			Reference
	Ethernet module	CC-Link IE Controller Network module	CC-Link IE Field Network module	
READ	○	○	○	Page 1454 JP.READ, GP.READ
SREAD	○	○	○	Page 1461 JP.SREAD, GP.SREAD
WRITE	○	○	○	Page 1469 JP.WRITE, GP.WRITE
SWRITE	○	○	○	Page 1477 JP.SWRITE, GP.SWRITE
SEND	○	○	○	Page 1485 JP.SEND, GP.SEND
RECV	○	○	○	Page 1492 JP.RECV, GP.RECV
RECVS	○	○	○	Page 1497 G.RECVS, Z.RECVS
J(P).ZNRD	○	○	×	Page 1501 J(P).ZNRD
J(P).ZNWR	○	○	×	Page 1506 J(P).ZNWR
REQ	○	○	○	Page 1511 J(P).REQ, G(P).REQ
RIRD	×	○	○	Page 1525 J(P).RIRD, G(P).RIRD
RIWT	×	○	○	Page 1530 J(P).RIWT, G(P).RIWT

Target networks and target station types

The network common instructions can access networks not only in the own station but also in other stations. (Excluding the RIRD and RIRW instructions)

The following table lists the target stations of individual instructions.

Instruction symbol	Target station (another station)	
	Target network	Target station type
READ SREAD	• Ethernet • CC-Link IE Controller Network • CC-Link IE Field Network • MELSECNET/H • MELSECNET/10	• RCPU • QCPU • LCPU • QSCPU • QnACPU • Intelligent device station
WRITE SWRITE	• Ethernet • CC-Link IE Controller Network • CC-Link IE Field Network • MELSECNET/H • MELSECNET/10	• RCPU • QCPU • LCPU • QnACPU • Intelligent device station
SEND RECV RECVS	• Ethernet • CC-Link IE Controller Network • CC-Link IE Field Network • MELSECNET/H • MELSECNET/10	• RCPU • QCPU • LCPU • QnACPU • Interface board for personal computer[1]
J(P).ZNRD J(P).ZNWR	• Ethernet • CC-Link IE Controller Network • MELSECNET/H • MELSECNET/10	• QCPU • LCPU • QnACPU • ACPU
REQ	• Ethernet • CC-Link IE Controller Network • CC-Link IE Field Network • MELSECNET/H • MELSECNET/10	• RCPU • QCPU • LCPU • QSCPU • QnACPU • Ethernet adapter module
RIRD RIWT	• CC-Link IE Controller Network • CC-Link IE Field Network[2]	• CC-Link IE Controller Network device • CC-Link IE Field Network device

[1] These instructions can access the following personal computer interface boards having the SEND/RECV functions.
CC-Link IE Field Network interface board, CC-Link IE Controller Network interface board, MELSECNET/H interface board, and MELSECNET/10 interface board

[2] These instructions cannot be executed from the local station to the intelligent device station. Execute them in the master station.

Range of available channel numbers

The range of own station channels that can be specified by the network common instructions varies depending on the module.

The following table summarizes the ranges of channels that can be specified by instructions for individual modules.

■Available channel numbers

Instruction symbol	RJ71GP21-SX	RJ71GF11-T2
READ, SREAD, WRITE, SWRITE	1 to 10	1, 2
SEND, RECV, RECVS, REQ	1 to 8	1, 2
RRUN, RSTOP, RTMRD, RTMWR	1 to 8	—
REMFR, REMTO, REMFRD, REMTOD	—	1 to 32*1

Instruction symbol	RJ71EN71*2								
	RJ71EN71(E+E)		RJ71EN71(E+CCIEC)		RJ71EN71(CCIEC)	RJ71EN71(E+CCIEF)		RJ71EN71 (CCIEF)	RJ71EN71 (Q)
	PORT1	PORT2	PORT1	PORT2	PORT1/PORT2	PORT1	PORT2	PORT1/ PORT2	PORT1
READ, SREAD, WRITE, SWRITE	1 to 8	11 to 18	1 to 8	11 to 18	1 to 8	1 to 8	11 to 18	1 to 8	1 to 8
SEND, RECV, RECVS, REQ	1 to 8	11 to 18	1 to 8	11 to 18	1 to 8	1 to 8	11 to 18	1 to 8	1 to 8
RRUN, RSTOP, RTMRD, RTMWR	—	—	—	11 to 18	1 to 8	—	—	—	—
REMFR, REMTO, REMFRD, REMTOD	—	—	—	—	—	—	1 to 32*1	1 to 32*1	—

Instruction symbol	RnENCPU (network part)*3					
	_RJ71EN71(E+CCIEC)		_RJ71EN71(CCIEC)	_RJ71EN71(E+CCIEF)		_RJ71EN71(CCIEF)
	PORT1	PORT2	PORT1/PORT2	PORT1	PORT2	PORT1/PORT2
READ, SREAD, WRITE, SWRITE	1 to 8	11 to 18	1 to 8	1 to 8	11 to 18	1 to 8
SEND, RECV, RECVS, REQ	1 to 8	11 to 18	1 to 8	1 to 8	11 to 18	1 to 8
RRUN, RSTOP, RTMRD, RTMWR	—	11 to 18	1 to 8	—	—	—
REMFR, REMTO, REMFRD, REMTOD	—	—	—	—	1 to 32*1	1 to 32*1

*1 The own station channel numbers specified by REMFR, REMTO, REMFRD, and REMTOD instructions can be used in combination with the own station channel numbers specified by link dedicated instructions.

*2 The range of channel numbers of an Ethernet module varies depending on the combination of network types at PORT1 and PORT2.

- RJ71EN71(E+E): PORT1 and PORT2 are "Ethernet".
- RJ71EN71(E+CCIEC): PORT1 is "Ethernet" and PORT2 is "CC-Link IE Controller Network".
- RJ71EN71(CCIEC): PORT1 and PORT2 are "CC-Link IE Controller Network".
- RJ71EN71(E+CCIEF): PORT1 is "Ethernet" and PORT2 is "CC-Link IE Field Network".
- RJ71EN71(CCIEF): PORT1 and PORT2 are "CC-Link IE Field Network".
- RJ71EN71(Q): PORT1 is "Q-compatible Ethernet". (PORT2 cannot be used.)

*3 The range of channel numbers of the RnENCPU (network part) varies depending on the combination of network types at PORT1 and PORT2.

- _RJ71EN71(E+CCIEC): PORT1 is "Ethernet" and PORT2 is "CC-Link IE Controller Network".
- _RJ71EN71(CCIEC): PORT1 and PORT2 are "CC-Link IE Controller Network".
- _RJ71EN71(E+CCIEF): PORT1 is "Ethernet" and PORT2 is "CC-Link IE Field Network".
- _RJ71EN71(CCIEF): PORT1 and PORT2 are "CC-Link IE Field Network".

■Corresponding Network type

Instruction symbol	CC-Link IE Controller Network	CC-Link IE Field Network	Ethernet	Q-compatible Ethernet
READ, SREAD, WRITE, SWRITE	○	○	○	○
SEND, RECV, RECVS, REQ	○	○	○	○
RRUN, RSTOP, RTMRD, RTMWR	○	—	—	—
REMFR, REMTO, REMFRD, REMTOD	—	○	—	—

Specifications of character string data specified by link dedicated instructions

Some operands specified in link dedicated instructions are specified in character strings. The following table summarizes the specifications of character string data specified.

Instruction symbol	Applicable operand	Specifications of character string data
READ	(s2)	■Specification method Specify a character string by enclosing it in single quotes (') or double quotes ("). ■Specifiable device types • Word device: D, W, SW, SD, T, C, ST, ZR, R • Digit-specified bit device: X, Y, M, L, B, F, SB, SM • Digit-specified word device and bit device index modification: Z, ZZ ■ASCII string specifications • An ASCII string can consist of a maximum of 32 characters. (Specify NULL as the termination character.) • ASCII strings are zero-suppressed. (Example: D1 and D001 are handled the same.) • ASCII strings are case-sensitive. ■Precautions Digit-specified bit devices can be specified only when the following conditions are all satisfied. • The device number is a multiple of 16 (10H). • The number of specified digits is 4 (K4). An instruction specifying bit devices by digits can be executed only when the target station is RCPU, basic model QCPU, universal model QCPU, or LCPU. If an instruction uses a digit specification for other types of CPU modules, an abnormal response is returned from the target station and the instruction is completed with an error. (The network module does not detect an error during execution of the instruction.) The target device of an instruction executed in index modification mode is determined by the value of the index register of the own station.
SREAD	(s2)	
WRITE	(d1)	
SWRITE	(d1)	
SREAD	(d3)	■Specification method Specify a character string by enclosing it in single quotes (') or double quotes ("). ■Specifiable device types • Bit specification of word device: D, W, SW, SD, ZR, R • Bit device: X, Y, M, L, B, F, SB, SM • Bit device index modification: Z, ZZ ■ASCII string specifications • An ASCII string can consist of a maximum of 32 characters. (Specify NULL as the termination character.) • ASCII strings are zero-suppressed. (Example: D1 and D001 are handled the same.) • ASCII strings are case-sensitive. ■Precautions The target device of an instruction executed in index modification mode is determined by the value of the index register of the own station.
SWRITE	(d3)	
J(P).ZNRD	(s2)	■Specification method Specify a character string by enclosing it in single quotes (') or double quotes ("). ■Specifiable device types • Word device: T, C, D, W • Word device index modification: Z, ZZ ■ASCII string specifications • An ASCII string can consist of a maximum of 32 characters. (Specify NULL as the termination character.) • ASCII strings are zero-suppressed. (Example: D1 and D001 are handled the same.) • ASCII strings are case-sensitive. ■Precautions The target device of an instruction executed in index modification mode is determined by the value of the index register of the own station.
J(P).ZNWR	(d1)	

Calculating the value to be set for the arrival monitoring time

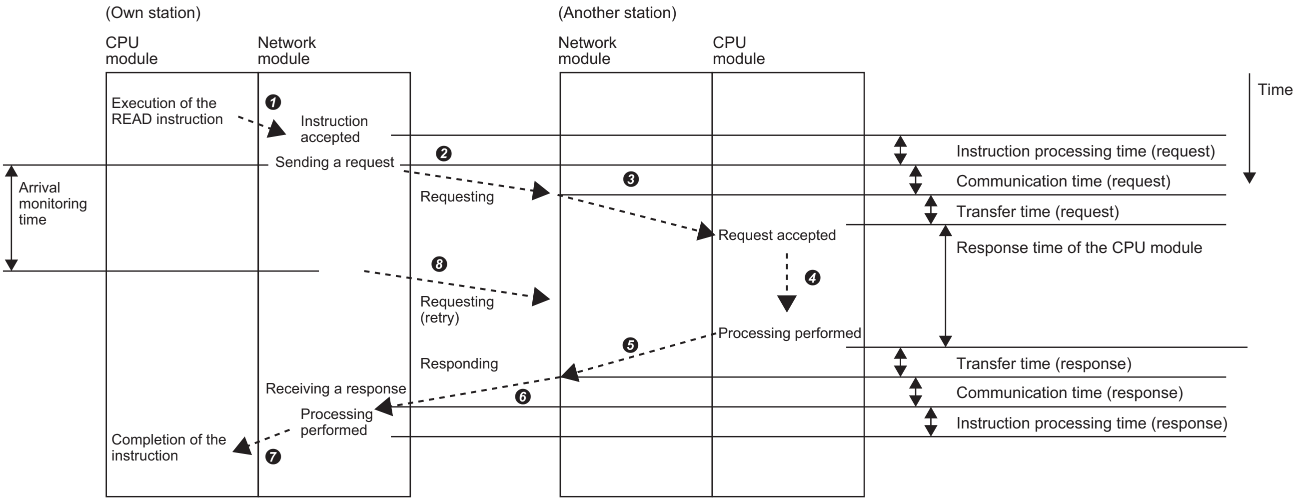
The following shows the method for determining the value to be set for the arrival monitoring time specified by link dedicated instructions.

■Outline of communication processing for link dedicated instructions

When a link dedicated instruction is executed in the same network, the communication process at occurrence of retransmission is as follows.

Ex.

The READ instruction is executed in the same network.



❶ The CPU module executes the instruction and the network module accepts the instruction.
❷ The own station network module sends the request to another station.
❸ Another station network module receives the request and relays it to the CPU module.
❹ The CPU module processes the request.
❺ Another station network module relays the response from the CPU module.
❻ Another station network module sends the response to the own station.
❼ The own station network module receives the response and processes it.
❽ The own station network module requests retransmission if the response does not arrive within the arrival monitoring time.

■Calculating the arrival monitoring time

The value to be set for the arrival monitoring time must be greater than the time taken from ❷ to ❼.

[Calculation formula]

Arrival monitoring time>Tc_req+Tt_req+Tcpu+Tt_ans+Tc_ans+Tnet

Item	Description	Details
Tc_req	Communication time (request)	In the case of the CC-Link IE Controller Network or CC-Link IE Field Network, calculate based on the link scan time. For concurrent multiple transmissions including other transient transmissions, further increase the time in units of link scans. The link scan time can be checked with the diagnostic window of each network. When Ethernet is used, the time varies depending on the amount of data and network load ratio (line congestion). Calculate the time by adding the delay of the switching hub and the line congestion to 1ms.
Tc_ans	Communication time (response)	
Tt_req	Network module relay time (request)	Set 20ms in total.
Tt_ans	Network module relay time (response)	
Tnet	Network module instruction processing time (response)	
Tcpu	CPU module response time	This time varies depending on the CPU module type and operating status. Calculate the response time by "sequence scan time" × "number of times the device/label access service is performed per scan". For the scan time and device/label access service processing, refer to the following. 📖 MELSEC iQ-R CPU Module User's Manual (Application)

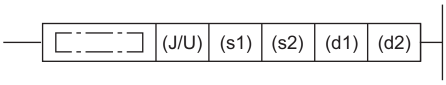
15.1 Link Dedicated Instructions

Reading data from the programmable controller on another station

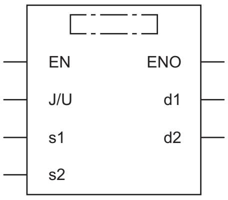
JP.READ, GP.READ

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These instructions read data from a device in the programmable controller of another station. (in units of words)

Ladder	ST
	ENO:=JP_READ(EN,J,s1,s2,d1,d2); ENO:=GP_READ(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.READ GP.READ	↑

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.READ	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.READ	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(s2)		Target station start device where the data to be read is stored	—	String[*2]	ANYSTRING_SINGLE[*2]
(d1)		Own station start device (a continuous area for the length of the read data) for storing the data that has been read	—	Device name	ANY16[*1]
(d2)		Device of the own station, which 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] When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

[*2] For the specifications of the string data to be specified, refer to the following.
☞ Page 1452 Specifications of character string data specified by link dedicated instructions

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	JP.READ	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.READ	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	—	—	—	—	—	—	—	—	○	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

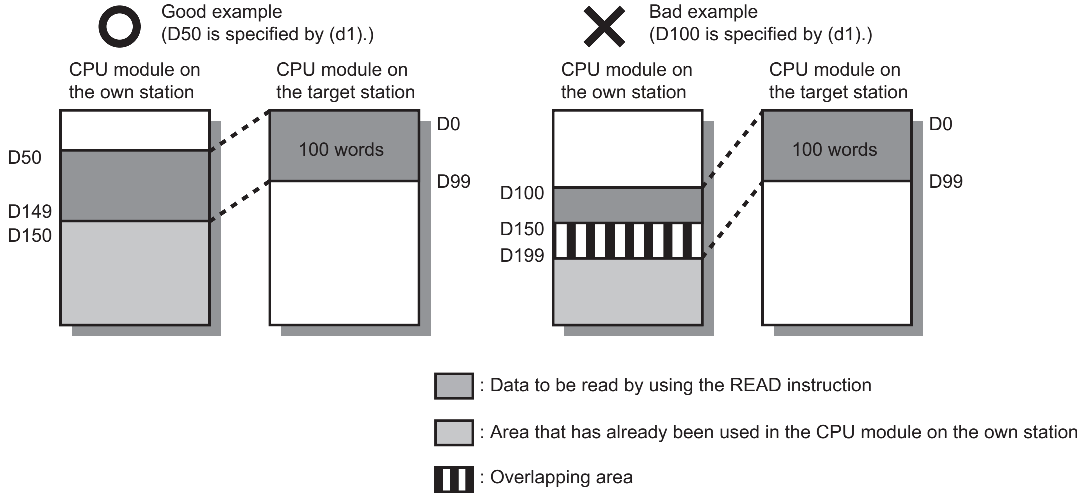
*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Point

- The READ instruction cannot be executed when the target station is ACPU.
- Specify the own station start device (d1) for storing the data that has been read, by considering the range in which the data that has been read can be stored.

(Example: When areas D150 and after in the own station CPU module are already in use)

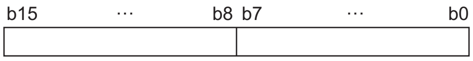


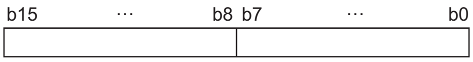
■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Error completion type	b15 … b8 b7 … b0 (3) 0 (2) (1) 0 (1) Error completion type (bit 7) Specify whether to set data at completion with an error. • 0: Do not set data in (s1)+12 and later at completion with an error. • 1: Set data in (s1)+12 and later at completion with an error. (2) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units (3) Target station address specification method (b15) • 0: Specify "network No." in (s1)+4, and "station number" in (s1)+5. • 1: Specify "IP address" in (s1)+4, 5 (Ethernet only)	0000H 0080H 0100H 0180H 8000H 8080H 8100H 8180H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User
+4	Target network number	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the network number (1 to 239) of the target station. [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the network number (1 to 239) of the target station. ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (third and fourth octets) of the target station. b15 … b8 b7 … b0 • b8 to b15: 3rd octet • b0 to b7: 4th octet	■(s1)+4 Network No.: 1 to 239 ■(s1)+5 Station No.: 1 to 120, 125, 126 ■(s1)+4, 5 IP address: 00000001H to FFFFFFFEH (1 to 4294967294)	User
+5	Target station number	[CC-Link IE Controller Network] Specify the station number (1 to 120) of the target station. [CC-Link IE Field Network] Specify the station number of the target station. • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the station number (1 to 120) of the target station. ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (first and second octets) of the target station. b15 … b8 b7 … b0 • b8 to b15: 1st octet • b0 to b7: 2nd octet		User
+6	Not used	—	—	—

Operand: (s1)

Device	Item	Description	Setting range	Set by
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 Specify the TCP resend timer value or greater for the monitoring time till completion of processing. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
+9	Read data length	Specify the number of words to be read. ■When reading data from RCPU, QCPU, or LCPU • Channels 1 to 8 are used: 1 to 960 (words) • Channels 9 and 10 are used: 1 to 8192 (words) ■When reading data from QnACPU • 1 to 480 (words)	1 to 8192	User
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8bits: Year (00H to 99H: Upper two digits of the year)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	■When "0" is specified in bit 15 of (s1)+0 The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number) ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) The IP address (third and fourth octets) of the station where an error was detected is stored.  • b8 to b15: 3rd octet • b0 to b7: 4th octet	—	System

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+17	Error-detected station number	■When "0" is specified in bit 15 of (s1)+0 The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) The IP address (first and second octets) of the station where an error was detected is stored.  • b8 to b15: 1st octet • b0 to b7: 2nd octet	—	System

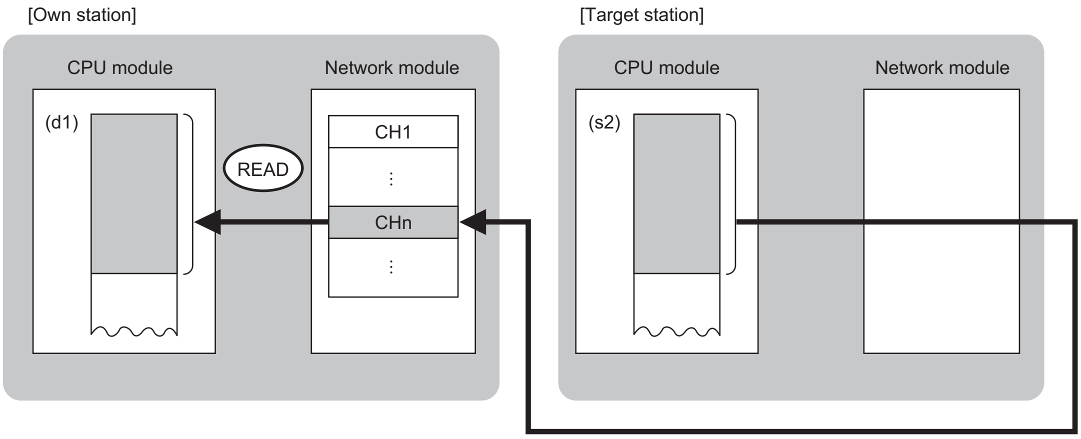
Point

- The continuous area (a maximum of 8192 words) specified by (s1)+9 is required in the read data storage device (d1).
- The number of resends (s1)+7 must be set every time the instruction is executed.

Processing details

- These instructions read the data from the specified word device in the target station specified by the target network number and target station number of the control data or the target station specified by the IP address. Upon completion of reading the device data, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When "network number" and "station number" are specified ("0" is specified in bit 15 of (s1)+0) by the target station address specification method, device data can be read also from stations connected to networks other than the stations connected to the own station network. (If "IP address" is specified ("1" is specified in bit 15 of (s1)+0), device data cannot be read from stations connected via a relay station.)
- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the READ instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the READ instruction completes, and turns off during the next END processing.

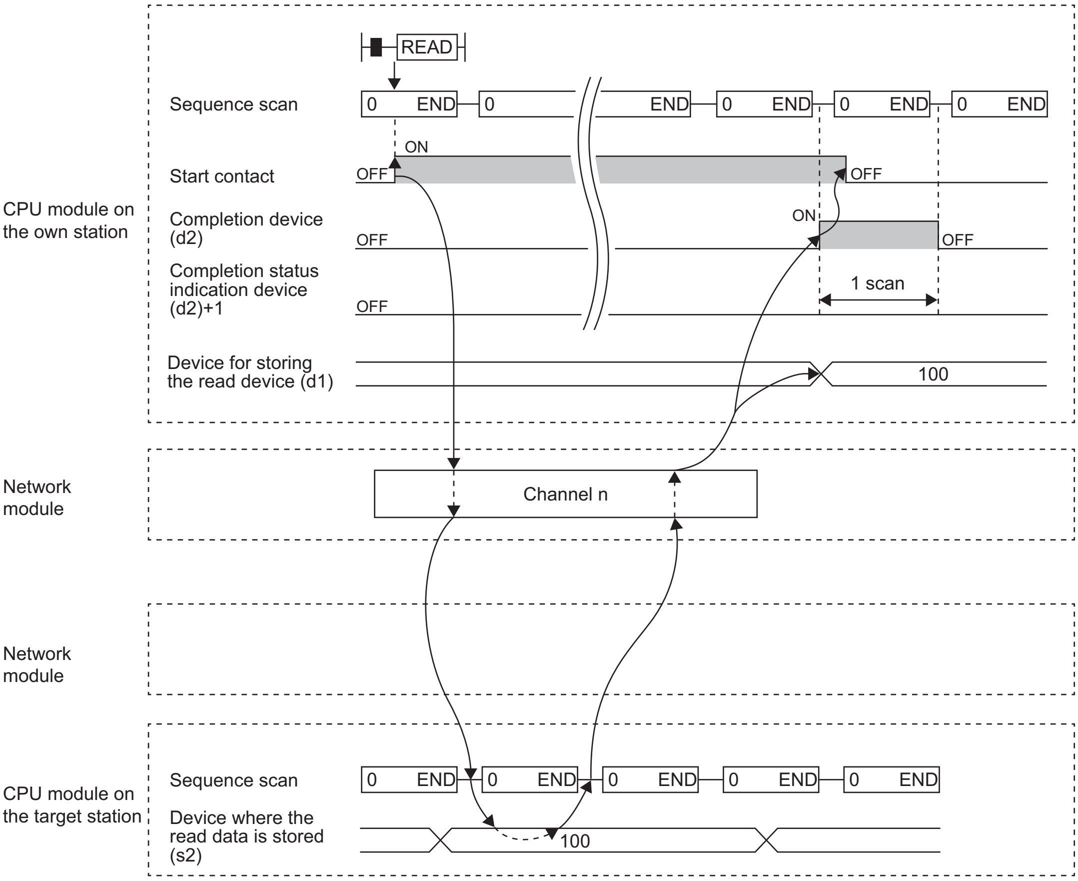
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the READ instruction.

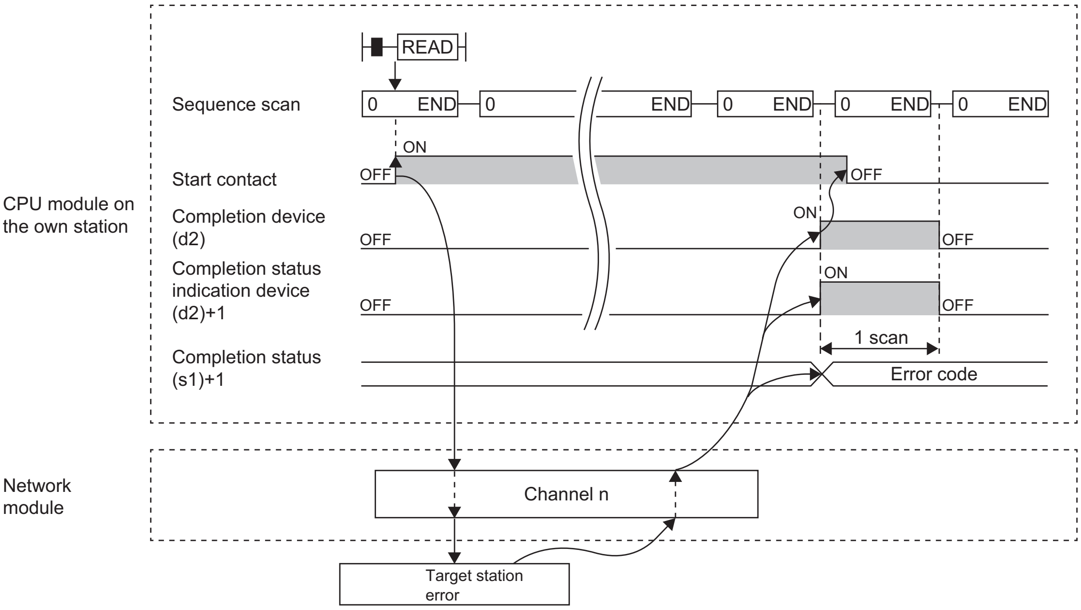
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the READ instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the READ instruction.
  - When completed successfully



  - When completed with an error



- Read processing is performed only once on the rising edge when the read command turns on.

Operation error

Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

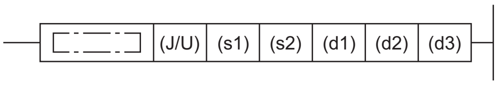
Reading data from the programmable controller on another station (with notification)

JP.SREAD, GP.SREAD

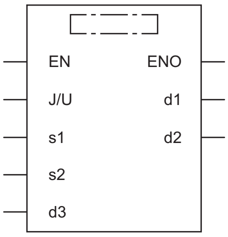
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions read data from a device in the programmable controller of another station. (in units of words)

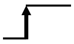
After the data reading is completed by the SREAD instruction, the device of another station is turned on. The other station can recognize that data has been read by the SREAD instruction.

Ladder	ST
	ENO:=JP_SREAD(EN,J,s1,s2,d1,d2,d3); ENO:=GP_SREAD(EN,U,s1,s2,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.SREAD GP.SREAD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.SREAD	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.SREAD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(s2)		Target station start device where the data to be read is stored	—	String[*2]	ANYSTRING_SINGLE[*2]
(d1)		Own station start device for storing the read data	—	Device name	ANY16[*1]
(d2)		Device of the own station, which 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)
(d3)		Device of the target station, which turns on for one scan upon completion of the instruction. (The target station can recognize that data has been read from another station.)	—	String[*2]	ANYSTRING_SINGLE[*2]
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

*2 For the specifications of the string data to be specified, refer to the following.

☞ Page 1452 Specifications of character string data specified by link dedicated instructions

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	JP.SREAD	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.SREAD	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○[2]	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	—	—	—	—	—	—	—	—	○	—	—
(d1)		—	—	○[2]	—	—	—	—	○	—	—	—	—	—
(d2)		○[1]	—	○[3]	—	—	—	—	—	—	—	—	—	—
(d3)		—	—	—	—	—	—	—	—	—	—	○	—	—

[1] FX and FY cannot be used.

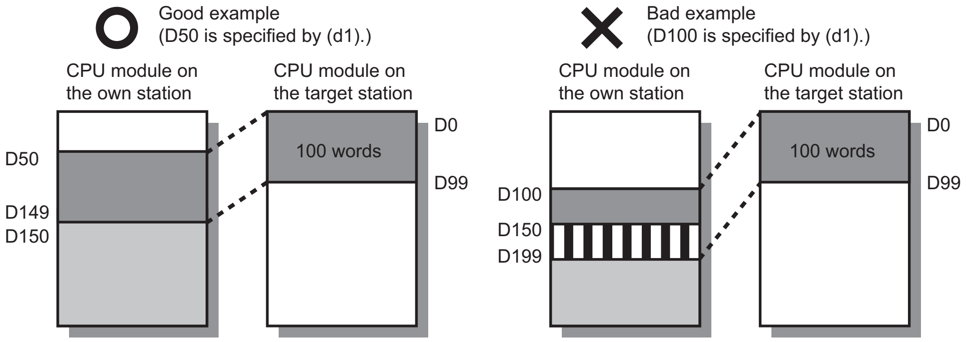
[2] FD cannot be used.

[3] T, ST, C, and FD cannot be used.

Point

- The SREAD instruction cannot be executed when the target station is ACPU.
- Specify the own station start device (d1) for storing the data that has been read, by considering the range in which the data that has been read can be stored.

(Example: When areas D150 and after in the own station CPU module are already in use)



■ : Data to be read by using the SREAD instruction

□ : Area that has already been used in the CPU module on the own station

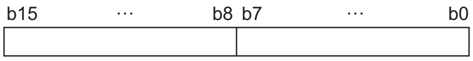
▥ : Overlapping area

- When the target station for which the SREAD instruction is executed is the Basic model QCPU, the read notification device for the target station specified by argument (d3) is ignored. The operations of the SREAD instruction are the same as those of the READ instruction.
- The SREAD instruction can be programmed by omitting argument (d3). However, the operations are the same as those of the READ instruction. The operations of the SREAD instruction can be selected according to whether (d3) is omitted or not.

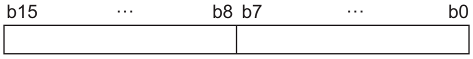
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Error completion type	b15 ··· b8 b7 ··· b0 (3) 0 (2) (1) 0 (1) Error completion type (bit 7) Specify whether to set data at completion with an error. • 0: Do not set data in (s1)+12 and later at completion with an error. • 1: Set data in (s1)+12 and later at completion with an error. (2) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units (3) Target station address specification method (b15) • 0: Specify "network No." in (s1)+4, and "station number" in (s1)+5. • 1: Specify "IP address" in (s1)+4, 5 (Ethernet only)	0000H 0080H 0100H 0180H 8000H 8080H 8100H 8180H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User
+4	Target network number	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the network number (1 to 239) of the target station. [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the network number (1 to 239) of the target station. ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (third and fourth octets) of the target station. b15 ··· b8 b7 ··· b0 • b8 to b15: 3rd octet • b0 to b7: 4th octet	■(s1)+4 Network No.: 1 to 239 ■(s1)+5 Station No.: 1 to 120, 125, 126 ■(s1)+4, 5 IP address: 00000001H to FFFFFFFEH (1 to 4294967294)	User
+5	Target station number	[CC-Link IE Controller Network] Specify the station number (1 to 120) of the target station. [CC-Link IE Field Network] Specify the station number of the target station. • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the station number (1 to 120) of the target station. ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (first and second octets) of the target station. b15 ··· b8 b7 ··· b0 • b8 to b15: 1st octet • b0 to b7: 2nd octet		User
+6	Not used	—	—	—

Operand: (s1)

Device	Item	Description	Setting range	Set by
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 Specify the TCP resend timer value or greater for the monitoring time till completion of processing. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
+9	Read data length	Specify the number of words to be read. ■When reading data from RCPU, QCPU, or LCPU • Channels 1 to 8 are used: 1 to 960 (words) • Channels 9 and 10 are used: 1 to 8192 (words) ■When reading data from QnACPU • 1 to 480 (words)	1 to 8192	User
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8bits: Year (00H to 99H: Upper two digits of the year)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	■When "0" is specified in bit 15 of (s1)+0 The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number) ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) The IP address (third and fourth octets) of the station where an error was detected is stored.  • b8 to b15: 3rd octet • b0 to b7: 4th octet	—	System

Operand: (s1)

Device	Item	Description	Setting range	Set by
+17	Error-detected station number	■When "0" is specified in bit 15 of (s1)+0 The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) The IP address (first and second octets) of the station where an error was detected is stored.  • b8 to b15: 1st octet • b0 to b7: 2nd octet	—	System

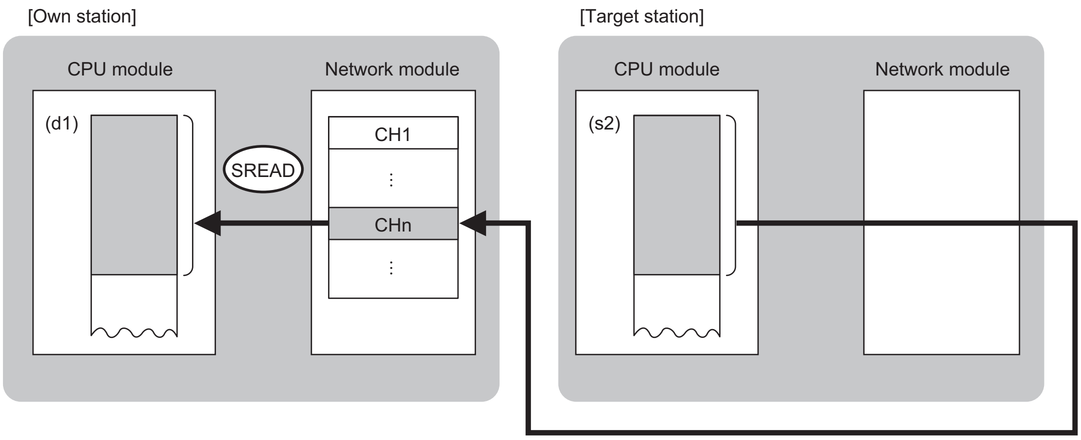
Point

- The continuous area (a maximum of 8192 words) specified by (s1)+9 is required in the read data storage device (d1).
- The number of resends (s1)+7 must be set every time the instruction is executed.

Processing details

- These instructions read the data from the specified word device in the target station specified by the target network number and target station number of the control data or the target station specified by the IP address. Upon completion of reading the device data, the completion device specified by (d2) turns on. In another station, upon completion of sending the device data specified by (s2), the device specified by (d3) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When "network number" and "station number" are specified ("0" is specified in bit 15 of (s1)+0) by the target station address specification method, device data can be read also from stations connected to networks other than the stations connected to the own station network. (If "IP address" is specified ("1" is specified in bit 15 of (s1)+0), device data cannot be read from stations connected via a relay station.)
- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the SREAD instruction can be checked with the send/receive instruction flag, the completion device (d2), and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the SREAD instruction completes, and turns off during the next END processing.

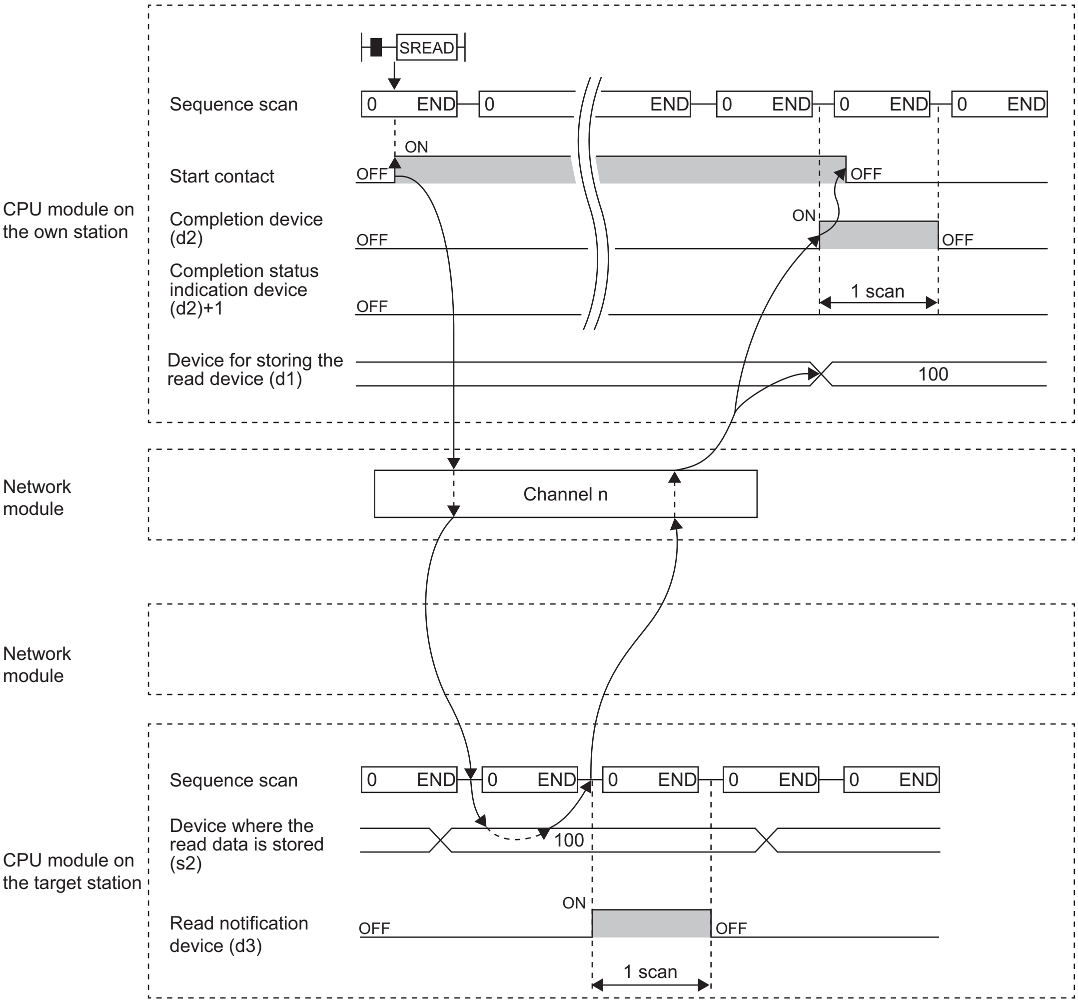
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the SREAD instruction.

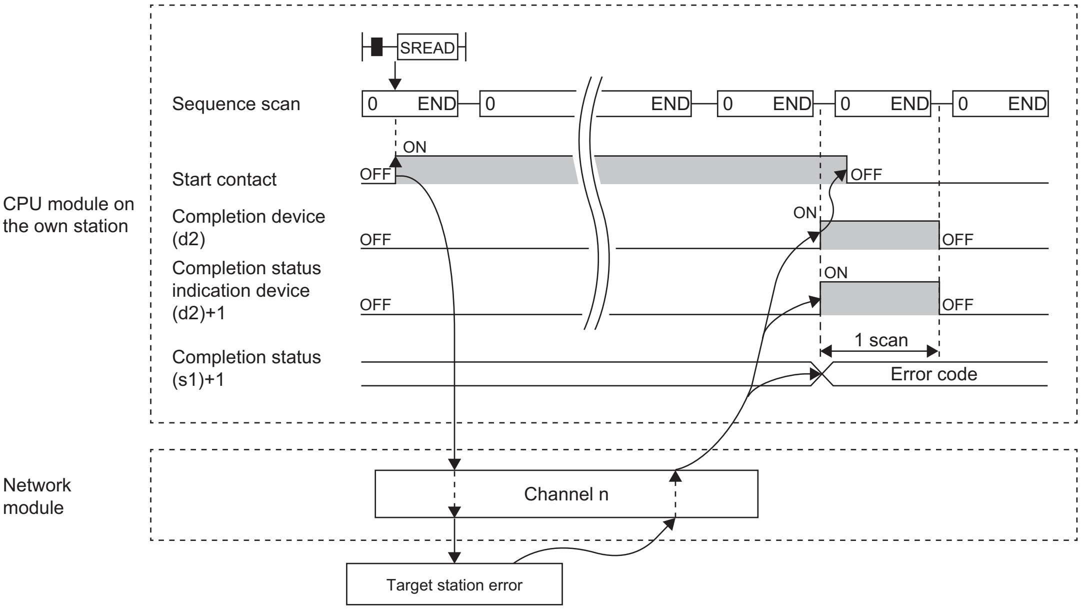
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SREAD instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SREAD instruction.
  - When completed successfully



  - When completed with an error



- Read processing is performed only once on the rising edge when the read command turns on.

Operation error

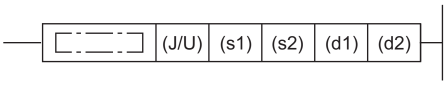
Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Writing data to the programmable controller on another station

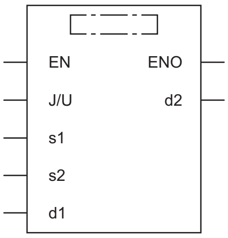
JP.WRITE, GP.WRITE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

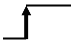
These instructions write data to a device in the programmable controller of another station (in units of words).

Ladder	ST
	ENO:=JP_WRITE(EN,J,s1,s2,d1,d2); ENO:=GP_WRITE(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.WRITE GP.WRITE	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.WRITE	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.WRITE	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16*1
(s2)		Own station start device containing write data	—	Device name	ANY16*1
(d1)		Target station start device to which data is to be written	—	String*2	ANYSTRING_SINGLE*2
(d2)		Device of the own station, which 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 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

*2 For the specifications of the string data to be specified, refer to the following.
☞ Page 1452 Specifications of character string data specified by link dedicated instructions

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	JP.WRITE	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.WRITE	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○[2]	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○[2]	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	—	—	—	—	—	—	—	—	○	—	—
(d2)		○[1]	—	○[3]	—	—	—	—	—	—	—	—	—	—

[1] FX and FY cannot be used.

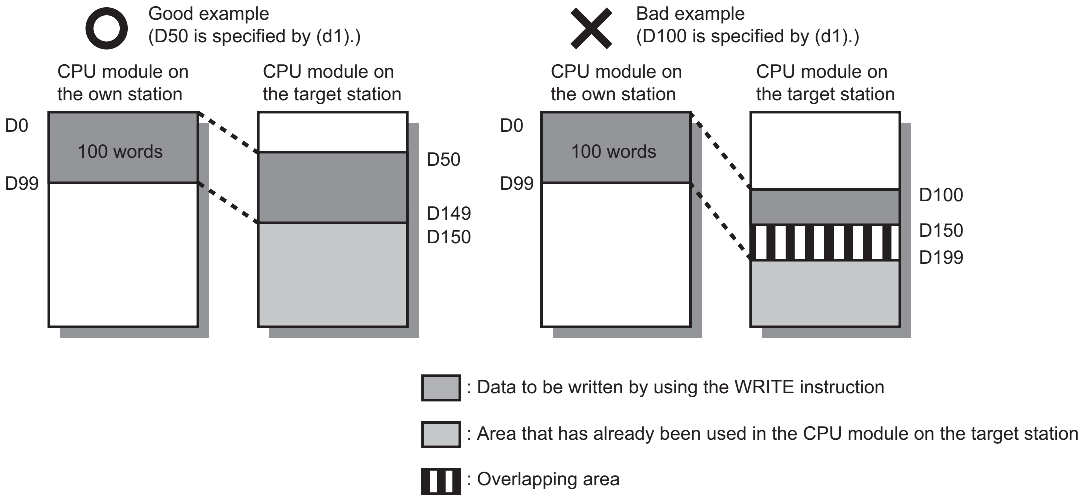
[2] FD cannot be used.

[3] T, ST, C, and FD cannot be used.

Point

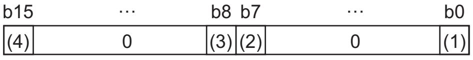
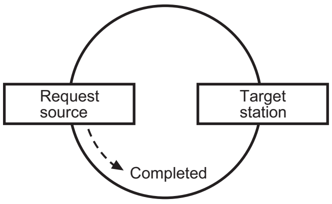
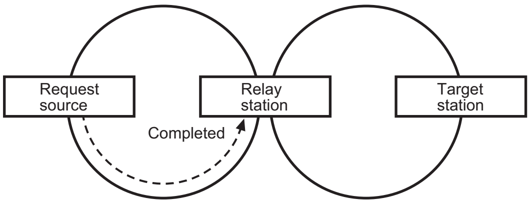
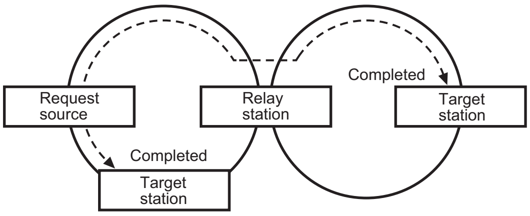
- The WRITE instruction cannot be executed when the target station is ACPU.
- Specify the target station start device (d1), to which data is to be written, by considering the range in which the data that has been written can be stored.

(Example: When areas D150 and after in the target station CPU module are already in use)



■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Execution/error completion type	 (1) Execution type (b0) • 0: No arrival acknowledgment When the target station is in the own network: Completed when data has been sent from the own station.  When the target station is in another network: Completed when data has arrived the relay station of the own network.  • 1: Arrival acknowledgment used Sending data is completed when the data is written to the target station.  (2) Error completion type (bit 7) Specify whether to set clock data when completed with an error. • 0: Do not set data in (s1)+11 and later at completion with an error. • 1: Set data in (s1)+11 and later at completion with an error. (3) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units (4) Target station address specification method (b15) • 0: Specify "network No." in (s1)+4, and "station number" in (s1)+5. • 1: Specify the IP address in (s1)+4, 5 (Ethernet only)	0000H 0001H 0080H 0081H 0100H 0101H 0180H 0181H 8000H 8001H 8080H 8081H 8100H 8101H 8180H 8181H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+4	Target network number	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the network number (1 to 239) of the target station. [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the network number (1 to 239) of the target station. ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) Specify the IP address (third and fourth octets) of the target station. b15 ··· b8 b7 ··· b0 • b8 to b15: 3rd octet • b0 to b7: 4th octet	■(s1)+4 Network No.: 1 to 239 ■(s1)+5 Station No.: 1 to 120, 125, 126 Group number: 0081H to 00A0H All-station specification: 00FFH ■(s1)+4, 5 IP address: 00000001H to FFFFFFFEH (1 to 4294967294)	User
+5	Target station number	[CC-Link IE Controller Network] Specify the station number of the target station. (1) Station number specification • 1 to 120: Station number (2) Group number specification • 0081H to 00A0H: All stations with group numbers 0001H to 0020H (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") (3) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") [CC-Link IE Field Network] Specify the station number of the target station. (1) Station number specification • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station (2) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") [Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the station number of the target station. (1) Station number specification • 1 to 120: Station number (2) Group number specification • 0081H to 00A0H: All stations with group numbers 0001H to 0020H (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") (3) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (first and second octets) of the target station. b15 ··· b8 b7 ··· b0 • b8 to b15: 1st octet • b0 to b7: 2nd octet		User
+6	Not used	—	—	—
+7	Number of resends	Effective when the execution type specified by (s1) is "1: Arrival acknowledgment used". ■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 Specify the TCP resend timer value or greater for the monitoring time till completion of processing. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
+9	Write data length	Specify the number of words to be written. ■Writing to RCPU, QCPU, or LCPU • Channels 1 to 8 are used: 1 to 960 (words) • Channels 9 and 10 are used: 1 to 8192 (words) ■Writing to QnACPU • 1 to 480 (words)	1 to 8192	User
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

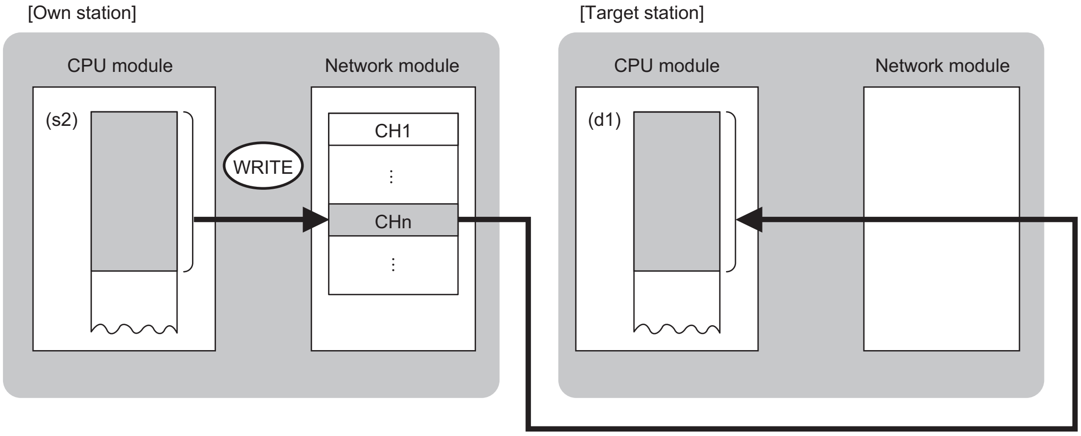
Point

- The continuous area (a maximum of 8192 words) for the write data length ((s1)+9) is required in the write data storage device (d1).
- When a number from 1 to 120 is specified for the target station number, the WRITE instruction should be executed with the execution type set to "Arrival acknowledgment used". This is to improve data reliability. When a number from 81H to A0H or FFH is specified for the target station number, the WRITE instruction should be executed with the execution type set to "No arrival acknowledgment".
- When performing device writing to the same station from multiple stations, do not overlap the write timing. When the execution type is set to "No arrival acknowledgment", successful completion results in the write source station if communication is completed successfully even when the send data contains an error. Also, even when the send data is normal, a timeout results in the write source station if the WRITE instructions are executed for the same station from multiple stations.
- The number of resends (s1)+7 must be set every time the WRITE instruction is executed.

Processing details

- These instructions write the data in the device/label specified by (s2) in the own station to the word device in the target station specified by the target network number and target station number of the control data or the target station specified by the IP address. Upon completion of writing device data to another station number, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When "network number" and "station number" are specified ("0" is specified in bit 15 of (s1)+0) by the target station address specification method, device data can be written also to the stations connected to networks other than the stations connected to the own station network. (If "IP address" is specified ("1" is specified in bit 15 of (s1)+0), device data cannot be read from stations connected via a relay station.)
- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the WRITE instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the WRITE instruction completes, and turns off during the next END processing.

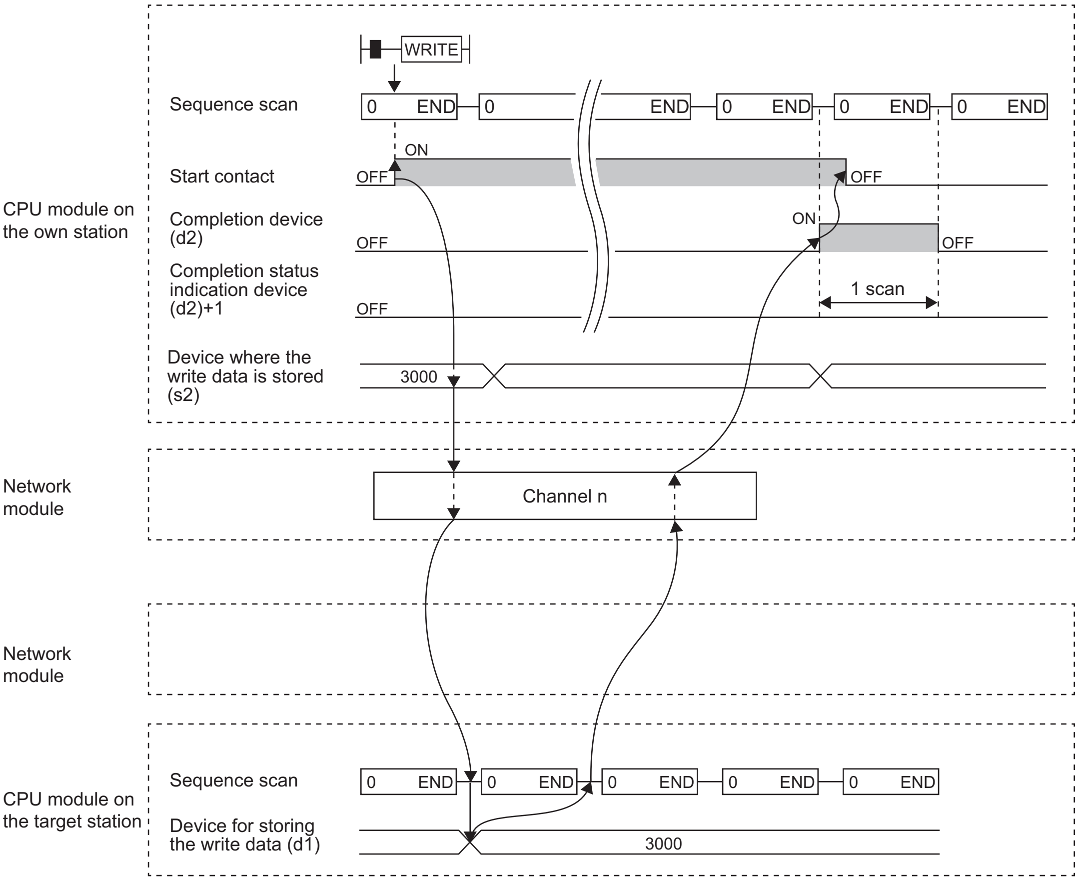
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the WRITE instruction.

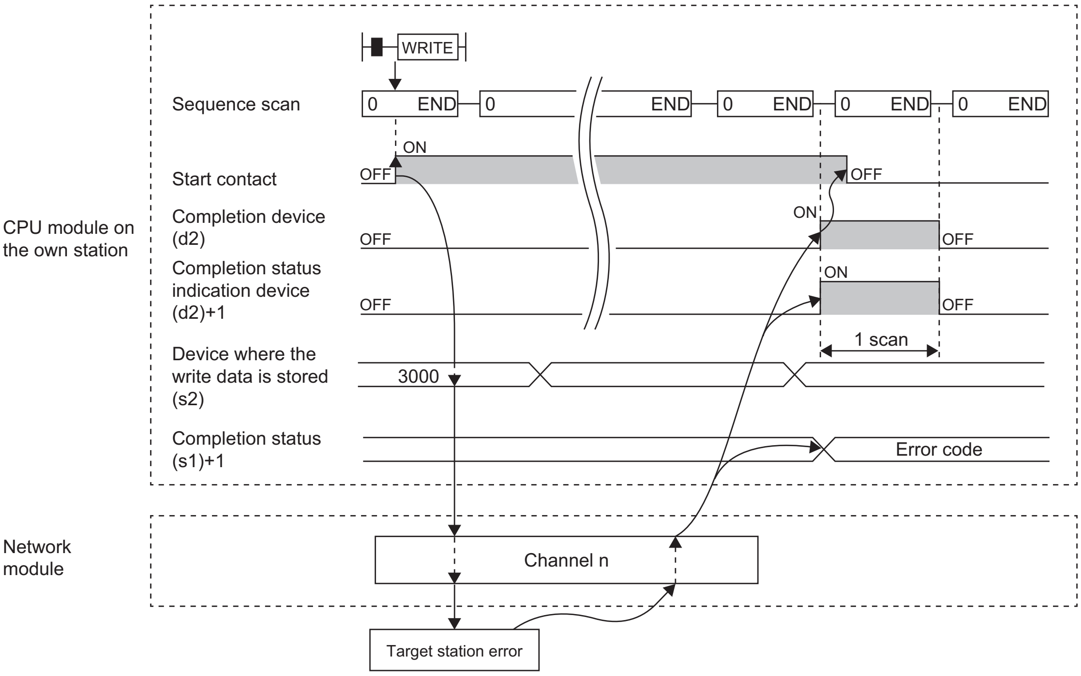
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the WRITE instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the WRITE instruction.
  - When completed successfully



  - When completed with an error



- Write processing is performed only once on the rising edge when the write command turns on.

Operation error

Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

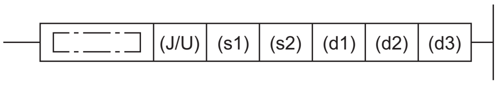
Writing data to the programmable controller on another station (with notification)

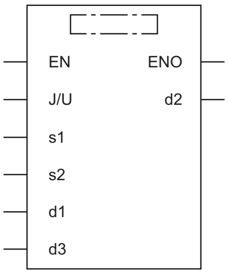
JP.SWRITE, GP.SWRITE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

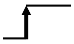
These instructions write data to a device in the programmable controller of another station (in units of words).

After the data writing is completed by the SWRITE instruction, the device of another station is turned on. The other station can recognize that data has been written by the SWRITE instruction.

Ladder	ST
	ENO:=JP_SWRITE(EN,J,s1,s2,d1,d2,d3); ENO:=GP_SWRITE(EN,U,s1,s2,d1,d2,d3);

FBD/LD


■Execution condition

Instruction	Execution condition
JP.SWRITE GP.SWRITE	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.SWRITE	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.SWRITE	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(s2)		Own station start device containing write data	—	Device name	ANY16[*1]
(d1)		Target station start device to which data is to be written	—	String[*2]	ANYSTRING_SINGLE[*2]
(d2)		Device of the own station, which 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)
(d3)		Target station device that is turned on one scan upon completion of instruction (enabling the target station to recognize that data has been written from another station)	—	String[*2]	ANYSTRING_SINGLE[*2]
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

*2 For the specifications of the string data to be specified, refer to the following.
☞ Page 1452 Specifications of character string data specified by link dedicated instructions

■Applicable devices

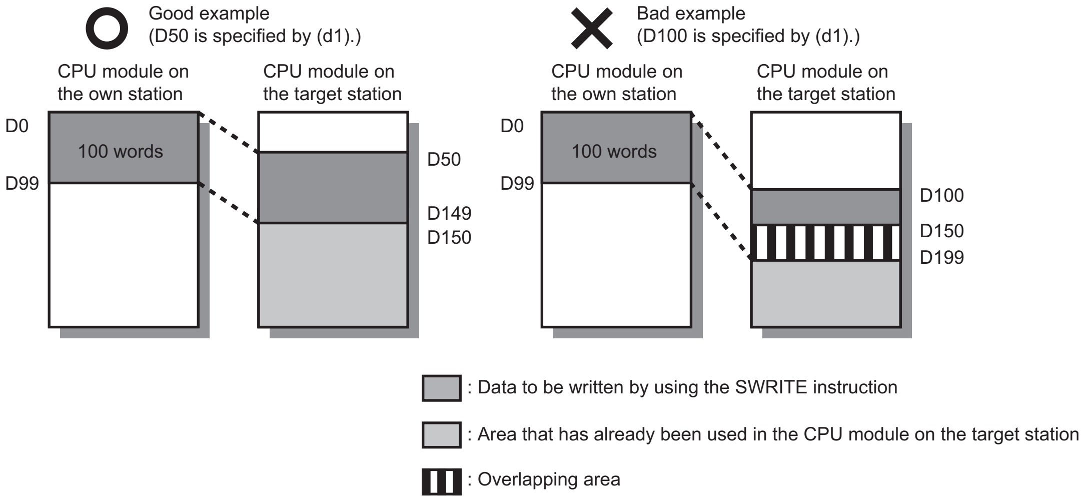
Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	J	U
(J/U)	JP.SWRITE	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.SWRITE	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	—	—	—	—	—	—	—	—	○	—	—
(d2)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—	—
(d3)		—	—	—	—	—	—	—	—	—	—	○	—	—

*1 FX and FY cannot be used.
*2 FD cannot be used.
*3 T, ST, C, and FD cannot be used.

Point

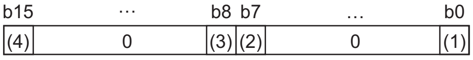
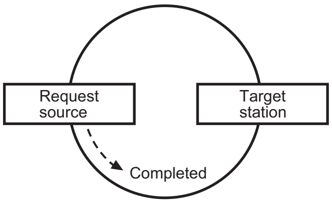
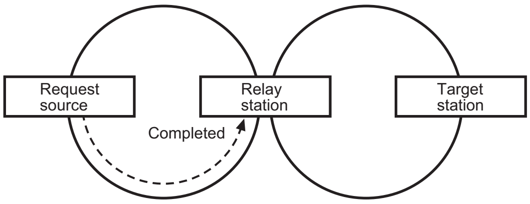
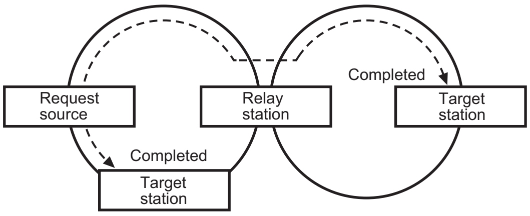
- The SWRITE instruction cannot be executed when the target station is ACPU.
- Specify the target station start device (d1), to which data is to be written, by considering the range in which the data that has been written can be stored.

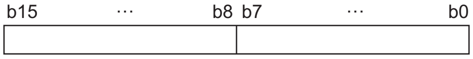
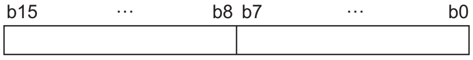
(Example: When areas D150 and after in the target station CPU module are already in use)



- When the target station for which the SWRITE instruction is executed is the basic model QCPU, the write notification device for the target station specified by argument (d3) is ignored. The operations of the SWRITE instruction are the same as those of the WRITE instruction.
- The SWRITE instruction can be programmed by omitting argument (d3). However, the operations are the same as those of the WRITE instruction. The operations of the SWRITE instruction can be selected according to whether (d3) is omitted or not.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/error completion type	 (1) Execution type (b0) • 0: No arrival acknowledgment When the target station is in the same network: Completed when data has been sent from the own station.  When the target station is in another network: Completed when data has arrived the relay station of the own network.  • 1: Arrival acknowledgment used Sending data is completed when the data is written to the target station.  (2) Error completion type (bit 7) Specify whether to set clock data when completed with an error. • 0: Do not set data in (s1)+11 and later at completion with an error. • 1: Set data in (s1)+11 and later at completion with an error. (3) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units (4) Target station address specification method (b15) • 0: Specify "network No." in (s1)+4, and "station number" in (s1)+5. • 1: Specify the IP address in (s1)+4, 5 (Ethernet only)	0000H 0001H 0080H 0081H 0100H 0101H 0180H 0181H 8000H 8001H 8080H 8081H 8100H 8101H 8180H 8181H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+4	Target network number	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the network number (1 to 239) of the target station.	■(s1)+4 Network No.: 1 to 239 ■(s1)+5 Station No.: 1 to 120, 125, 126 Group number: 0081H to 00A0H All-station specification: 00FFH ■(s1)+4, 5 IP address: 00000001H to FFFFFFFEH (1 to 4294967294)	User
		[Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the network number (1 to 239) of the target station. ■When "1" is specified in bit 15 of (s1)+0 (Ethernet only) Specify the IP address (third and fourth octets) of the target station.  • b8 to b15: 3rd octet • b0 to b7: 4th octet		
+5	Target station number	[CC-Link IE Controller Network] Specify the station number of the target station. (1) Station number specification • 1 to 120: Station number (2) Group number specification • 0081H to 00A0H: All stations with group numbers 0001H to 0020H (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") (3) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment")		User
		[CC-Link IE Field Network] Specify the station number of the target station. (1) Station number specification • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station (2) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment")		
		[Ethernet] ■When "0" is specified in bit 15 of (s1)+0 Specify the station number of the target station. (1) Station number specification • 1 to 120: Station number (2) Group number specification • 0081H to 00A0H: All stations with group numbers 0001H to 0020H (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") (3) All-station specification • 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") ■When "1" is specified in bit 15 of (s1)+0 Specify the IP address (first and second octets) of the target station.  • b8 to b15: 1st octet • b0 to b7: 2nd octet		
+6	Not used	—	—	—
+7	Number of resends	Effective when the execution type specified by (s1) is "1: Arrival acknowledgment used". ■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. ■When "0" is specified in bit 8 of (s1)+0 Specify the TCP resend timer value or greater for the monitoring time till completion of processing. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s ■When "1" is specified in bit 8 of (s1)+0 • 0: 10s • 1 to 65535: 1 to 65535×100ms	0 to 65535	User
+9	Write data length	Specify the number of words to be written. ■Writing to RCPU, QCPU, or LCPU • Channels 1 to 8 are used: 1 to 960 (words) • Channels 9 and 10 are used: 1 to 8192 (words) ■Writing to QnACPU • 1 to 480 (words)	1 to 8192	User
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

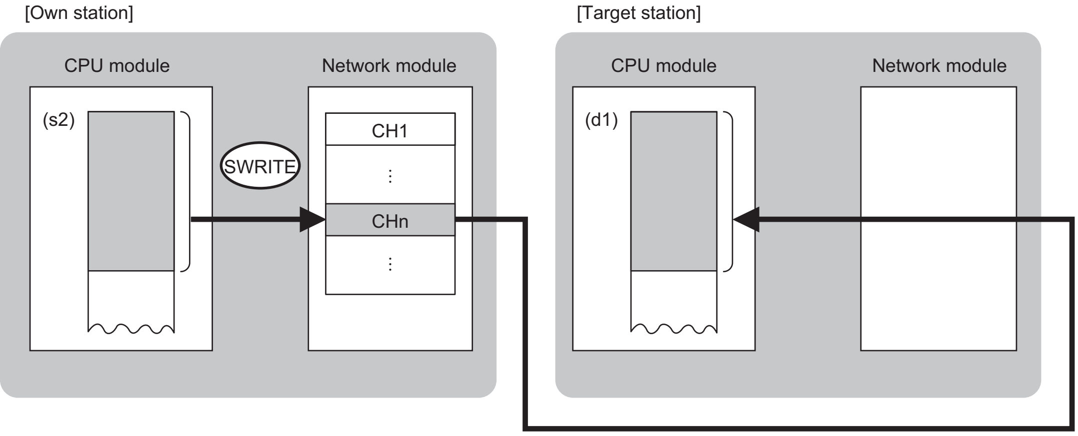
Point

- The continuous area (a maximum of 8192 words) for the write data length ((s1)+9) is required in the write data storage device (d1).
- When a number from 1 to 120 is specified for the target station number, the SWRITE instruction should be executed with the execution type set to "Arrival acknowledgment used". This is to improve data reliability. When a number from 81H to A0H or FFH is specified for the target station number, the SWRITE instruction should be executed with the execution type set to "No arrival acknowledgment".
- When performing device writing to the same station from multiple stations, do not overlap the write timing. When the execution type is set to "No arrival acknowledgment", successful completion results in the write source station if communication is completed successfully even when the send data contains an error. Also, even when the send data is normal, a timeout results in the write source station if the SWRITE instructions are executed for the same station from multiple stations.
- The number of resends (s1)+7 must be set every time the SWRITE instruction is executed.

Processing details

- These instructions write the data in the device/label specified by (s2) in the own station to the word device in the target station specified by the target network number and target station number of the control data or the target station specified by the IP address. Upon completion of writing device data to another station number, the completion device specified by (d2) turns on. In another station, upon completion of writing the device data specified by (s2), the device specified by (d3) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When "network number" and "station number" are specified ("0" is specified in bit 15 of (s1)+0) by the target station address specification method, device data can be written also to the stations connected to networks other than the stations connected to the own station network. (If "IP address" is specified ("1" is specified in bit 15 of (s1)+0), device data cannot be read from stations connected via a relay station.)
- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the SWRITE instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the SWRITE instruction completes, and turns off during the next END processing.

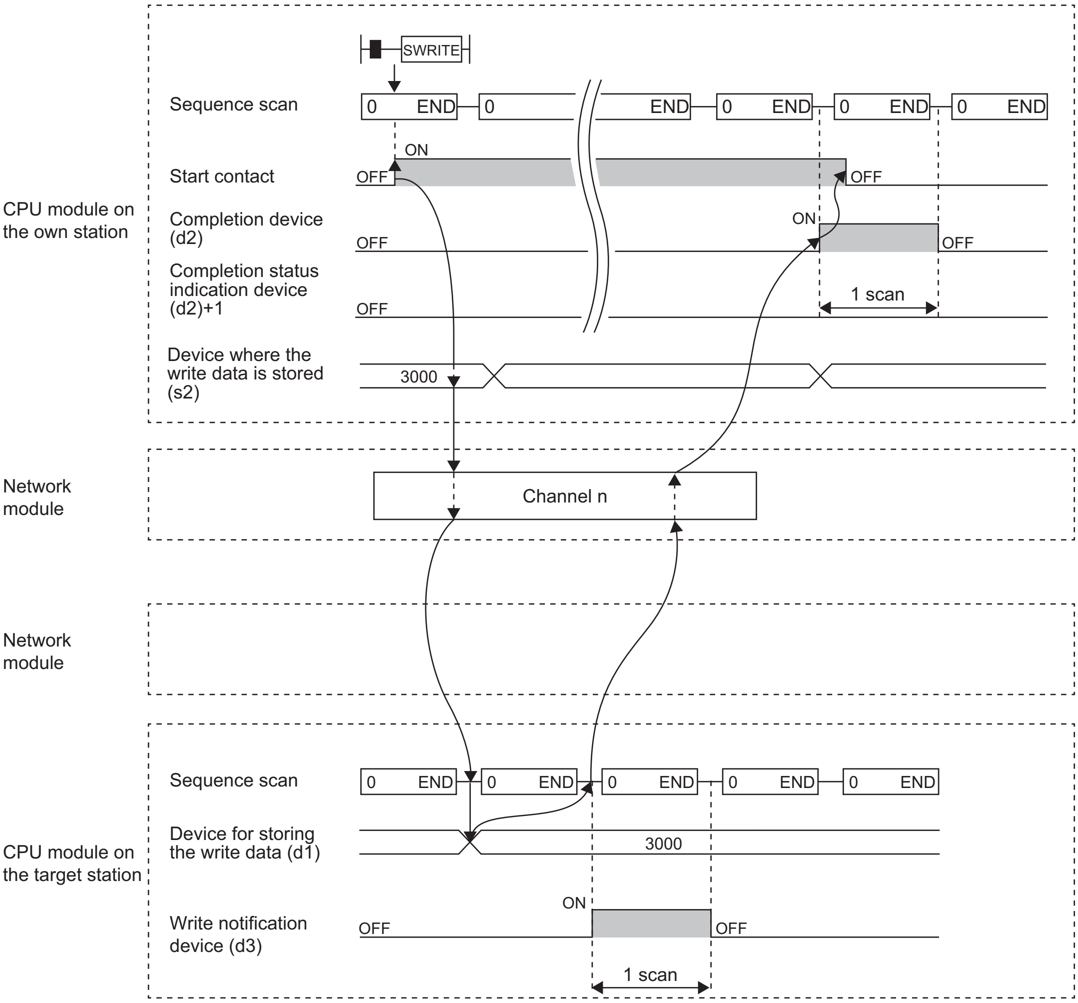
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the SWRITE instruction.

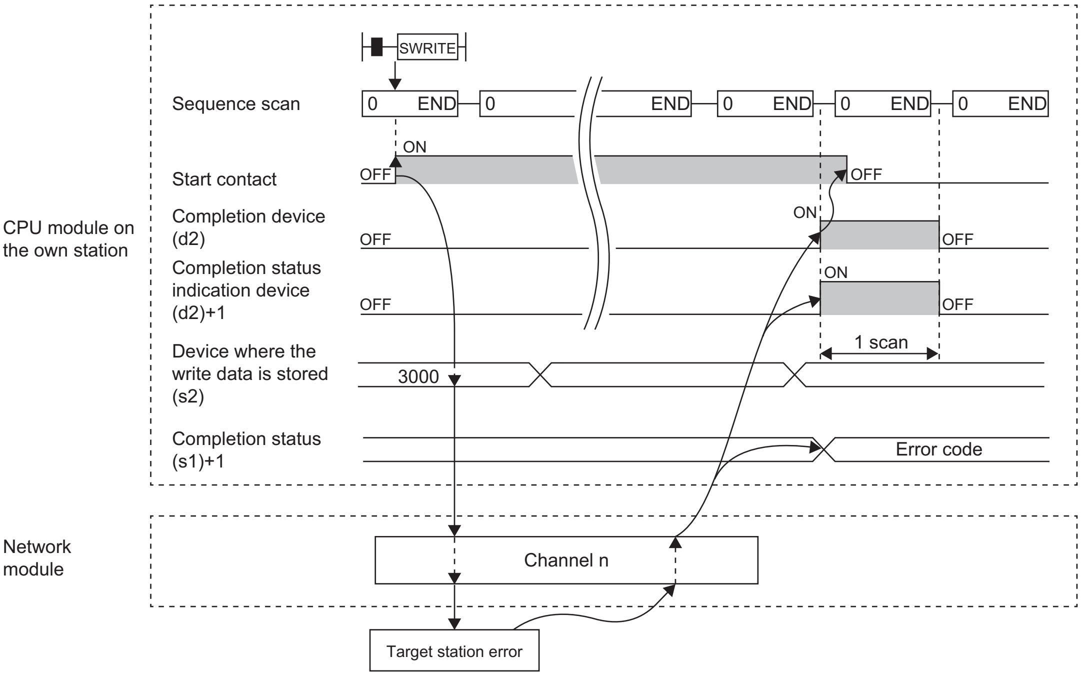
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SWRITE instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SWRITE instruction.
  - When completed successfully



  - When completed with an error



- Write processing is performed only once on the rising edge when the write command turns on.

Operation error

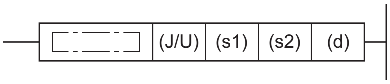
Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Sending data to the programmable controller on another station

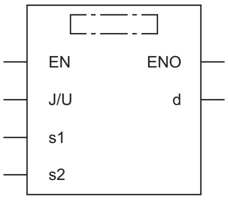
JP.SEND, GP.SEND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

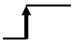
These instructions send data to the programmable controller of another station.

Ladder	ST
	ENO:=JP_SEND(EN,J,s1,s2,d); ENO:=GP_SEND(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.SEND GP.SEND	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.SEND	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.SEND	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(s2)		Own station head device where send data is stored	—	Device name	ANY16[*1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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.

Point

The SEND instruction cannot be executed when the target station is ACPU.

■Applicable devices

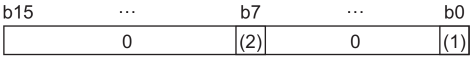
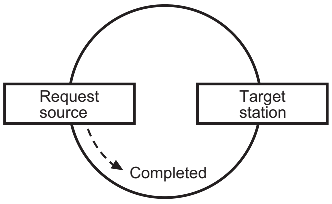
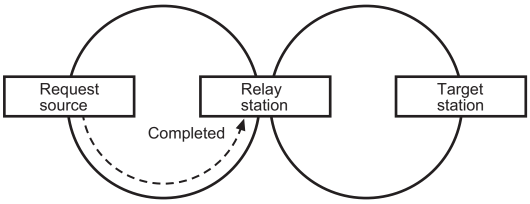
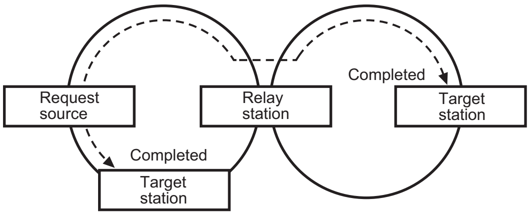
Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	J	U
(J/U)	JP.SEND	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.SEND	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/error completion type	 (1) Execution type (b0) • 0: No arrival acknowledgment When the target station is in the same network: Completed when data has been sent from the own station.  When the target station is in another network: Completed when data has arrived the relay station of the own network.  • 1: Arrival acknowledgment used Completed when data has been stored in the specified channel of the target station.  (2) Error completion type (bit 7) Specify whether to set clock data when completed with an error. • 0: Clock data at error occurrence is not stored in (s1)+11 and later. • 1: Clock data at error occurrence is stored in (s1)+11 and later.	0000H 0001H 0080H 0081H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	User
+3	Target station storage channel	Specify the channel of the target station for storing data. (CC-Link IE Field Network only: 1 or 2)	1 to 8	User

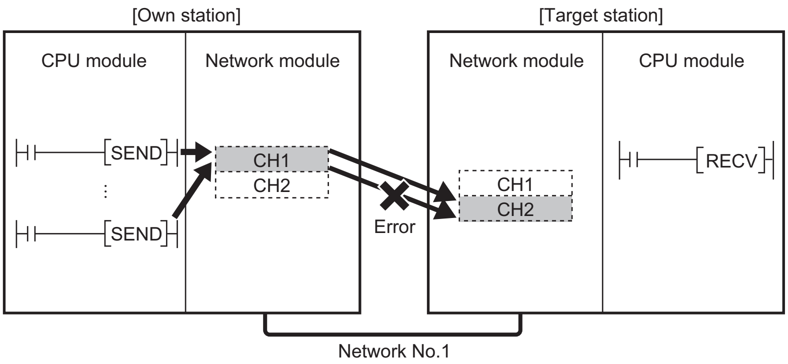
Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+4	Target network number	Specify the network number of the target station. • 1 to 239 (Network number)	1 to 239	User
+5	Target station number	Specify the station number of the target station. (1) Station number specification [Ethernet or CC-Link IE Controller Network] • 0001H to 0078H (1 to 120): Station number [CC-Link IE Field Network] • 007DH (125): Master station • 007EH (126): Master operating station • 0001H to 0078H (1 to 120): Local station, intelligent device station, submaster station (2) Group number specification [Ethernet or CC-Link IE Controller Network] 0081H to 00A0H: All stations with group numbers 01H to 20H (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment") (3) All-station specification 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be set when the execution type specified by (s1) is "0: No arrival acknowledgment")	0001H to 0078H, 007DH, 007EH, 0081H to 00A0H, 00FFH	User
+6	Not used	—	—	—
+7	Number of resends (retries)	Effective when the execution type specified by (s1) is "1: Arrival acknowledgment used". ■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
		[Ethernet] Specify the TCP resend timer value or greater for the monitoring time till completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7 (the number of resends). • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s	0 to 16383	User
+9	Send data length	Specify the number of send data from (s2) to (s2)+n. (No information is stored if an error is detected in the own station.) ■Sending to RCPU, QCPU, or LCPU • 1 to 960 (words) ■Sending to QnACPU • 1 to 480 (words)	1 to 960, 1 to 480	User
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System

Operand: (s1)

Device	Item	Description	Setting range	Set by
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

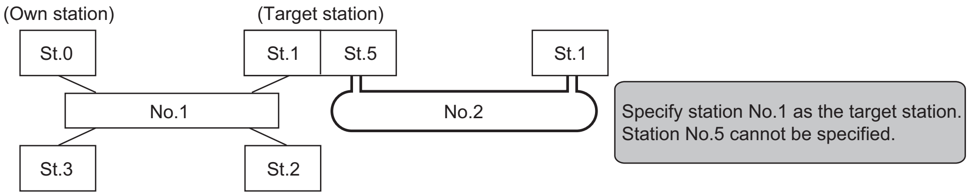
Point

- The continuous area (a maximum of 960 words) for the send data length ((s1)+9) is required in the send data storage device (d2).
- When a number from 1 to 120 is specified for the target station number, the instruction should be executed with the execution type set to "Arrival acknowledgment used" to improve data reliability. When a group number or all stations are specified, the instruction should be executed with the execution type set to "No arrival acknowledgment".
- When sending data to the same channel in the receiving station, execute the instruction after data has been read by the RECV instruction in the receiving station. When the execution type is set to "No arrival acknowledgment", successful completion results in the sending station if communication is completed successfully even when the send data contains an error. Also, even when the send data is normal, a timeout results in the sending station if the instructions are executed for the same station from multiple stations.
- With the execution type set to "Arrival acknowledgment used", if the sending station sends data to the same channel in the receiving station before the receiving station reads data using the RECV instruction, a buffer full error results in the sending station.



- When multiple network modules are mounted in the target station, specify the network number and station number of the network module that receives a request from the own station.

(Example: In the following figure, specify station No.1 of network No.1 as a target station. (Station No. 5 of network No. 2. cannot be specified.))



No.: Network No.

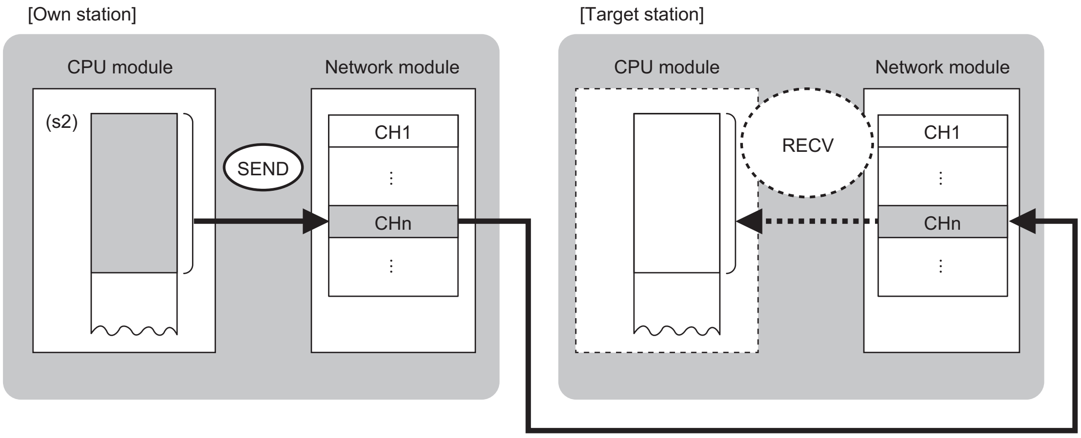
St.: Station No.

- The number of resends (s1)+7 must be set every time the instruction is executed.

Processing details

- These instructions send the data in the device specified by (s2) and later in the own station to the station connected to the target station specified by the target network number and target station number of the control data. The data that has been sent is stored in the channel specified by (s2) in the target station. To read the sent data in the target station, use the RECV/RECVS instruction. Upon completion of sending data to the target station number, the completion device specified by (d) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- Data can be sent not only to the stations connected to the own station network but also to stations connected to the network number specified by MELSECNET/H, MELSECNET/10, or Ethernet.
- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the SEND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the SEND instruction completes, and turns off during the next END processing.

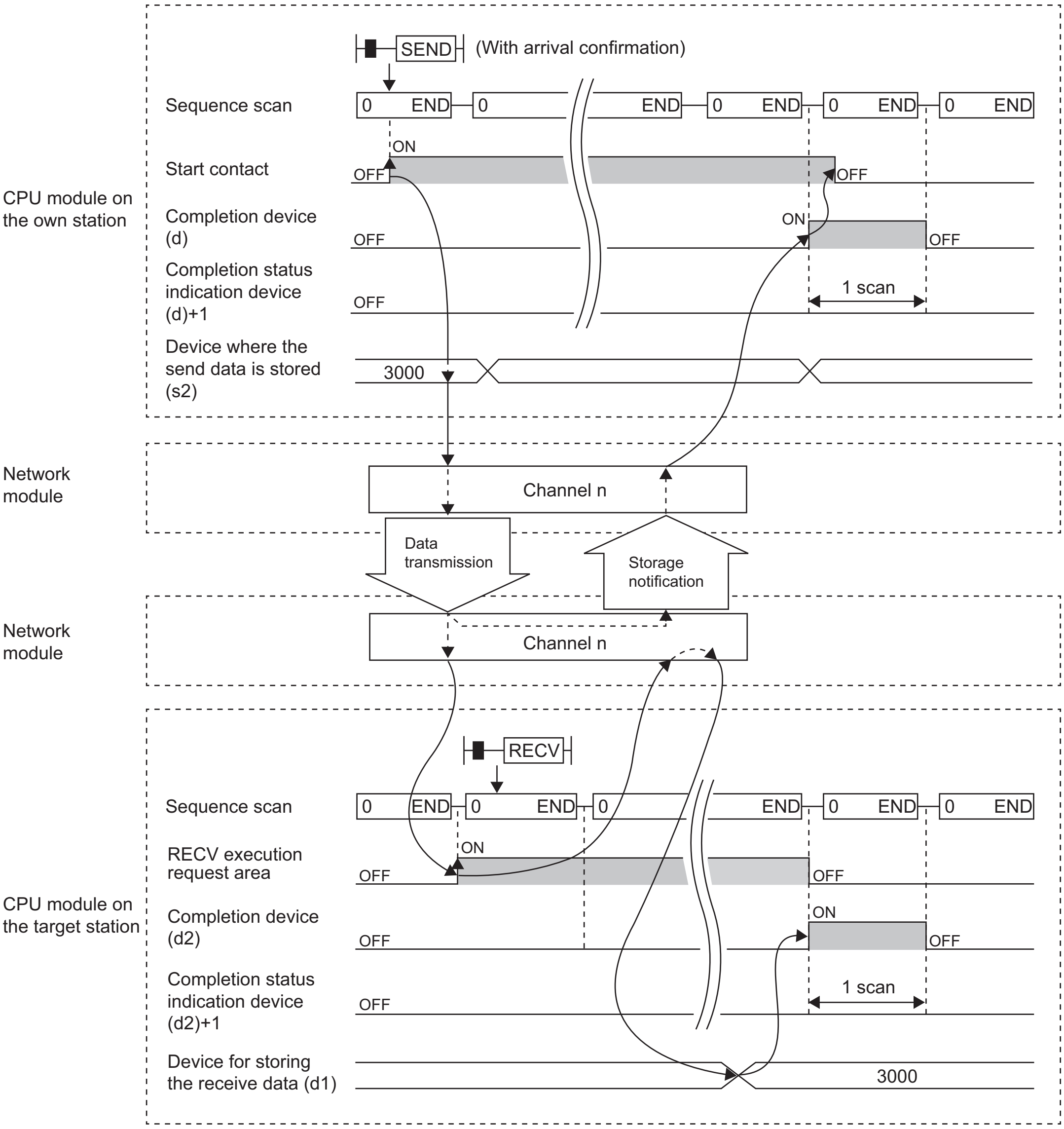
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SEND instruction.

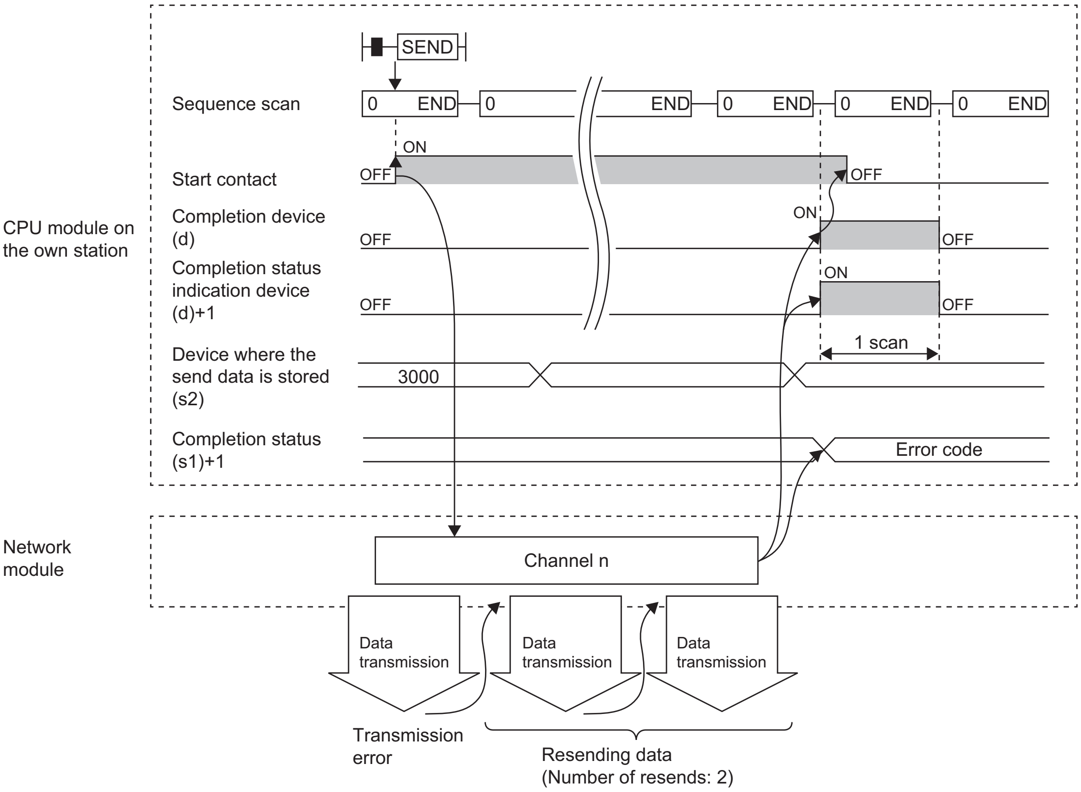
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SEND instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SEND instruction.
  - When completed successfully



- When completed with an error



- Send processing is performed only once on the rising edge when the send command turns on.

Operation error

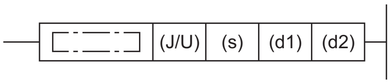
Error code ((s1)+1)	Description
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	🕮 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	🕮 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

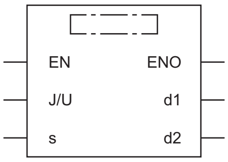
Receiving data from the programmable controller on another station

JP.RECV, GP.RECV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read the data received from the programmable controller of another station (for the main routine program).

Ladder	ST
	ENO:=JP_RECV(EN,J,s,d1,d2); ENO:=GP_RECV(EN,U,s,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
JP.RECV GP.RECV	↑

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	JP.RECV	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	GP.RECV	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)		Own station head device for storing the receive data (A continuous area for the receive data length is required.)	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] 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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	JP.RECV	—	—	—	—	—	—	—	—	—	—	—	○	—
	GP.RECV	—	—	○	—	—	—	—	○	○	—	—	—	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

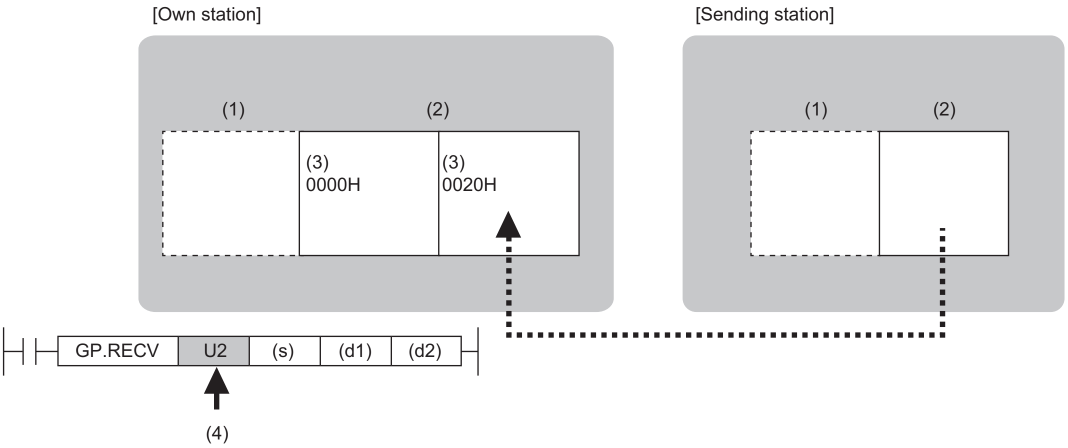
Device	Item	Description	Setting range	Set by
+0	Error completion type	b15 ··· b7 ··· b0 0 | (1) | 0 (1) Error completion type (bit 7) Specify whether to set clock data when completed with an error. • 0: Clock data at error occurrence is not stored in (s1)+11 and later. • 1: Clock data at error occurrence is stored in (s1)+11 and later.	0000H 0080H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station storage channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	User
+3	Channel used by send station	The channel number (1 to 8) used by the sending station is stored.	1 to 8	System
+4	Send station network number	The network number (1 to 239) of the sending station is stored.	1 to 239	System
+5	Sending-station No.	The station number of the sending station is stored. [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number (receiving from station number) [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System
+6	Not used	—	—	—
+7	Not used	—	—	—
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. The instruction is completed with an error if it fails to complete within the monitoring time. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
		[Ethernet] Specify the TCP resend timer value or greater for the monitoring time till completion of processing. The instruction is completed with an error if it fails to complete within the monitoring time. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s	0 to 16383	User
+9	Receive data length	The number of data received and stored in (d1) to (d1)+n is stored. • 0: No receive data • 1 to 960: Number of words of received data	0 to 960	System
+10	Not used	—	—	—
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s)+12 and later is stored. Note that the data in (s)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

Point

- The continuous area (a maximum of 960 words) for the receive data length ((s)+9) is required in the receive data storage device (d1).
- The RECV instruction is executed each time the bit of the relevant channel in the RECV instruction request area (address 205) in the buffer memory is set to ON (received data exists).
- When multiple network modules with the same network number are mounted in the own station, Specify "U" (start I/O number) to execute the RECV instruction. The RECV instruction cannot be executed by specifying "J" (network number).

(Example: When executing the GP.RECV instruction in station number 3 with regard to the SEND instruction from station number 2, specify "U2".)



(1) CPU module

(2) Network module

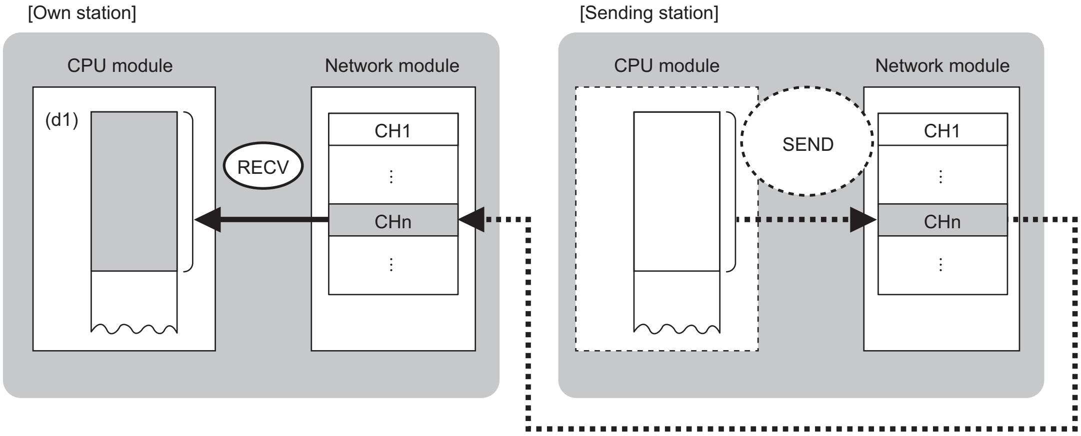
(3) Start I/O number

(4) Specify U (start I/O number)

Processing details

- These instructions read the data sent from the station connected to the target station specified by the network number and station number of control data from the module of the own station. Upon completion of reading the received data from the network module, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- The SEND instruction is executed to send data. The data received from the sending station is stored in the own station channel specified by the sending station and the corresponding bit in the RECV instruction execution request area for each network module is set to ON. When the relevant bit in the RECV instruction execution request area is set to ON, the received data is read from the receive data storage channel. The following table lists the RECV instruction execution request areas for individual networks.

Network	Name of RECV instruction execution request area	Device number/address of applicable bit
CC-Link IE Field Network CC-Link IE Controller Network	RECV instruction channel 1 execution request flag	SB0030
	RECV instruction channel 2 execution request flag	SB0031
	RECV instruction channel 3 execution request flag	SB0032
	RECV instruction channel 4 execution request flag	SB0033
	RECV instruction channel 5 execution request flag	SB0034
	RECV instruction channel 6 execution request flag	SB0035
	RECV instruction channel 7 execution request flag	SB0036
	RECV instruction channel 8 execution request flag	SB0037
Ethernet (Q series compatible)	RECV instruction execution request area	Buffer memory address 205 (CDH) bit 0
		Buffer memory address 205 (CDH) bit 1
		Buffer memory address 205 (CDH) bit 2
		Buffer memory address 205 (CDH) bit 3
		Buffer memory address 205 (CDH) bit 4
		Buffer memory address 205 (CDH) bit 5
		Buffer memory address 205 (CDH) bit 6
		Buffer memory address 205 (CDH) bit 7
Ethernet[1]	RECV instruction execution request area	Buffer memory address 5301 (14B5H) bit 0
		Buffer memory address 5301 (14B5H) bit 1
		Buffer memory address 5301 (14B5H) bit 2
		Buffer memory address 5301 (14B5H) bit 3
		Buffer memory address 5301 (14B5H) bit 4
		Buffer memory address 5301 (14B5H) bit 6
		Buffer memory address 5301 (14B5H) bit 7

[1] When the network type of the Ethernet module is "Ethernet", the buffer memory address of PORT2 is increased by 2000000.
📖 MELSEC iQ-R Ethernet User's Manual (Application)

- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- When the RECV instruction is used to read receive data from the same channel, the RECVS instruction (for use in interrupt programs) cannot be used in combination.
- The execution status and the completion status of the RECV instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the RECV instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the RECV instruction.

When completed successfully: The device remains off.

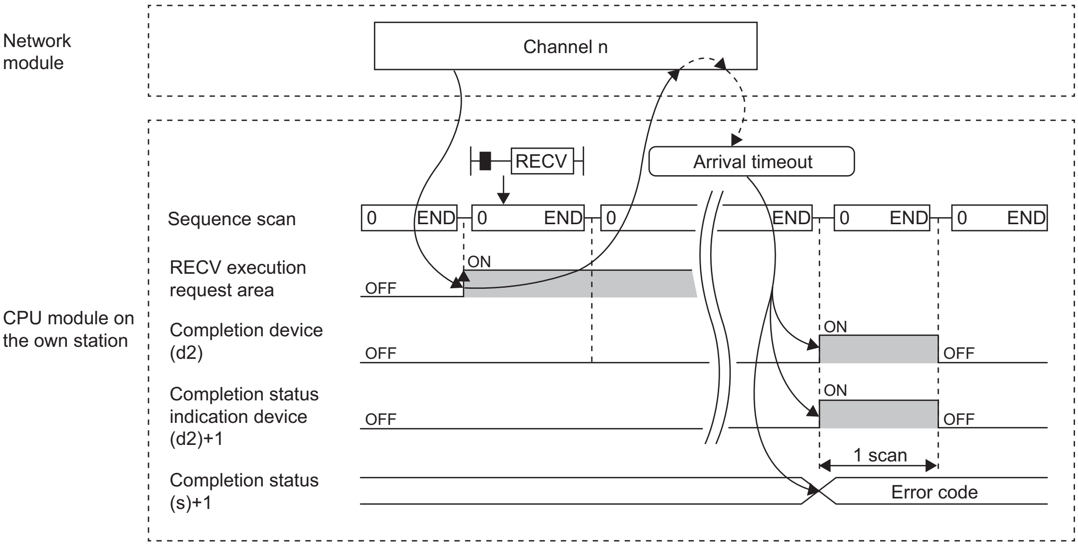
When completed with an error: The device turns on during END processing of the scan where the RECV instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the RECV instruction.
  - When completed successfully

For the execution timing, refer to the following functions.

☞ Page 1485 JP.SEND, GP.SEND

- When completed with an error



- Read processing is performed only once on the rising edge when the read command turns on.

Operation error

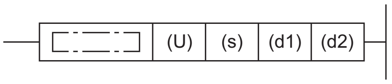
Error code ((s)+1)	Description
C000H to CFFFH	🕮 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	🕮 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Receiving data from the programmable controller on another station (for interrupt programs)

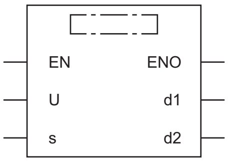
G.RECVS, Z.RECVS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read the data received from the programmable controller of another station (for interrupt programs).

Ladder	ST
 (U) (s) (d1) (d2)	ENO:=G_RECVS(EN,U,s,d1,d2); ENO:=Z_RECVS(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RECVS Z.RECVS	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G.RECVS	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z.RECVS	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)		Own station head device for storing the receive data (A continuous area for the receive data length is required.)	—	Device name	ANY16[1]
(d2)		Dummy	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	G.RECVS	—	—	○	—	—	—	—	○	○	—	—	○
	Z.RECVS	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○[2]	—	—	—	—	○	—	—	—	—
(d1)		—	—	○[2]	—	—	—	—	○	—	—	—	—
(d2)		○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station storage channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	User
+3	Channel used by send station	The channel number (1 to 8) used by the sending station is stored.	1 to 8	System
+4	Send station network number	The network number (1 to 239) of the sending station is stored.	1 to 239	System
+5	Sending-station No.	The station number of the sending station is stored. [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number (receiving from station number) [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System
+6 to +8	System area	—	—	—
+9	Receive data length	The number of data received and stored in (d1)+0 to (d1)+□ is stored. • 0: No receive data • 1 to 960: Number of words of received data	0 to 960	System

■Receive data

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	Receive data	Receive data is stored.	—	System

Point

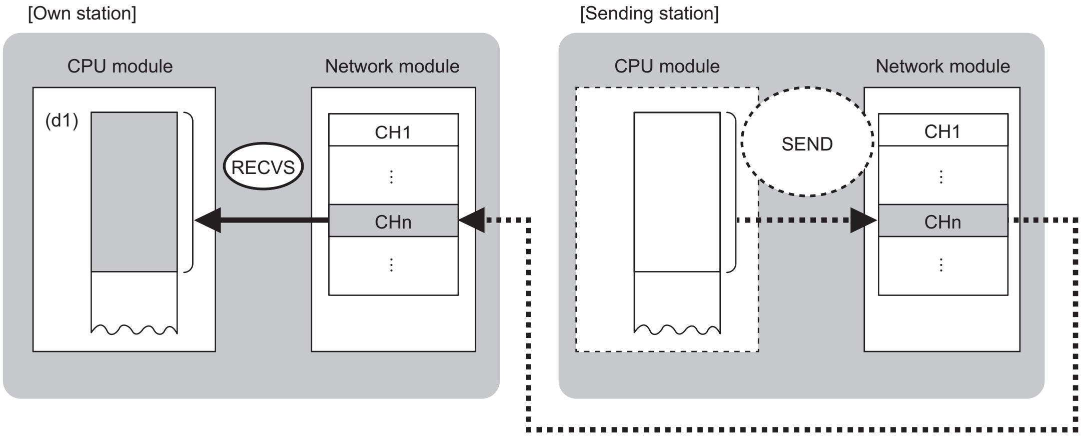
The continuous area (a maximum of 960 words) for the receive data length ((s)+9) is required in the receive data storage device (d1).

Processing details

- These instructions read the receive data of the target station sent to (s)+2 (own station storage channel) of control data.
- For the target stations that can be specified, refer to the following.

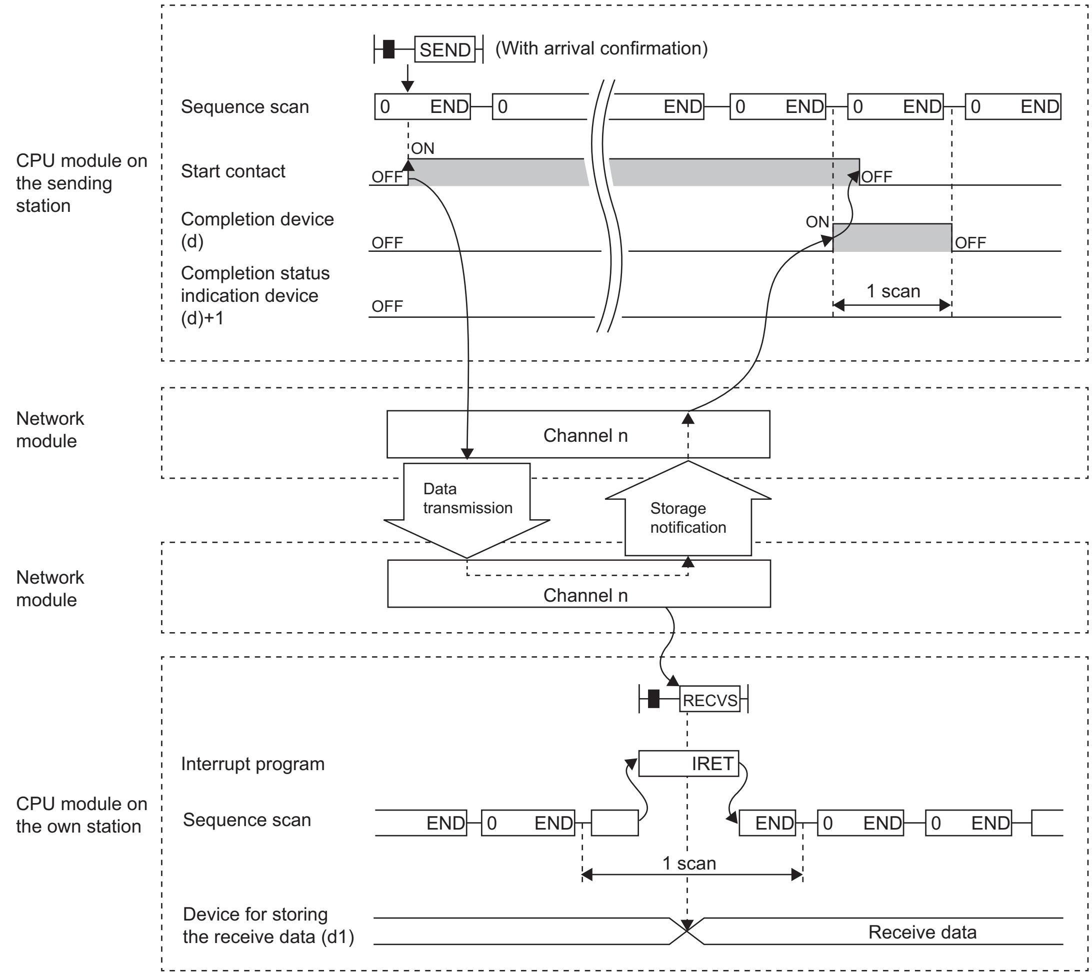
☞ Page 1449 Target networks and target station types

- The SEND instruction is executed to send data. The data received from the sending station is stored in the own station channel specified by the sending station, and the interrupt program with the interrupt number specified by the engineering tool starts. The RECVS instruction is used in this interrupt program to read receive data.

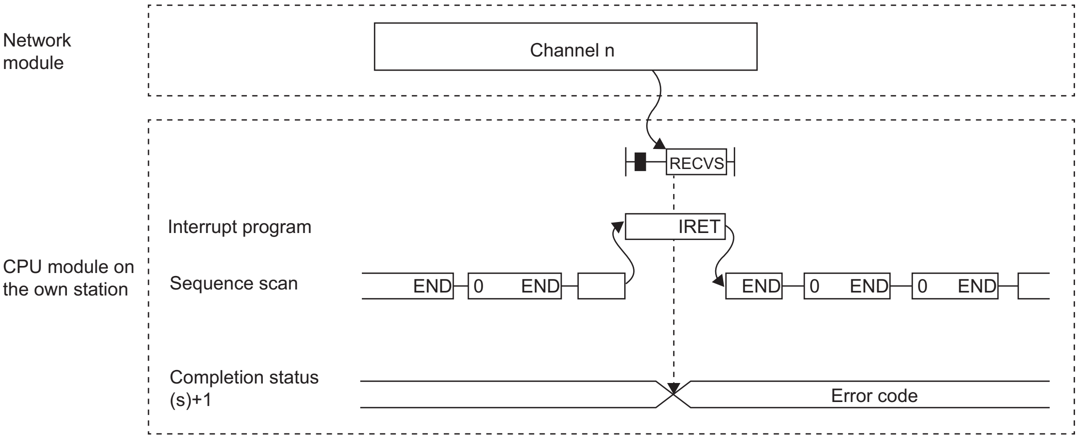


CH: Channel

- The following figure shows the execution timing of the RECVS instruction.
  - When completed successfully



- When completed with an error



- The RECVS instruction is used in interrupt programs and processing is completed in a single scan.
- When the RECVS instruction is used to read receive data from the same channel, the RECV instruction (for use in the main program) cannot be used in combination.

Operation error

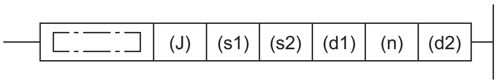
Error code ((s)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Reading data from the programmable controller (Q series-compatible)

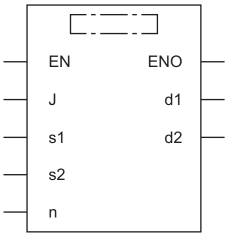
J(P).ZNRD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

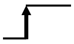
These instructions read data in units of words from a device in the programmable controller (MELSEC-A/QnA/Q/L series) of another station.

Ladder	ST
	ENO:=J_ZNRD(EN,J,s1,s2,n,d1,d2); ENO:=JP_ZNRD(EN,J,s1,s2,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.ZNRD	
JP.ZNRD	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(J)	Target station network number	1 to 239	16-bit unsigned binary	ANY16
(s1)	Target station number	1 to 64	16-bit unsigned binary	ANY16
(s2)	Target station start device where the data to be read is stored	—	String*1	ANYSTRING_SINGLE*1
(d1)	Own station start device for storing the read data (A continuous area for the read data length is required.)	—	Device name	ANY16*2
(n)	Read data length	■When the target station is AnUCPU/QnACPU/QCPU/LCPU 1 to 230 ■When the target station is a MELSEC-A series CPU module other than AnUCPU 1 to 32	16-bit unsigned binary	ANY16
(d2)	Device of the own station, which 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 For the specifications of the string data to be specified, refer to the following.
☞ Page 1452 Specifications of character string data specified by link dedicated instructions

*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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J)
	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	$	
(J)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d1)	—	—	○*2	—	—	—	—	○	—	—	—	—
(n)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d2)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Point

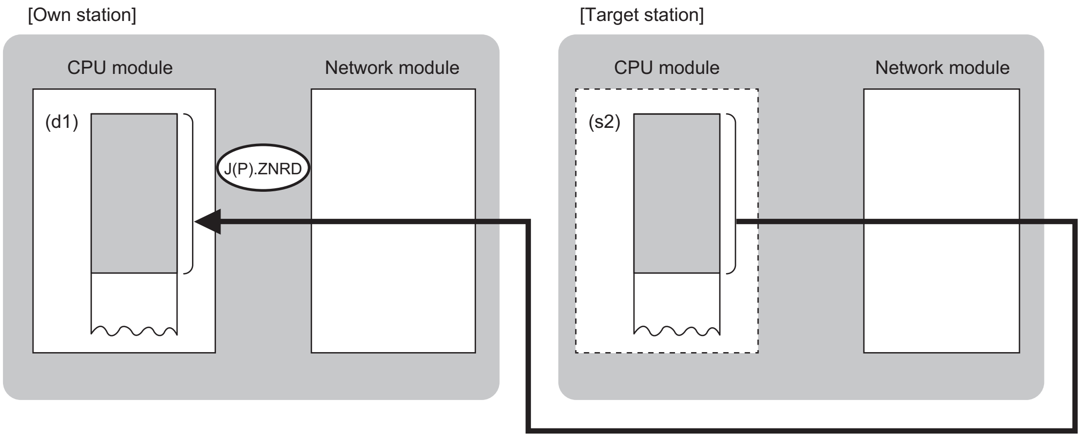
The J(P).ZNRD instruction is executed using the following fixed values in addition to setting data.

- Arrival monitoring time: 10s
- Number of resends: 5

Processing details

- These instructions read the data from the specified word device in the target station specified by the target network number (J) and target station number (s1). Upon completion of reading the device data, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



Point

- The J(P).ZNRD instruction cannot be executed for RCPU. If executed, error code 4001H is stored in the completion status and the instruction is completed with an error.
- The J(P).ZNRD instruction is an instruction (Q series compatible instruction) for replacement of the equivalent instruction used in the programs running on the MELSEC-Q series. When creating a new program, use the READ instruction.

- Device data can be read also from stations connected to networks other than the stations connected to the own station network.
- The execution status and the completion status of the J(P).ZNRD instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the J(P).ZNRD instruction completes, and turns off during the next END processing.

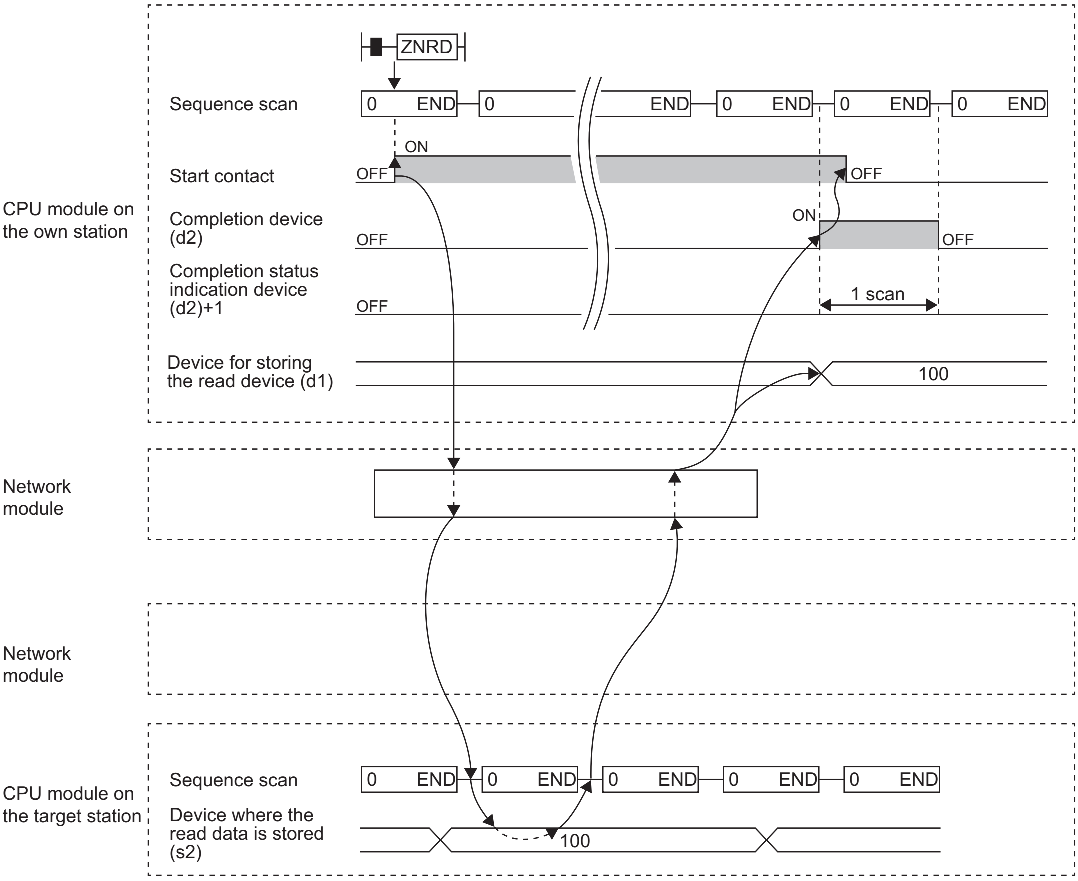
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the J(P).ZNRD instruction.

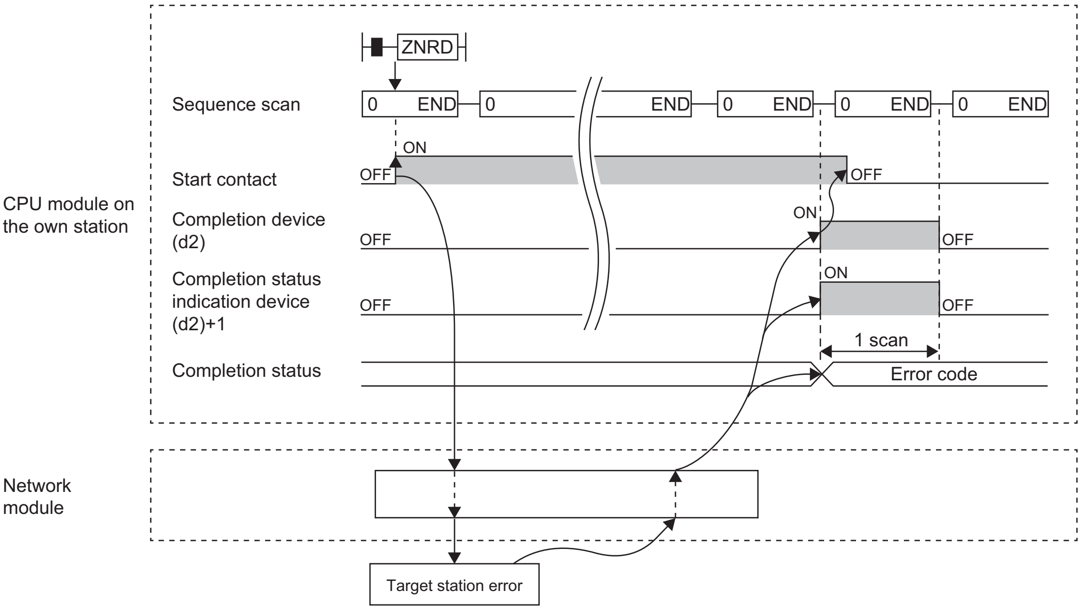
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the J(P).ZNRD instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the J(P).ZNRD instruction.
  - When completed successfully



  - When completed with an error



The completion status is stored in the special register (SW) when the CC-Link IE Controller Network is used or in the buffer memory when Ethernet is used.

- When the J(P).ZNRD instruction is executed, read processing is performed only once on the rising edge when the read command turns on.

Precautions

The J(P).ZNRD instruction cannot be executed when the CPU module on the target station is one of the following:

- AnUCPU with the version AX (manufactured in July 1995) or earlier
- A2USCPU(-S1) with the version CN (manufactured in July 1995) or earlier

If executed, the dedicated instruction response timer causes a timeout in the instruction start source and the J(P).ZNRD instruction is completed with an error. If the dedicated instruction response timer causes a timeout, use a CPU module satisfying one of the following versions.

- AnUCPU with the version AY (manufactured in July 1995) or later
- A2USCPU(-S1) with the version CP (manufactured in July 1995) or later

Operation error

Error code[4]	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

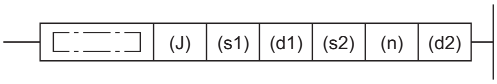
[4] The completion status in which an error code is stored is as follows.
CC-Link IE Controller Network: SW003A
Ethernet: Buffer memory address 5323 (14CBH)

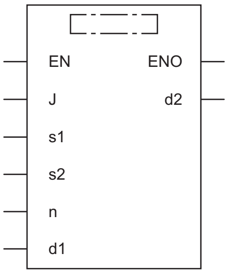
Writing data to the programmable controller (Q series-compatible)

J(P).ZNWR

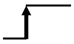
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions write data in units of words to a device in the programmable controller (MELSEC-A/QnA/Q/L series) of another station.

Ladder	ST
	ENO:=J_ZNWR(EN,J,s1,s2,n,d1,d2); ENO:=JP_ZNWR(EN,J,s1,s2,n,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
J.ZNWR	
JP.ZNWR	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(J)	Target station network number	1 to 239	16-bit unsigned binary	ANY16
(s1)	Target station number	■Station number specification 1 to 64: Station number station ■Group specification 0081H to 00A0H: All stations with group numbers 1 to 32 ■All-station specification 00FFH: All stations of target network number	16-bit unsigned binary	ANY16
(d1)	Target station start device to which data is to be written (A continuous area for the write data length is required.)	—	String*1	ANYSTRING_SINGLE*1
(s2)	Own station start device where write data is stored	—	Device name	ANY16*2
(n)	Write data length	■When the target station is AnUCPU/QnACPU/QCPU/LCPU 1 to 230 ■When the target station is a MELSEC-A series CPU module other than AnUCPU 1 to 32	16-bit unsigned binary	ANY16
(d2)	Device of the own station, which 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 For the specifications of the string data to be specified, refer to the following.
☞ Page 1452 Specifications of character string data specified by link dedicated instructions

*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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J)
	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	$	
(J)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d1)	—	—	—	—	—	—	—	—	—	—	○	—
(s2)	—	—	○*2	—	—	—	—	○	—	—	—	—
(n)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d2)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Point

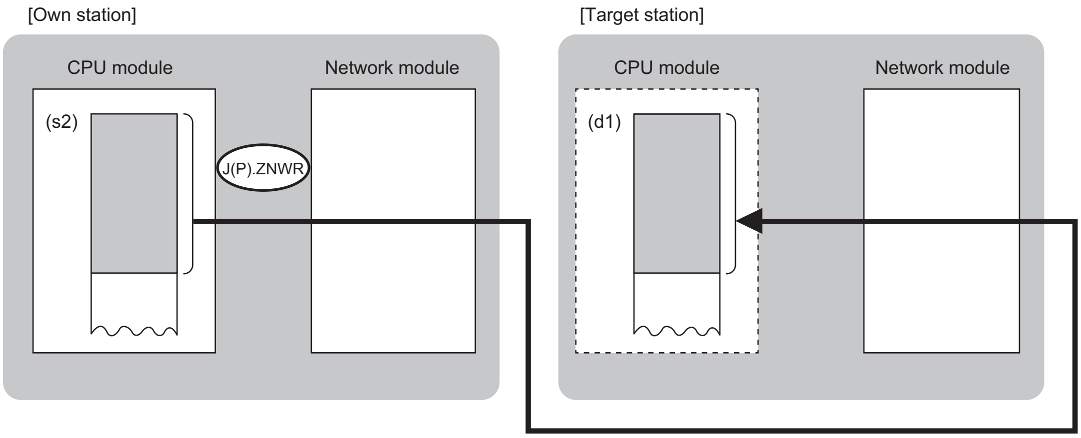
The J(P).ZNWR instruction is executed using the following fixed values in addition to setting data.

- Arrival monitoring time: 10s
- Number of resends: 5

Processing details

- These instructions write the data in the device/label specified by (s2) in the own station to the word device in the target station specified by the target network number and target station number of the control data. Upon completion of writing device data to another station number, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



Point

- The J(P).ZNWR instruction cannot be executed for RCPU. If executed, error code 4001H is stored in the completion status and the instruction is completed with an error.
- The J(P).ZNWR instruction is an instruction (Q series compatible instruction) for replacement of the equivalent instruction used in the programs running on the MELSEC-Q series. When creating a new program, use the WRITE instruction.

- Device data can be written also to stations connected to networks other than the stations connected to the own station network.
- The execution status and the completion status of the J(P).ZNWR instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the J(P).ZNWR instruction completes, and turns off during the next END processing.

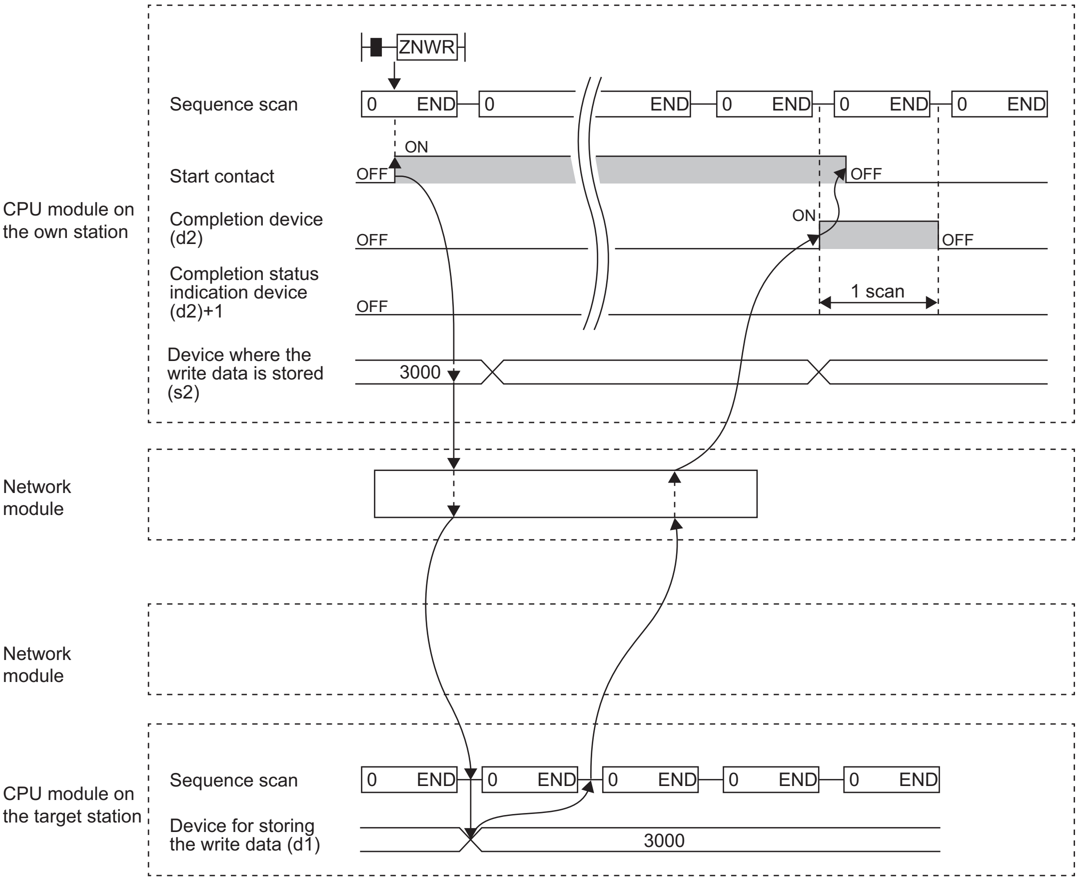
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the J(P).ZNWR instruction.

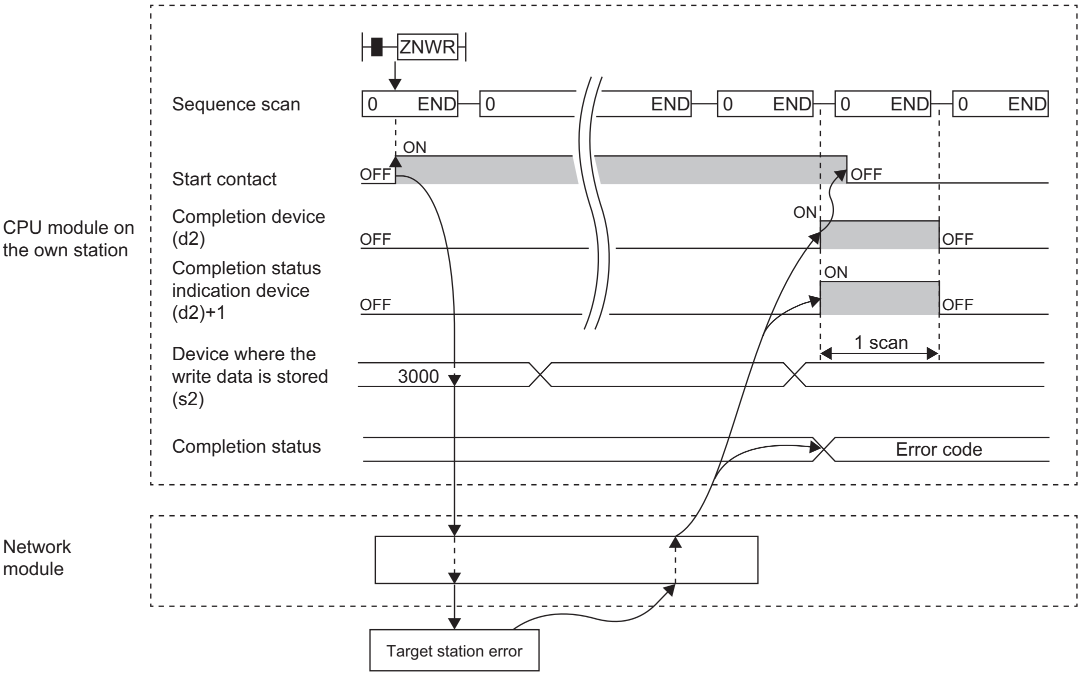
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the J(P).ZNWR instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the J(P).ZNWR instruction.
  - When completed successfully



- When completed with an error



The completion status is stored in the special register (SW) when the CC-Link IE Controller Network is used or in the buffer memory when Ethernet is used.

- When the J.ZNWR instruction is used for execution, one complete cycle of write processing is followed by another while the write command is on. When the JP.ZNWR instruction is executed, write processing is performed only once on the rising edge when the write command turns on.

Precautions

The J(P).ZNWR instruction cannot be executed for the following CPU modules.

- AnUCPU with the version AX (manufactured in July 1995) or earlier
- A2USCPU(-S1) with the version CN (manufactured in July 1995) or earlier

If executed, the dedicated instruction response timer causes a timeout in the instruction start source and the J(P).ZNWR instruction is completed with an error. If the dedicated instruction response timer causes a timeout, use a CPU module satisfying one of the following versions.

- AnUCPU with the version AY (manufactured in July 1995) or later
- A2USCPU(-S1) with the version CP (manufactured in July 1995) or later

Operation error

Error code[4]	**Description**
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

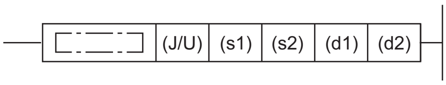
[4] The completion status in which an error code is stored is as follows.
CC-Link IE Controller Network: SW003B
Ethernet: Buffer memory address 5325 (14CDH)

Remote RUN/STOP

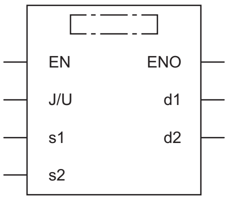
J(P).REQ, G(P).REQ

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

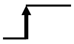
These instructions execute remote RUN or STOP for the programmable controller of another station.

Ladder	ST
	ENO:=J_REQ(EN,J,s1,s2,d1,d2); ENO:=JP_REQ(EN,J,s1,s2,d1,d2); ENO:=G_REQ(EN,U,s1,s2,d1,d2); ENO:=GP_REQ(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.REQ G.REQ	
JP.REQ GP.REQ	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).REQ	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).REQ	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(s2)		Own station start device where request data is stored	—	Device name	ANY16[*1]
(d1)		Own station start device for storing response data	—	Device name	ANY16[*1]
(d2)		Device of the own station, which 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] 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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).REQ	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).REQ	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 FD cannot be used.
*3 T, ST, C, and FD cannot be used.

Point

The REQ instruction cannot be executed when the target station is ACPU.

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Error completion type	b15 … b7 … b4 … b0 0 | (1) | 0 | 1 | 0 | 1 (1) Error completion type (bit 7) Specify whether to set data when completed with an error. • 0: Do not set data in (s1)+11 and later at completion with an error. • 1: Set data in (s1)+11 and later at completion with an error.	0011H 0091H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User
+4	Target network number	Specify the network number of the target station. • 1 to 239: Network number	1 to 239	User
+5	Target station number	Specify the station number of the target station. (1) Station number specification [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station (2) Group number specification [Ethernet or CC-Link IE Controller Network] 0081H to 00A0H: All stations with group numbers 0001H to 0020H (Can be specified for remote RUN/STOP) (3) All-station specification 00FFH: All stations of target network number (broadcast (excluding the own station)) (Can be specified for remote RUN/STOP)	1 to 120, 125, 126, 0081H to 00A0H, 00FFH	User
+6	Not used	—	—	—

Operand: (s1)

Device	Item	Description	Setting range	Set by
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s	0 to 16383	User
+9	Request data length	Specify the number of request data (words). (Number of words of data stored in the request data storage device (s2)) • Remote RUN: 4 • Remote STOP: 3	3, 4	User
+10	Response data length	The number of response data (words) is stored. (Number of words of data stored in the response data storage device) • Remote RUN/STOP: 2	2	System
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state) (No information is stored if an error is detected in the own station.)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

■Request data

Operand: (s2)					
Device	**Item**	**Description**	**Remote RUN**	**Remote STOP**	**Set by**
+0	Request type	• 0010H: When the station number is specified by (s1)+5 • 0030H: When All Station or the group is specified by (s1)+5	○	○	User
+1	Subrequest type	• 0001H: Remote RUN • 0002H: Remote STOP	○	○	User
+2	Operation mode	Specify whether to forcibly execute Remote RUN/STOP. ■Remote RUN • 0001H: Do not forcibly execute. • 0003H: Forcibly execute (can be specified when executing Remote RUN). ■Remote STOP • 0003H (fixed) (Forced execution is the function used, when the station that executed Remote STOP can no longer executed Remote RUN, to allow another station to forcibly execute Remote RUN.)	○	○	User
+3	Clear mode	Effective only for Remote RUN. Specify whether to clear the CPU device memory. • 0000H: Do not clear (the local device is cleared, though). • 0001H: Clear (excluding the latch range) • 0002H: Clear (including the latch range)	○	—	User

■Response data

Operand: (d1)					
Device	**Item**	**Description**	**Remote RUN**	**Remote STOP**	**Set by**
+0	Request type	• 0090H: When the station number is specified by (s1)+5	○	○	System
+1	Subrequest type	• 0001H: Remote RUN • 0002H: Remote STOP	○	○	System

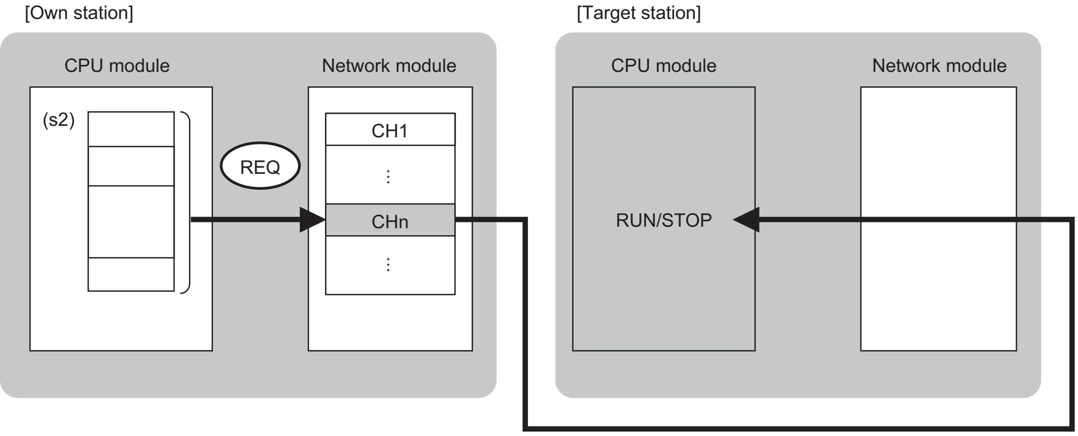
Point

- For details on the remote operation, refer to the user's manual for the CPU module used.
- Remote RUN/STOP is valid when the RUN/STOP key switch on the RCPU, QCPU, LCPU, or QnACPU of the target station is set to RUN.
- Remote RUN/STOP is disabled when System Protect is applied to the CPU module of the target station.
- While another station already executes Remote STOP/PAUSE for the target station, the own station cannot execute RUN if "Do not forcibly execute" is set in (s2)+2.
- If the RCPU, QCPU, LCPU, or QnACPU of the target station for which Remote RUN/STOP has been executed is reset, the Remote RUN/STOP information is deleted.
- The clear mode set in (s2)+3 is used to specify the clear processing (initialization) mode of the device memory of RCPU, QCPU, LCPU, or QnACPU when the CPU operation is started by Remote RUN. The RCPU, QCPU, LCPU, or QnACPU performs clearing as specified and runs according to the parameter setting (initial device value).
- The number of resends (s1)+7 must be set every time the instruction is executed.

Processing details

- These instructions send the request data specified by (S2) in the own station to the target station specified by the target network number and target station number of the control data. Upon completion of the request to the target station, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the REQ instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the REQ instruction completes, and turns off during the next END processing.

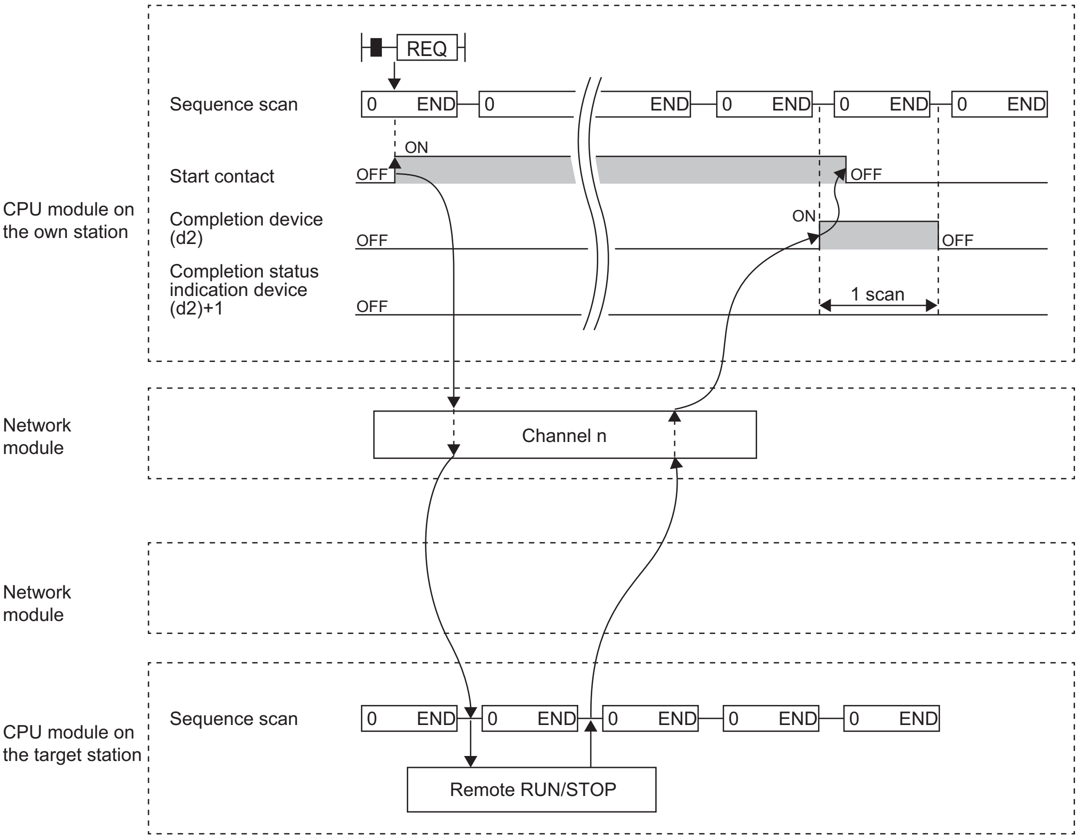
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the REQ instruction.

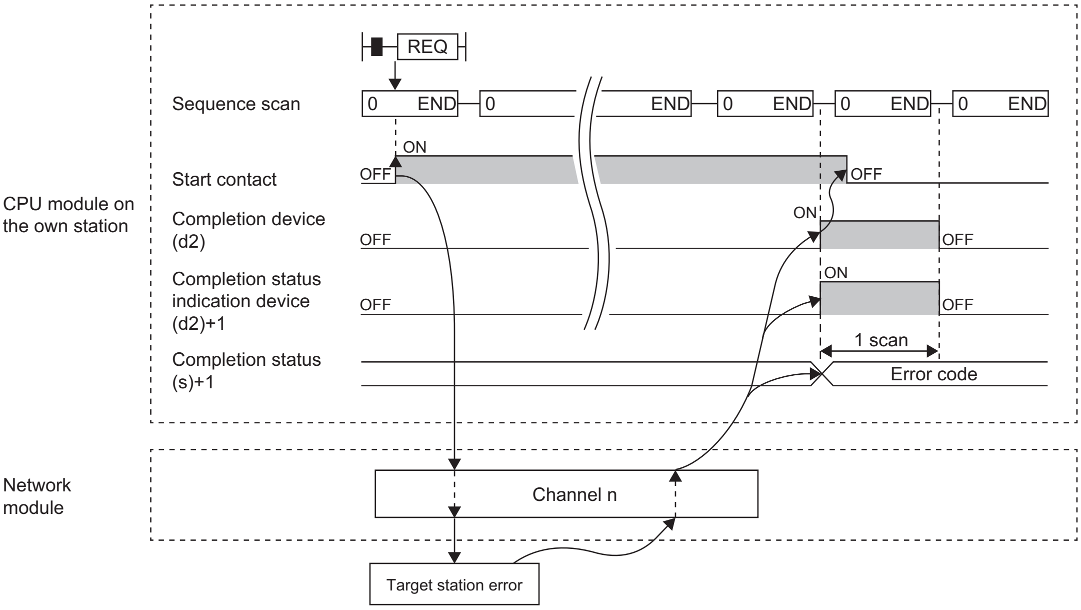
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the REQ instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the REQ instruction.
  - When completed successfully



- When completed with an error



- When the J.REQ/G.REQ instruction is used for execution, one complete cycle of write processing is followed by another while the write command is on. When the JP.REQ/GP.REQ instruction is executed, write processing is performed only once on the rising edge when the write command turns on.

Operation error

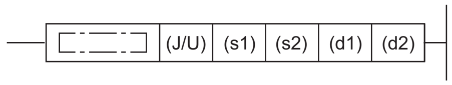
Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Reading/writing clock data

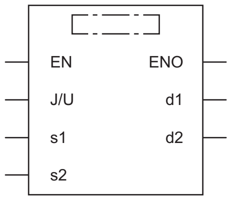
J(P).REQ, G(P).REQ

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

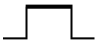
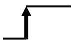
These instructions read or write clock data from/to the programmable controller of another station.

Ladder	ST
	ENO:=J_REQ(EN,J,s1,s2,d1,d2); ENO:=JP_REQ(EN,J,s1,s2,d1,d2); ENO:=G_REQ(EN,U,s1,s2,d1,d2); ENO:=GP_REQ(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.REQ G.REQ	
JP.REQ GP.REQ	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).REQ	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).REQ	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(s2)		Own station start device where request data is stored	—	Device name	ANY16[1]
(d1)		Own station start device for storing response data	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] 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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).REQ	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).REQ	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d2)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—	—

[*1] FX and FY cannot be used.

[*2] FD cannot be used.

[*3] T, ST, C, and FD cannot be used.

Point

The REQ instruction cannot be executed when the target station is ACPU.

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Error completion type	b15 … b7 … b4 … b0 0 (1) 0 1 0 1 (1) Error completion type (bit 7) Specify whether to set data when completed with an error. • 0: Do not set data in (s1)+11 and later at completion with an error. • 1: Set data in (s1)+11 and later at completion with an error.	0011H 0091H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	User
+3	Target station CPU type	Specify the CPU type of the target station. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU • 03D1H: Addressed to standby system CPU • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	User
+4	Target network number	Specify the network number of the target station. • 1 to 239: Network number	1 to 239	User
+5	Target station number	Specify the station number of the target station. (1) Station number specification [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 126: Master operating station • 1 to 120: Local station, intelligent device station, submaster station (2) Group number specification [Ethernet or CC-Link IE Controller Network] 0081H to 00A0H: All stations with group numbers 0001H to 0020H (can be specified only for clock data writing) (3) All-station specification 00FFH: All stations of target network number (broadcast (excluding the own station)) (can be specified only for clock data writing)	1 to 120, 125, 126, 0081H to 00A0H, 00FFH	User
+6	Not used	—	—	—

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	[CC-Link IE Controller Network or CC-Link IE Field Network] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
		[Ethernet] Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0 to (TCP resend timer value): The TCP resend timer value is assumed as the monitoring time. • (TCP resend timer value + 1) to 16383: (TCP resend timer value + 1) to 16383s	0 to 16383	User
+9	Request data length	Specify the number of request data (words). (Number of words of data stored in the request data storage device (s2)) • Reading clock data: 2 • Writing clock data: 6	2, 6	User
+10	Response data length	The number of response data (words) is stored. (Number of words of data stored in the response data storage device) • Reading clock data: 6 • Writing clock data: 2	2, 6	System
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state) (No information is stored if an error is detected in the own station.)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	1 to 239	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) [Ethernet or CC-Link IE Controller Network] • 1 to 120: Station number [CC-Link IE Field Network] • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	1 to 120, 125	System

■Request data

Operand: (s2)					
Device	**Item**	**Description**	**Reading clock data**	**Writing clock data**	**Set by**
+0	Request type	• 0001H: Reading clock data • 0011H: Writing clock data (with a station number specified in (s1)+5) • 0031H: Writing clock data (with All Station or a group specified in (s1)+5)	○	○	User
+1	Subrequest type	• 0002H: Reading clock data • 0001H: Writing clock data	○	○	User
+2	Change pattern	■Change pattern (bits 0 to 6) Specify which item from the upper byte in (s2)+2 to (s2)+5 should be written. • 0: Not changed • 1: Changed ■Changed year (bits 8 to 15) Specify the year (two lower digits) in BCD code.[*1] b15 … b8 b7 b6 b5 b4 b3 b2 b1 b0 Year (00H to 99H) | 0 | | | | | | | b6: Day of week, b5: Second, b4: Minute, b3: Hour, b2: Day, b1: Month, b0: Year	—	○	User
+3	Changed clock data	Upper 8 bits: Day (01H to 31H) Lower 8 bits: Month (01H to 12H)	—	○	User
+4		Upper 8 bits: Minute (00H to 59H) Lower 8 bits: Hour (00H to 23H)	—	○	User
+5		Upper 8 bits: Day of week (00H (Sunday) to 06H (Saturday)) Lower 8 bits: Second (00H to 59H)	—	○	User

[*1] Writing clock data using the REQ instruction cannot change the upper two digits of the year. If the year needs to be changed including the upper two digits as well, use the engineering tool to set the clock data.

■Response data

Operand: (d1)					
Device	**Item**	**Description**	**Reading clock data**	**Writing clock data**	**Set by**
+0	Request type	• 0081H: Reading clock data • 0091H: Writing clock data	○	○	System
+1	Subrequest type	• 0002H: Reading clock data • 0001H: Writing clock data	○	○	System
+2	Clock data that has been read	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Lower 2 digits of year (00H to 99H)	○	—	System
+3		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	○	—	System
+4		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	○	—	System
+5		Upper 8 bits: (00H) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	○	—	System

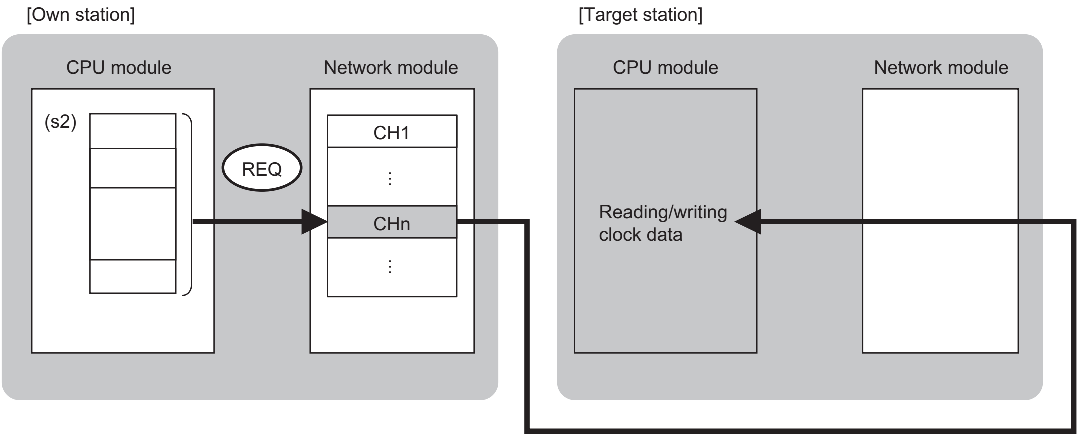
Point

- For details on the clock function, refer to the user's manual for the CPU module used.
- Clock data cannot be read or written when System Protect is applied to the CPU module of the target station.
- When clock data is read, a continuous area for six words is required for the response data storage device (d1).

Processing details

- These instructions send the request data specified by (S2) in the own station to the target station specified by the target network number and target station number of the control data. Upon completion of the request to the target station, the completion device specified by (d2) turns on.
- For the target stations that can be specified, refer to the following.

☞ Page 1449 Target networks and target station types



CH: Channel

- When executing multiple link dedicated instructions concurrently, be careful not to overlap the channels of the link dedicated instructions. Multiple link dedicated instructions specifying the same channel cannot be used concurrently.
- The execution status and the completion status of the REQ instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the REQ instruction completes, and turns off during the next END processing.

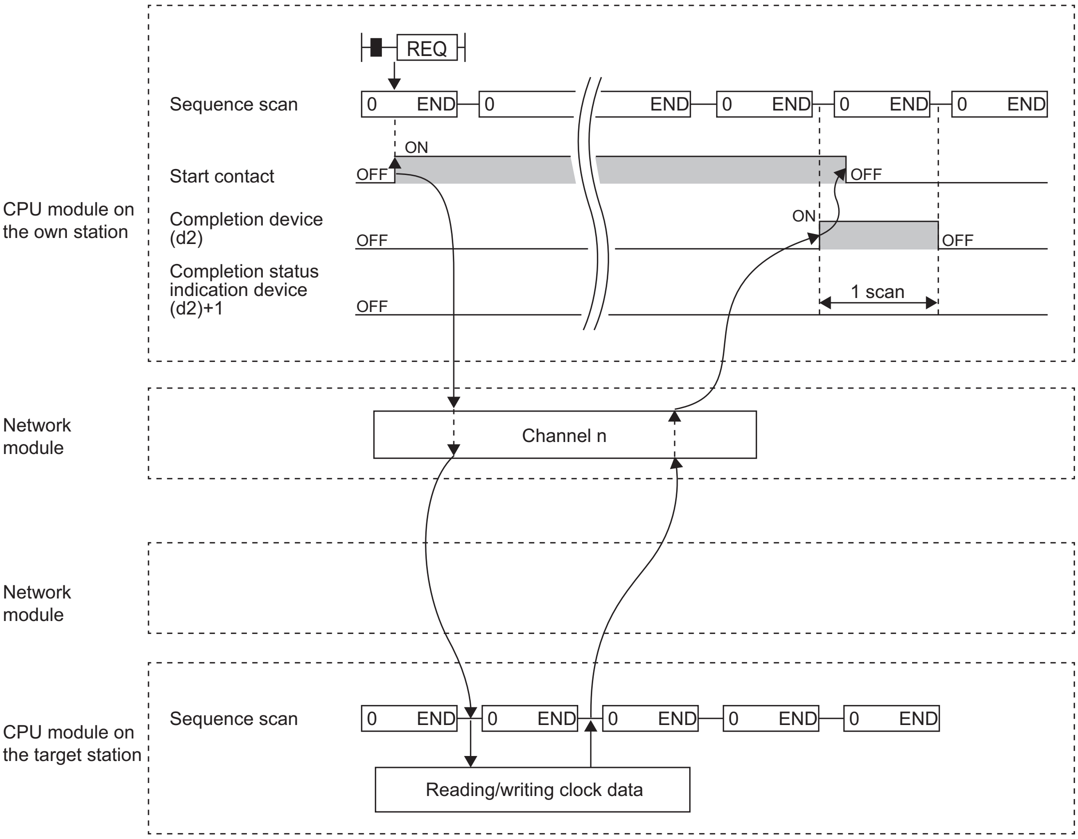
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the REQ instruction.

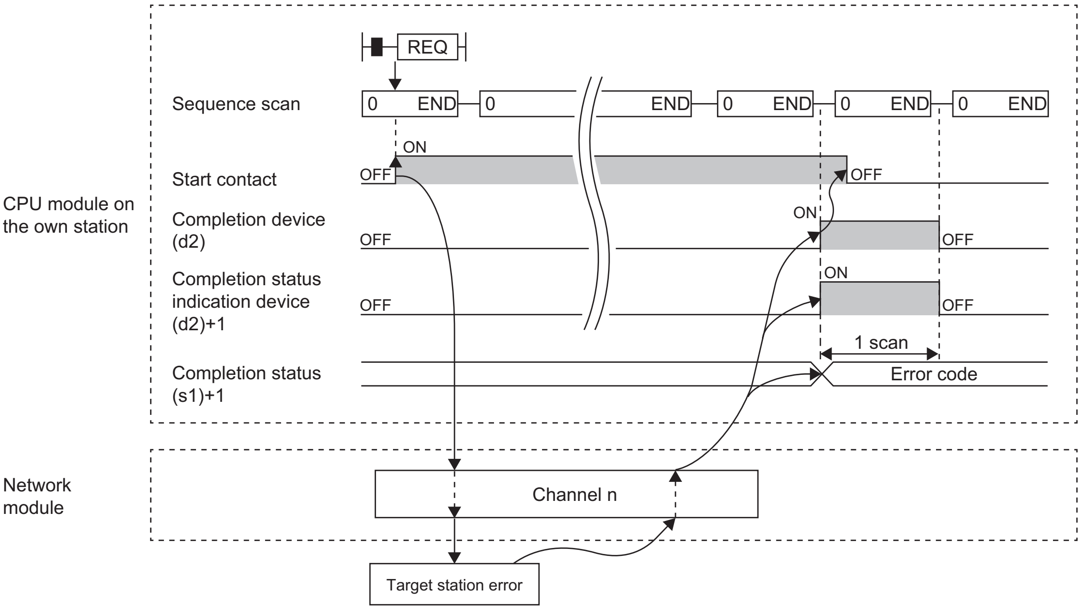
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the REQ instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the REQ instruction.
  - When completed successfully



  - When completed with an error



- When the J.REQ/G.REQ instruction is used for execution, one complete cycle of write processing is followed by another while the write command is on. When the JP.REQ/GP.REQ instruction is executed, write processing is performed only once on the rising edge when the write command turns on.

Operation error

Error code ((s1)+1)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

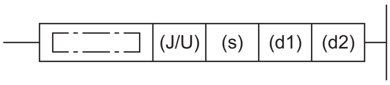
15.2 CC-Link Dedicated Instructions

Reading data from the target station

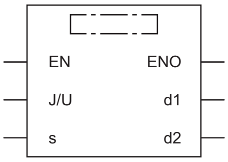
J(P).RIRD, G(P).RIRD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

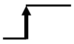
These instructions read the specified number of points of data from a device of the target station.

Ladder	ST
	ENO:=J_RIRD(EN,J,s,d1,d2); ENO:=JP_RIRD(EN,J,s,d1,d2); ENO:=G_RIRD(EN,U,s,d1,d2); ENO:=GP_RIRD(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.RIRD G.RIRD	
JP.RIRD GP.RIRD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RIRD	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RIRD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)		Own station start device for storing the read data (A continuous area for the read data length is required.)	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] 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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).RIRD	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RIRD	—	—	○	—	—	—	—	○	○	—	—	—	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+1	Target station number	Specifies the station number of the target station.	0 to 120	User
+2	Access/attribute code	Specify the access code and attribute code of the device to read. b15 ··· b8 b7 ··· b0 • b8 to b15: Access code • b0 to b7: Attribute code	Refer to "Access/ attribute code."	User
+3	Device number	Specify the start number of the device to read.	Within device range	User
+4	Number of read points	Specify the number of data in units of words to read from the device.	1 to 480	User

■Access/attribute code

Device category[*1]	Name	Date type		Unit	Access code[*2]	Attribute code[*2]
		Bit	Word			
Input relay	X	○	—	Hexadecimal	01H	05H
Output relay	Y	○	—	Hexadecimal	02H	05H
Internal relay	M	○	—	Decimal	03H	05H
Latch relay	L	○	—	Decimal	83H	05H
Link relay	B	○	—	Hexadecimal	23H	05H
Timer (contact)	T	○	—	Decimal	09H	05H
Timer (coil)	T	○	—	Decimal	0AH	05H
Timer (current value)	T	—	○	Decimal	0CH	05H
Retentive timer (contact)	ST	○	—	Decimal	89H	05H
Retentive timer (coil)	ST	○	—	Decimal	8AH	05H
Retentive timer (current value)	ST	—	○	Decimal	8CH	05H
Counter (contact)	C	○	—	Decimal	11H	05H
Counter (coil)	C	○	—	Decimal	12H	05H
Counter (current value)	C	—	○	Decimal	14H	05H
Data register[*3]	D	—	○	Decimal	04H	05H
Link register[*3]	W	—	○	Hexadecimal	24H	05H
File register	R	—	○	Decimal	84H	05H
Link special relay	SB	○	—	Hexadecimal	63H	05H
Link special register	SW	—	○	Hexadecimal	64H	05H
Special relay	SM	○	—	Decimal	43H	05H
Special register	SD	—	○	Decimal	44H	05H

*1 Any device other than the above cannot be accessed. To access a bit device, specify 0 or a multiple of 16.

*2 When the target station is other than the CC-Link IE Controller Network module and CC-Link IE Field Network master/local module, refer to the manual of the target station.

*3 None of D65536 and the subsequent extended data registers and of W10000 and the subsequent extended link registers are accepted.

Point

The RIRD instruction can set the arrival monitoring time and the number of resends in the following link special registers (SW).

- RIRD/RIWT instruction: Arrival monitoring time (SW0009)
- RIRD/RIWT instruction: Number of resends (SW000B)

When "RIRD/RIWT instruction: Number of resends (SW000B)" is set, the time taken for the RIRD instruction to be completed with an error is determined by the following.

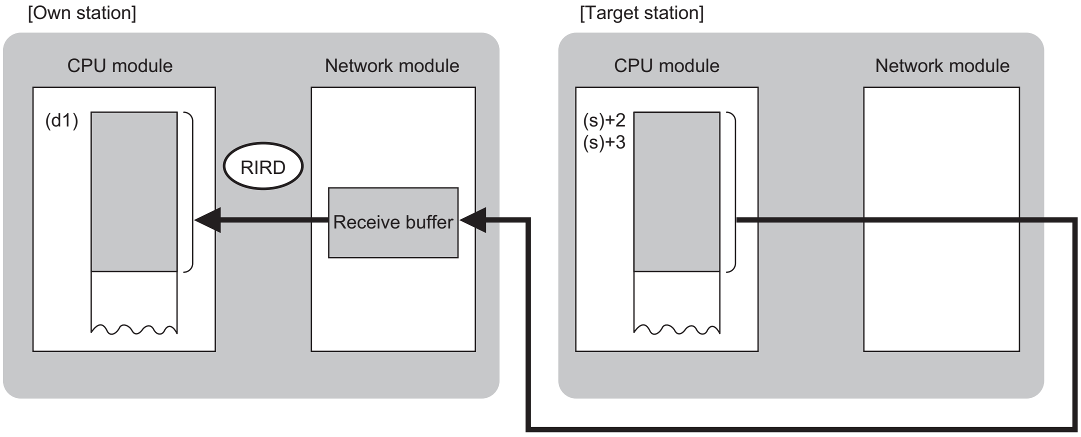
(RIRD/RIWT instruction: Number of resends + 1)×RIRD/RIWT instruction: Arrival monitoring time

However, if the target station (relay source station when addressed to another network) has been disconnected at the time of RIRD instruction execution, no retry is executed. If the RIRD instruction is completed with an error, re-execute the RIRD instruction after the target station (relay source station when addressed to another network) returns.

If an error (error code: E504H) occurs in CC-Link IE Controller Network, a retry cannot be performed. Wait until the instruction completes with an error, and execute the RIRD instruction again.

Processing details

- These instructions read the number of words of data specified by (s)+4 from the start device of the target station specified by (s)+2 and (s)+3, and put the data in the word device in the own station specified by (d1) and after. Specify the target station in (s)+1. Upon completion of reading the data from the target station, the completion device specified by (d2) turns on.



- The execution status and the completion status of the RIRD instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the RIRD instruction completes, and turns off during the next END processing.

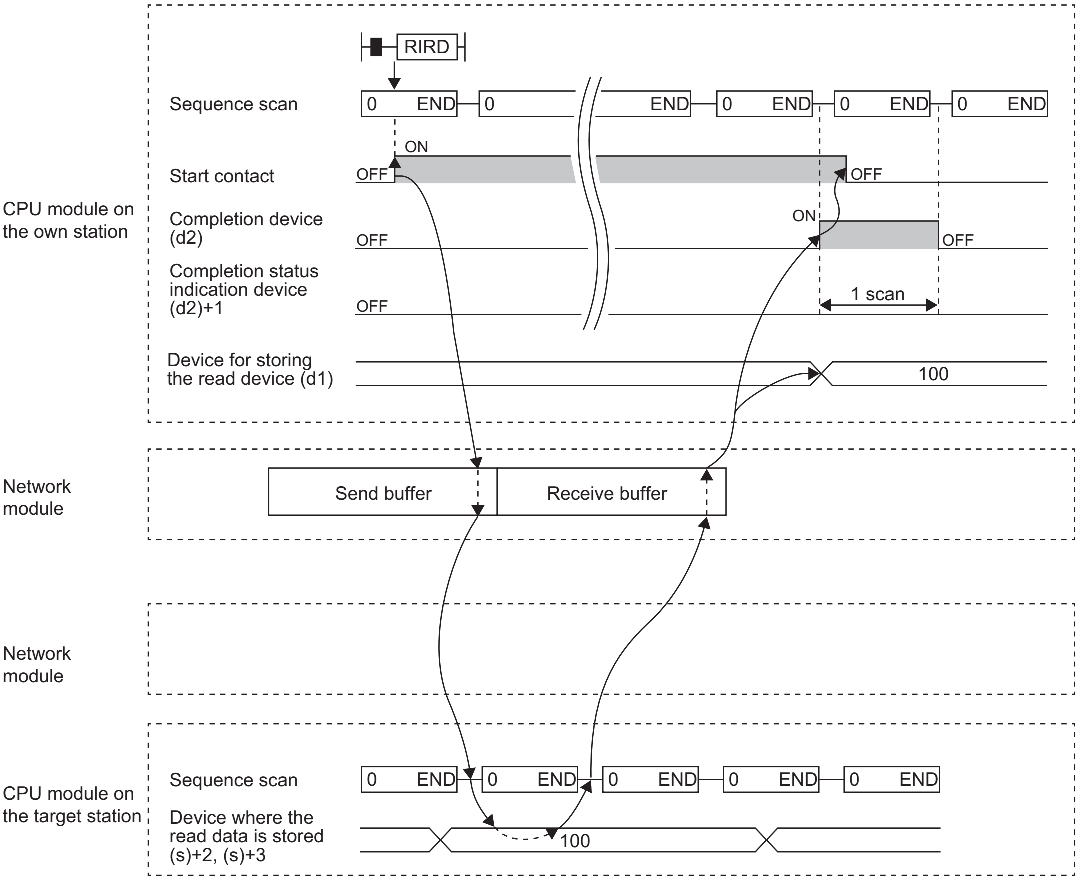
  - Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the RIRD instruction.

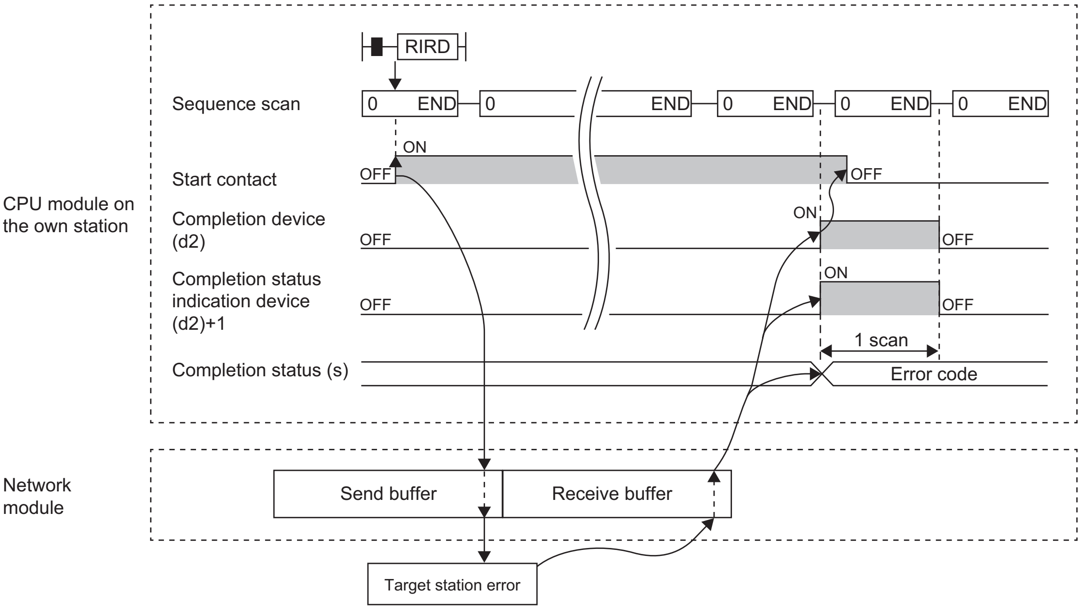
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the RIRD instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the RIRD instruction.
  - When completed successfully



- When completed with an error



Operation error

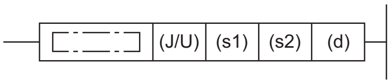
Error code ((s)+0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

Writing data to the target station

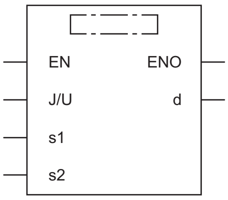
J(P).RIWT, G(P).RIWT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

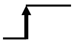
These instructions write the specified number of points of data to a device of the target station.

Ladder	ST
	ENO:=J_RIWT(EN,J,s1,s2,d); ENO:=JP_RIWT(EN,J,s1,s2,d); ENO:=G_RIWT(EN,U,s1,s2,d); ENO:=GP_RIWT(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
J.RIWT G.RIWT	
JP.RIWT GP.RIWT	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RIWT	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RIWT	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16*1
(s2)		Own station start device for storing written data	—	Device name	ANY16*1
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).RIWT	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RIWT	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Completion status	The completion status is stored upon completion of the instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+1	Target station number	Specifies the station number of the target station.	0 to 120	User
+2	Access/attribute code	Specify the access code and attribute code of the device to be written. b15 … b8 b7 … b0 • b8 to b15: Access code • b0 to b7: Attribute code	Refer to "Access/ attribute code."	User
+3	Device number	Specify the start number of the device to be written.	Within device range	User
+4	Number of write points	Specify the number of data in units of words to read from the device.	1 to 480	User

■Access/attribute code

Device category[*1]	Name	Date type		Unit	Access code[*2]	Attribute code[*2]
		Bit	Word			
Input relay	X	○	—	Hexadecimal	01H	05H
Output relay	Y	○	—	Hexadecimal	02H	05H
Internal relay	M	○	—	Decimal	03H	05H
Latch relay	L	○	—	Decimal	83H	05H
Link relay	B	○	—	Hexadecimal	23H	05H
Timer (contact)	T	○	—	Decimal	09H	05H
Timer (coil)	T	○	—	Decimal	0AH	05H
Timer (current value)	T	—	○	Decimal	0CH	05H
Retentive timer (contact)	ST	○	—	Decimal	89H	05H
Retentive timer (coil)	ST	○	—	Decimal	8AH	05H
Retentive timer (current value)	ST	—	○	Decimal	8CH	05H
Counter (contact)	C	○	—	Decimal	11H	05H
Counter (coil)	C	○	—	Decimal	12H	05H
Counter (current value)	C	—	○	Decimal	14H	05H
Data register[*3]	D	—	○	Decimal	04H	05H
Link register[*3]	W	—	○	Hexadecimal	24H	05H
File register	R	—	○	Decimal	84H	05H
Link special relay	SB	○	—	Hexadecimal	63H	05H
Link special register	SW	—	○	Hexadecimal	64H	05H
Special relay	SM	○	—	Decimal	43H	05H
Special register	SD	—	○	Decimal	44H	05H

*1 Any device other than the above cannot be accessed. To access a bit device, specify 0 or a multiple of 16.

*2 When the target station is other than the CC-Link IE Controller Network module and CC-Link IE Field Network master/local module, refer to the manual of the target station.

*3 None of D65536 and the subsequent extended data registers and of W10000 and the subsequent extended link registers are accepted.

Point

The RIWT instruction can set the arrival monitoring time and the number of resends in the following link special registers (SW).

- RIRD/RIWT instruction: Arrival monitoring time (SW0009)
- RIRD/RIWT instruction: Number of resends (SW000B)

When "RIRD/RIWT instruction: Number of resends (SW000B)" is set, the time taken for the RIWT instruction to be completed with an error is determined by the following.

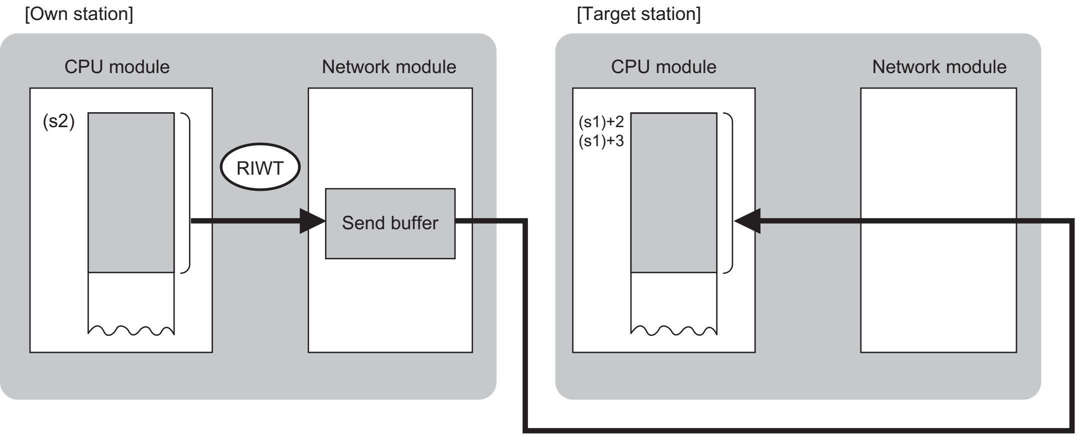
(RIRD/RIWT instruction: Number of resends + 1)×RIRD/RIWT instruction: Arrival monitoring time

However, if the target station (relay source station when addressed to another network) has been disconnected at the time of RIWT instruction execution, no retry is executed. If the RIWT instruction is completed with an error, re-execute the RIWT instruction after the target station (relay source station when addressed to another network) returns.

If an error (error code: E504H) occurs in CC-Link IE Controller Network, a retry cannot be performed. Wait until the instruction completes with an error, and execute the RIWT instruction again.

Processing details

- These instructions read the number of words of data specified by (s1)+4 from the start device of the own station specified by (s2), and write the data in the device in the target station specified by (s1)+2 and (s1)+3. Specify the target station in (s1)+1. Upon completion of writing the data, the completion device specified by (d) turns on.



- The execution status and the completion status of the RIWT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the RIWT instruction completes, and turns off during the next END processing.

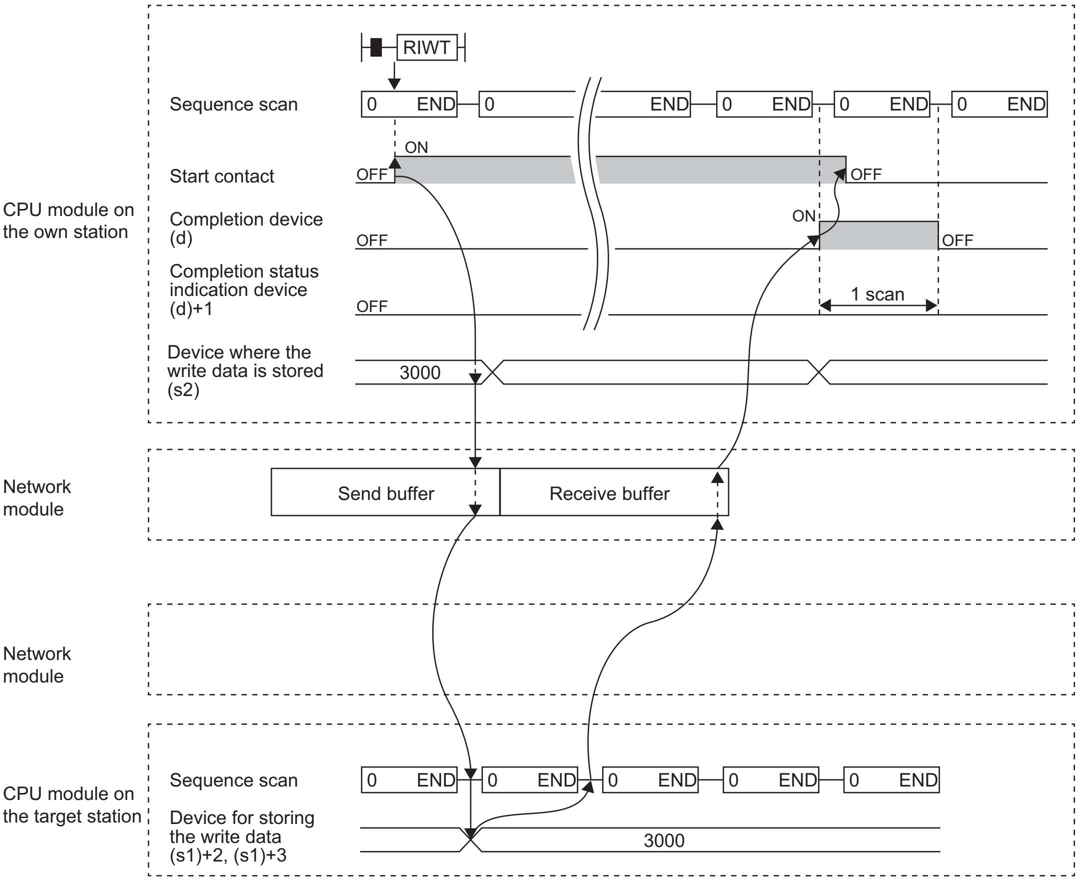
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the RIWT instruction.

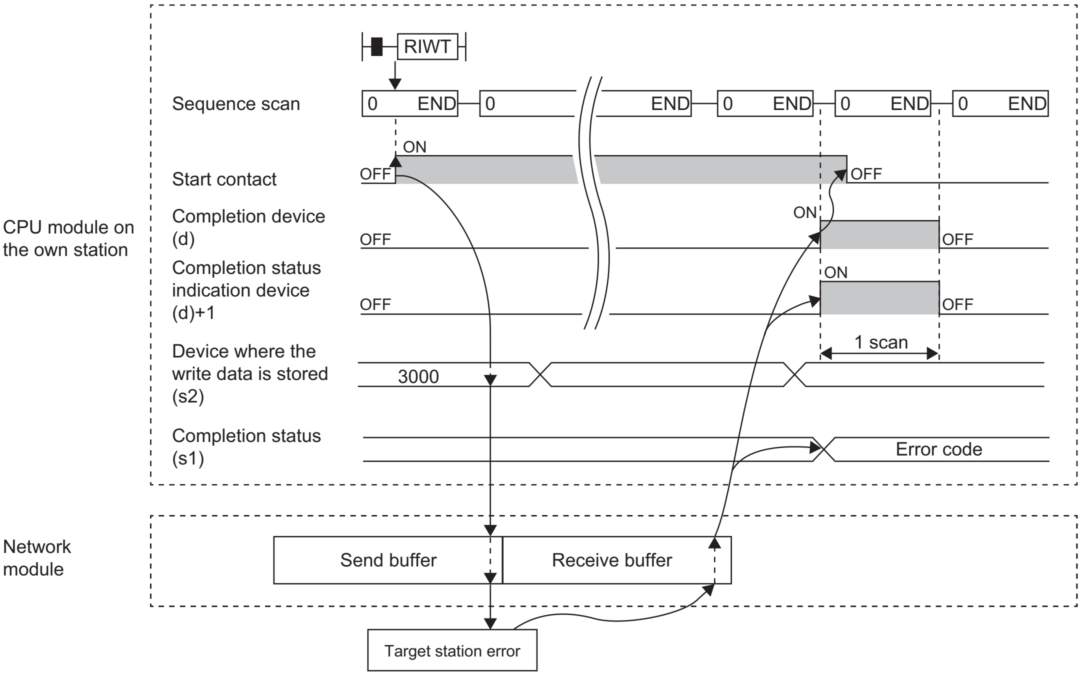
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the RIWT instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the RIWT instruction.
  - When completed successfully



- When completed with an error



Operation error

Error code ((s1)+0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

16 ETHERNET INSTRUCTIONS

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. For the instructions when MELSEC-Q series modules are used, refer to the manuals for each module used and create programs.
For precautions when using modules, refer to the following.
📖 MELSEC iQ-R Module Configuration Manual

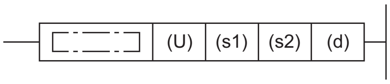
16.1 Open/Close Processing Instructions

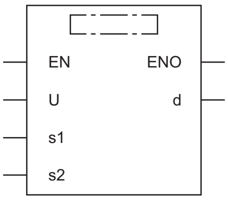
Opening a connection

GP.CONOPEN

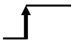
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

This instruction establishes (opens) a connection with an external device for data communication.

Ladder	ST
	ENO:=GP_CONOPEN(EN,U,s1,s2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
GP.CONOPEN	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Connection No.	1 to 128 (1 to 16: Port 1 fixed buffer communications, 17 to 64: Port 1 socket communications, 65 to 128: Port 2 socket communications)	16-bit unsigned binary	ANY16
(s2)	Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(s2)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(d)	○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

[1] FX and FY cannot be used.

[2] FD cannot be used.

[3] T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution type/End type	b15 … b8 … b0 0 (1) 0 (1) Execution type[*1] Specify whether to use the parameter value set by the engineering tool or the value set in (s2)+4 to (s2)+14 of control data for opening a connection. • 0: Performs open processing according to the "external device connection configuration" set by the engineering tool. • 1: Performs open processing according to the setting in (s2)+4 to (s2)+14 of control data. If the CP.CONOPEN instruction is executed by setting the value for the external device connection configuration of the engineering tool and also setting device (s2)+0 to 1, the instruction performs open processing according to the value set in its control data. Even if a value is set in (s2)+4 and later after setting (s2)+0 to 0, the value is ignored.	0000H 0100H	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
+2	System area	—	—	—
+3	System area	—	—	—
+4	Application setting area	Specify the application of a connection. ■Fixed buffer communications b15 … b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 0 (8) (7) (6) (5) (4) (3) 0 (2) (1) (1) Application of fixed buffer (b0) • 0: For sending • 1: For receiving (2) Communication code (b1) • 0: Binary code • 1: ASCII code (3) Online program change (b3) • 0: Disable • 1: Enable (4) Pairing open (b4) • 0: Disable pairing open. • 1: Enable pairing open.[*2] (5) Communication method (protocol) (b5) • 0: TCP/IP • 1: UDP/IP (6) Fixed buffer communications with/without procedure (b6 and b7) • 00: With procedure • 01: Without procedure (7) Alive check (b8 and b9) • 00: KeepAlive[*4] • 01: Alive check with UDP[*5] • 10: Disable the alive check. (8) Opening method (b10 and b11) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open ■Socket communications b15 … b12 b11 b10 b9 … b6 b5 b4 b3 b2 … b0 0 (3) 0 (2) 0 (1) 0 (1) Online program change (b3) • 0: Disable • 1: Enable (2) Communication method (protocol) (b5) • 0: TCP/IP • 1: UDP/IP (3) Opening method (b10 and b11) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open	Left	User

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+5	Own node port number	Specify the port number of the own node. (Port numbers 5000 to 5009 is reserved for the system and cannot be used.)	1024 to 4999 5010 to 65534 (0400H to 1387H, 1392H to FFFEH)	User
+6 +7	IP address of external device[6]	Specify the IP address (IPv4) of an external device. • The IP address is stored in (s2)+6 and (s2)+7. • To enable broadcast, store FFFFFFFFH.	00000001H to FFFFFFFFH	User
+8 to +13	System area	—	—	—
+14	Destination port number[6]	Specify the destination port number. When receiving data from all port numbers, store FFFFH.[3]	1024 to 65535 (0400H to FFFFH)	User

[1] When the execution type is set to 0, the device areas in (s2)+4 to (s2)+14 are used by the system and the user must not use the areas.

[2] This item can be set when the connection number in (s1) is one from 1 to 7 and 9 to 15.

[3] This specification is enabled when UDP/IP is specified for the communication method (protocol).
Data cannot be sent through the connection for which 65535 (FFFFH) is specified. Specify a value from 1 to 65534 to send data.

[4] This specification is enabled when TCP/IP is specified for the communication method (protocol).

[5] This specification is enabled when UDP/IP is specified for the communication method (protocol).

[6] Settings are ignored if the open method (bits 10 and 11) specified by (s2)+4 is "10: Unpassive open".

Processing details

The GP.CONOPEN instruction is used for socket communications or fixed buffer communications.

For the combinations of available settings for connection opening, refer to the following.

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

Operation error

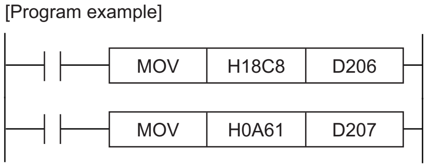
Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

Point

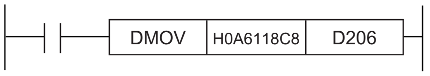
The following figure shows an IP address setting example.

Example: IP address "10.97.24.200 (0A.61.18.C8)"

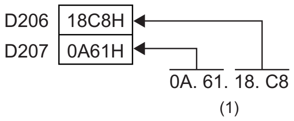
[Program example]



or



[Execution result]



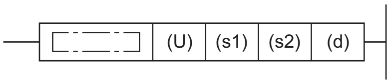
(1) The data is stored from the lower byte.

Closing a connection

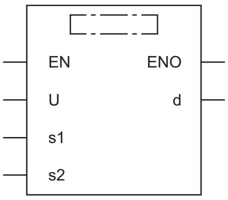
GP.CONCLOSE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

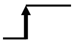
This instruction disconnects (closes) the connection from the external device during data communication.

Ladder	ST
	ENO:=GP_CONCLOSE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.CONCLOSE	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Connection No.	1 to 128 (1 to 16: Port 1 fixed buffer communications, 17 to 64: Port 1 socket communications, 65 to 128: Port 2 socket communications)	16-bit unsigned binary	ANY16
(s2)	Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—
(s2)	—	—	○[*2]	—	—	—	—	○	—	—	—	—
(d)	○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—

[*1] FX and FY cannot be used.

[*2] FD cannot be used.

[*3] T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+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

Processing details

The GP.CONCLOSE instruction is used for socket communications or fixed buffer communications.

Operation error

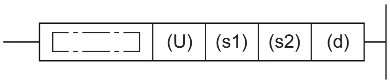
Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

Opening a connection

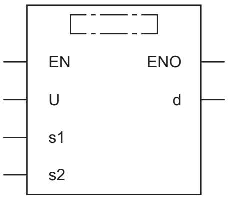
GP.OPEN, ZP.OPEN

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

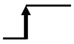
This instruction establishes (opens) a connection with an external device for data communication.

Ladder	ST
	ENO:=GP_OPEN(EN,U,s1,s2,d); ENO:=ZP_OPEN(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.OPEN ZP.OPEN	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.OPEN	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.OPEN	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Connection No.	1 to 16	16-bit unsigned binary	ANY16
(s2)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.OPEN	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.OPEN	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	Execution type/End type	Specify whether to use the parameter value set by the engineering tool or the value set in (s2)+2 to (s2)+6 of control data for opening a connection. • 0000H: Performs open processing according to the "external device connection configuration" set by the engineering tool. • 8000H: Performs open processing according to the setting in (s2)+2 to (s2)+6 of control data.	0000H 8000H	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
+2	Application setting area	Specify the application of a connection. b15 b14 b13 ··· b11 b10 b9 b8 b7 b6 ··· b2 b1 b0 (6) 0 (5) (4) (3) 0 (2) (1) (1) Application of fixed buffer (b0) • 0: For sending • 1: For receiving (2) Target station alive check (b1) • 0: Disable the alive check. • 1: Alive check[1] (3) Pairing open (b7) • 0: Disable pairing open. • 1: Enable pairing open.[2] (4) Protocol (b8) • 0: TCP/IP • 1: UDP/IP (5) Fixed buffer communications with/without procedure (b9 and b10) • 00: With procedure • 01: Without procedure • 10: Communication protocol (6) Opening method (b14 and b15) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open	Left	User
+3	Own station port number.	Specify the port number of the own station. (Port numbers 5000 to 5009 is reserved for the system and cannot be used.)	1024 to 4999, 5010 to 65534 (0400H to 1387H, 1392H to FFFEH)	User
+4 to +5	IP address of external device[3]	Specify the IP address (IPv4) of an external device. • The IP address is stored in (s2)+4 and (s2)+5. • To enable broadcast, specify FFFFFFFFH.	00000001H to FFFFFFFFH	User
+6	Destination port number[3]	Specify the destination port number. When receiving data from all port numbers, specify FFFFH.	1 to 65534, 65535 (0001H to FFFEH, FFFFH)	User
+7 to +9	System area	—	—	—

[1] When the TCP/IP protocol is used, the alive check method is fixed to KeepAlive. (When UDP/IP is used, it is fixed to Ping.)

[2] This item can be set when the connection number set in (s1) is one from 1 to 7 and 9 to 15.

[3] Settings are ignored if the open method (bits 14 and 15) specified by (s2)+2 is "10: Unpassive open".

Processing details

- These instructions open the connection specified by (s1) of the module specified by (U).
- The selection of the setting value used for open processing is specified by (s2)+0. (Specify whether to use the parameter value set by the engineering tool or the value set in (s2)+2 to (s2)+16 of control data.)
- The execution status and the completion status of the OPEN instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the OPEN instruction completes, and turns off during the next END processing.

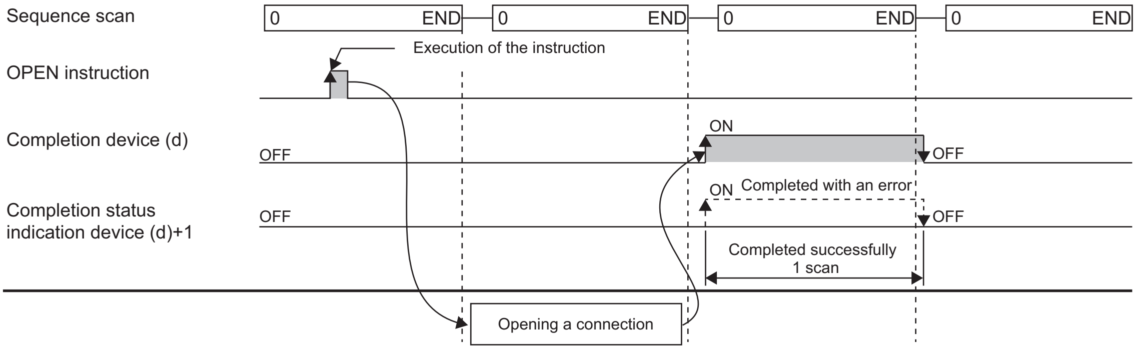
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the OPEN instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the OPEN instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the OPEN instruction.



- The OPEN instruction is executed on the rising edge (OFF to ON) of the open command.

Precautions

For the same connection, do not perform open and close processing using the OPEN and CLOSE instructions concurrently with open and close processing using other means. Concurrent use results in a malfunction.

Operation error

Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

Restriction

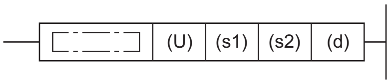
- The communication means of the target connection is fixed buffer communications (with procedure), fixed buffer communications (without procedure), or predefined protocol. Make settings in (b9, b10) (fixed buffer communications with/without procedure) of (s2)+2 in control data.
- When the protocol is set to TCP/IP, the alive check method is fixed to KeepAlive.
- The connection numbers that can be specified range from 1 to 16. Connection number 17 and after cannot be specified.
- If no parameter data is set in "external device connection configuration" of the engineering tool, the communication data code becomes "binary". If one or more parameter data are set, the instruction follows the value set in "Communication data code".
- If no parameter data is set in "external device connection configuration" of the engineering tool, the opening method becomes "Do not open with the program". If one or more parameter data are set, the instruction follows the value set in "Opening method".

Closing a connection

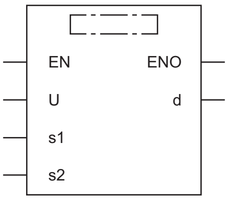
GP.CLOSE, ZP.CLOSE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

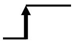
These instructions disconnect (close) the connection from the external device during data communication.

Ladder	ST
	ENO:=GP_CLOSE(EN,U,s1,s2,d); ENO:=ZP_CLOSE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.CLOSE ZP.CLOSE	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.CLOSE	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.CLOSE	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Connection No.	1 to 16	16-bit unsigned binary	ANY16
(s2)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.CLOSE	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.CLOSE	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—
(s2)		—	—	○[*2]	—	—	—	—	○	—	—	—	—
(d1)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—

[*1] FX and FY cannot be used.

[*2] FD cannot be used.

[*3] T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+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

Processing details

- These instructions close the connection specified by (s1) of the module specified by (U).
- The execution status and the completion status of the CLOSE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the CLOSE instruction completes, and turns off during the next END processing.

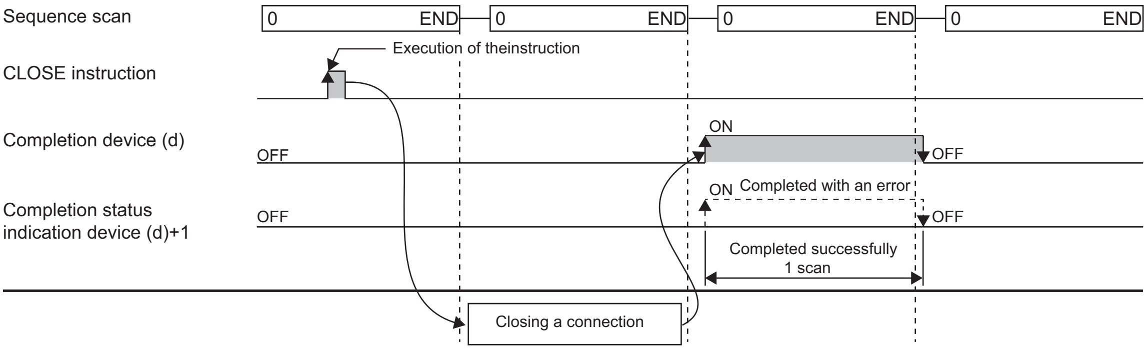
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the CLOSE instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the CLOSE instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the CLOSE instruction.



- The CLOSE instruction is executed on the rising edge (OFF to ON) of the close command.

Precautions

- For the same connection, do not perform open and close processing using the OPEN and CLOSE instructions concurrently with open and close processing using other means. Concurrent use results in a malfunction.
- If a connection for which the OPEN instruction is in execution is specified in TCP Unpassive/Fullpassive open mode, an error (C1B2H: OPEN/CLOSE instruction is in execution for the specified connection) occurs.

Operation error

Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

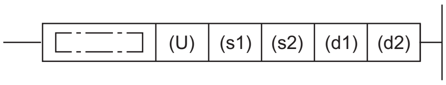
16.2 Socket Communications Instructions

Reading receive data

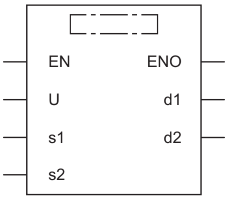
GP.SOCRCV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

This instruction reads receive data from the external device through socket communications.

Ladder	ST
	ENO:=GP_SOCRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCRCV	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Connection No.	17 to 128 (17 to 64: Port 1 socket communications, 65 to 128: Port 2 socket communications)	16-bit unsigned binary	ANY16
(s2)	Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)	Own station start device for storing the receive data	—	Device name	ANY16[1]
(d2)	Device of the own station, which 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] 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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—
(s2)	—	—	○[*2]	—	—	—	—	○	—	—	—	—
(d1)	—	—	○[*2]	—	—	—	—	○	—	—	—	—
(d2)	○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+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

■Receive data

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the socket communications receive data area is stored.	—	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.	—	System

Processing details

The GP.SOCRCV instruction is used for socket communications.

Operation error

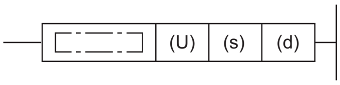
Error code ((s2)+1)	Description
C000H to CFFFH	MELSEC iQ-R Ethernet User's Manual (Application)

Reading receive data (for interrupt programs)

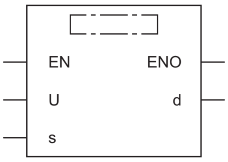
G.SOCRCVS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

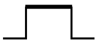
This instruction reads receive data from the external device through socket communications (for interrupt programs).

Ladder	ST
	ENO:=G_SOCRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.SOCRCVS	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s)	Connection No.	17 to 128 (17 to 64: Port 1 socket communications, 65 to 128: Port 2 socket communications)	16-bit unsigned binary	ANY16
(d)	Own station start device for storing the receive data	—	Device name	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d)	—	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

■Receive data

Operand: (d)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the socket communications receive data area is stored.	—	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.	—	System

Processing details

The G.SOCRCVS instruction is used for socket communications.

Operation error

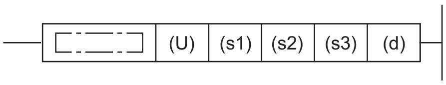
Error code (SD0)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

Sending data

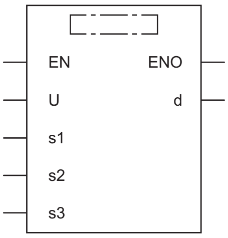
GP.SOCSND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction sends data to the external device through socket communications.

Ladder	ST
	ENO:=GP_SOCSND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCSND	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Connection No.	17 to 128 (17 to 64: Port 1 socket communications, 65 to 128: Port 2 socket communications)	16-bit unsigned binary	ANY16
(s2)	Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(s3)	Own station start device where send data is stored	—	Device name	ANY16[1]
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)	—	—	○*2	—	—	—	—	○	—	—	—	—
(s3)	—	—	○*2	—	—	—	—	○	—	—	—	—
(d)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+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

■Send data

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Send data length	Specify the send data length. (Number of bytes)	1 to 10238	User
+1 to +□	Send data	Specify the send data.	—	User

Processing details

The GP.SOCSND instruction is used for socket communications.

Operation error

Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

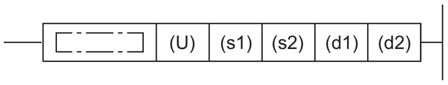
16.3 Fixed Buffer Communications Instructions

Reading receive data

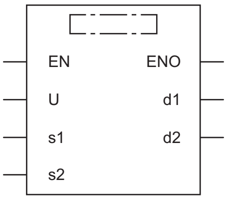
GP.BUFRCV, ZP.BUFRCV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read receive data from the external device through fixed buffer communications

Ladder	ST
 (U) (s1) (s2) (d1) (d2)	ENO:=GP_BUFRCV(EN,U,s1,s2,d1,d2); ENO:=ZP_BUFRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.BUFRCV ZP.BUFRCV	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.BUFRCV	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.BUFRCV	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Connection No.	1 to 16	16-bit unsigned binary	ANY16
(s2)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d1)		Own station start device for storing the receive data	—	Device name	ANY16[*1]
(d2)		Device of the own station, which 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] 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

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	$	
(U)	GP.BUFRCV	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.BUFRCV	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+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

■Receive data

Operand: (d1)

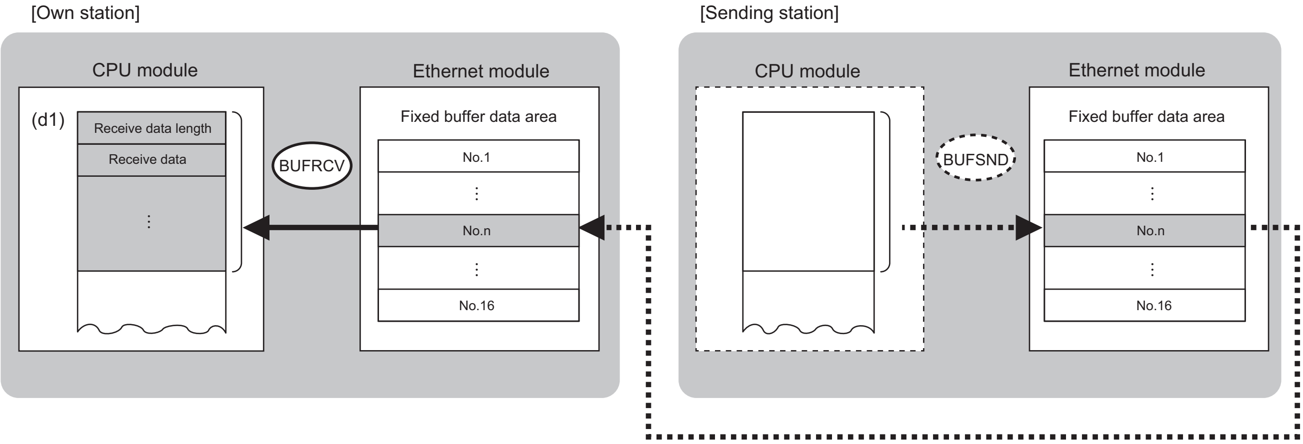
Device	Item	Description		Setting range	Set by
+0	Receive data length	The length of the data read from the fixed buffer receive data area is stored. (The data length is represented in words or bytes according to the procedure of fixed buffer communications.)	With procedure (binary): Number of words	1 to 5113 (1 to 1017)*1	System
			With procedure (ASCII): Number of words	1 to 2556 (1 to 508)*1	
			Without procedure (binary): Number of bytes	1 to 10238 (1 to 2046)*1	
+1 to +□	Receive data	The data read from the fixed buffer receive data areas is stored sequentially in ascending order of addresses.		—	System

*1 This setting range is applicable when the network type is "Q-compatible Ethernet".

Processing details

- These instructions read the receive data (fixed buffer communications area) of the connection specified by (s1) of the module specified by (U). These instructions can be used only for the connections for which "fixed buffer communication (with/without procedure)" is set as the communication means.

[Reading receive data from the sending station to the own station]



No.: Connection No.

- The execution status and the completion status of the BUFRCV instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the BUFRCV instruction completes, and turns off during the next END processing.

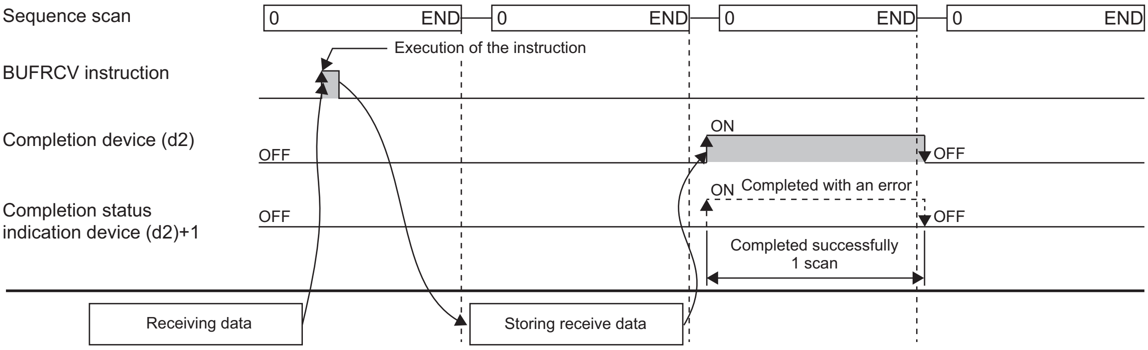
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the BUFRCV instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the BUFRCV instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the BUFRCV instruction.



- The BUFRCV instruction is executed on the rising edge (OFF to ON) of the socket/fixed buffer receive status signal (Un\G1900016 to Un\G1900023).

Restriction

When the BUFRCV instruction reads receive data from the same connection, it cannot be used in combination with the BUFRCVS instruction (for use in interrupt programs).

Operation error

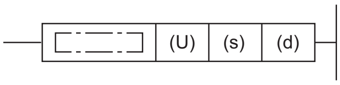
Error code ((s2)+1)	Description
C000H to CFFFH	MELSEC iQ-R Ethernet User's Manual (Application)

Reading receive data (for interrupt programs)

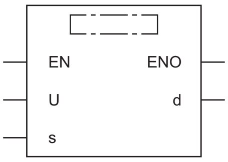
G.BUFRCVS, Z.BUFRCVS

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions read receive data from the external device through fixed buffer communications (for interrupt programs).

Ladder	ST
	ENO:=G_BUFRCVS(EN,U,s,d); ENO:=Z_BUFRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.BUFRCVS Z.BUFRCVS	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.BUFRCVS	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.BUFRCVS	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Connection No.	1 to 16	16-bit unsigned binary	ANY16
(d)		Own station start device for storing the receive data	—	Device name	ANY16[1]
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.BUFRCVS	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.BUFRCVS	—	—	○	—	—	—	—	○	—	—	○	○
(s)		○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(d)		—	—	○[2]	—	—	—	—	○	—	—	—	—

[1] FX and FY cannot be used.

[2] FD cannot be used.

■Receive data

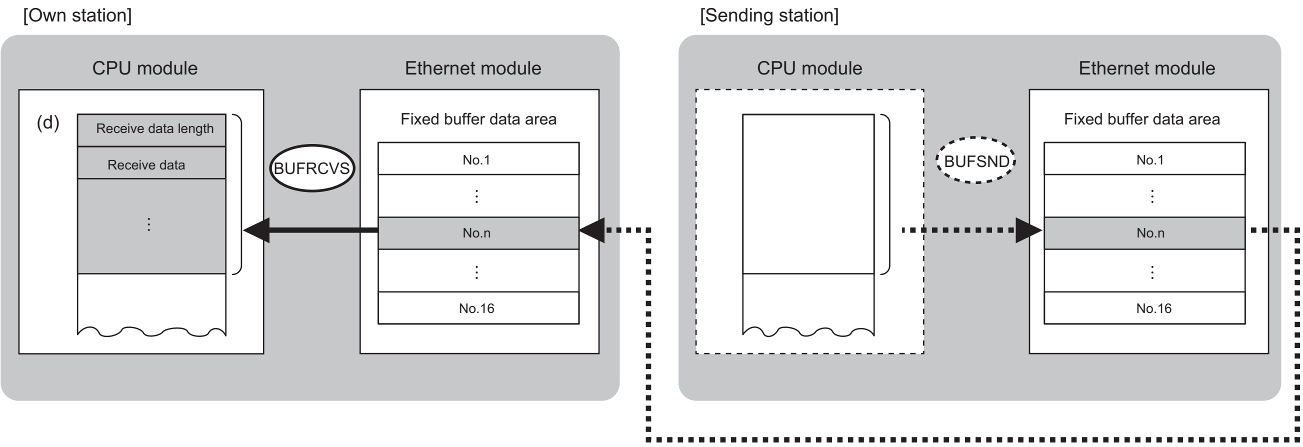
Operand: (d)					
Device	**Item**	**Description**		**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the fixed buffer receive data area is stored. (The data length is represented in words or bytes according to the procedure of fixed buffer communications.)	With procedure (binary): Number of words	1 to 5113 (1 to 1017)[1]	System
			With procedure (ASCII): Number of words	1 to 2556 (1 to 508)[1]	
			Without procedure (binary): Number of bytes	1 to 10238 (1 to 2046)[1]	
+1 to +□	Receive data	The data read from the fixed buffer receive data areas is stored sequentially in ascending order of addresses.		—	System

[1] This setting range is applicable when the network type is "Q-compatible Ethernet".

Processing details

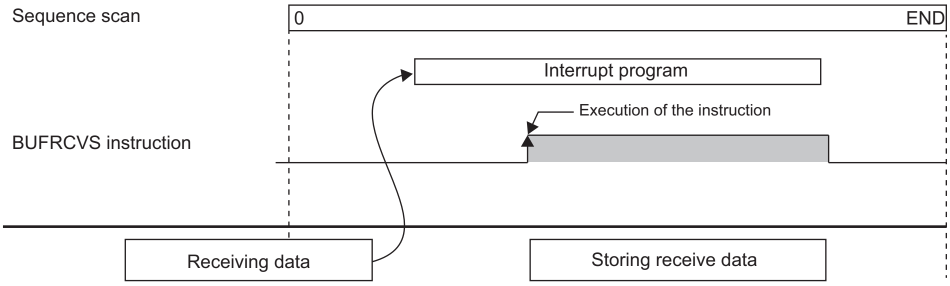
- These instructions read the receive data (fixed buffer communications area) of the connection specified by (s) of the module specified by (U). These instructions can be used only for the connections for which "fixed buffer communication (with/without procedure)" is set as the communication means.

[Reading receive data from the sending station to the own station]



No.: Connection No.

- The BUFRCVS instruction is used in interrupt programs and processing is completed in a single scan.



Restriction

When the BUFRCVS instruction reads receive data from the same connection, it cannot be used in combination with the BUFRCV instruction.

Operation error

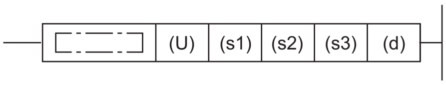
Error code (SD0)	Description
C000H to CFFFH	MELSEC iQ-R Ethernet User's Manual (Application)

Sending data

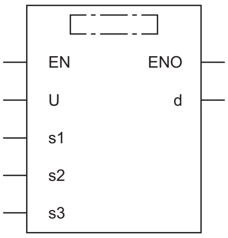
GP.BUFSND, ZP.BUFSND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions send data to the external device through fixed buffer communications.

Ladder	ST
	ENO:=GP_BUFSND(EN,U,s1,s2,s3,d); ENO:=ZP_BUFSND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.BUFSND ZP.BUFSND	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.BUFSND	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.BUFSND	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Connection No.	1 to 16	16-bit unsigned binary	ANY16
(s2)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(s3)		Own station start device where send data is stored	—	Device name	ANY16[1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.BUFSND	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.BUFSND	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—
(s3)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+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

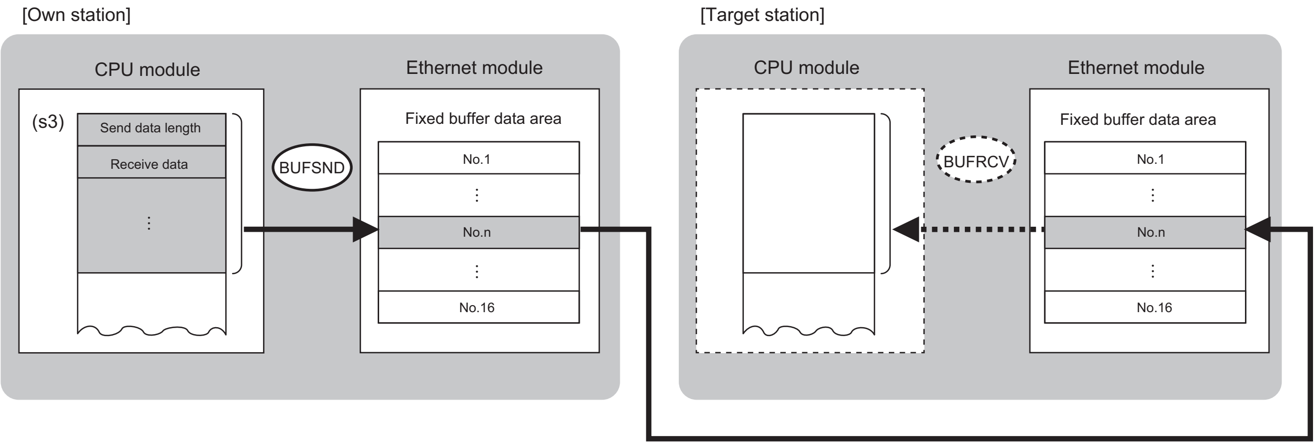
■Send data

Operand: (s3)					
Device	**Item**	**Description**		**Setting range**	**Set by**
+0	Send data length	Specify the send data length. (Specify the data length in words or bytes according to the procedure of fixed buffer communications.)	With procedure (binary): Number of words	1 to 5113 (1 to 1017)*1	User
			With procedure (ASCII): Number of words	1 to 2556 (1 to 508)*1	
			Without procedure (binary): Number of bytes	1 to 10238 (1 to 2046)*1	
+1 to +□	Send data	Specify the send data.		—	User

*1 This setting range is applicable when the network type is "Q-compatible Ethernet".

Processing details

- This instruction sends the data in the device specified by (s3) to the external device of the connection specified by (s1) of the module specified by (U). These instructions can be used only for the connections for which "fixed buffer communication (with/without procedure)" is set as the communication means.



No.: Connection No.

- The execution status and the completion status of the BUFSND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the BUFSND instruction completes, and turns off during the next END processing.

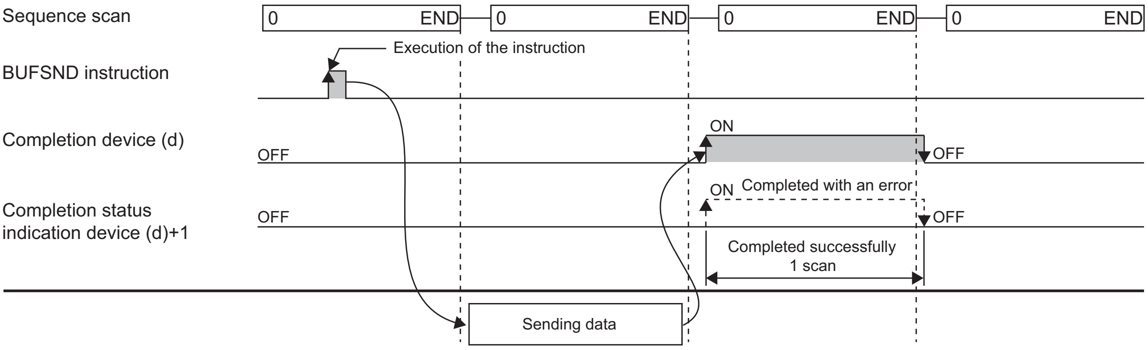
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the BUFSND instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the BUFSND instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the BUFSND instruction.



- The BUFSND instruction is executed on the rising edge (OFF to ON) of the send command.

Operation error

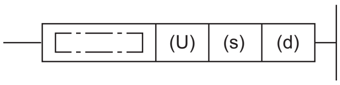
Error code ((s2)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

16.4 Reinitializing the Module

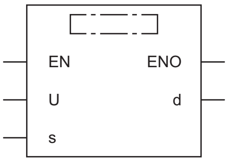
G(P).UINI, Z(P).UINI

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions reinitialize the module. The UINI instruction can be used only when the network type is "Q-compatible Ethernet".

Ladder	ST
	ENO:=G_UINI(EN,U,s,d); ENO:=GP_UINI(EN,U,s,d); ENO:=Z_UINI(EN,U,s,d); ENO:=ZP_UINI(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.UINI Z.UINI	
GP.UINI ZP.UINI	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

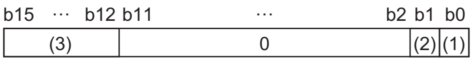
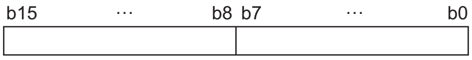
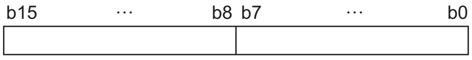
Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	G(P).UINI	—	—	○	—	—	—	—	○	○	—	—	○
	Z(P).UINI	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+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
+2	Change target specification	■To change the address information of the external device held by the Ethernet module Specify 0000H.*1 ■To change the own station IP address, operation setting, transmission speed, or communication mode Specify the parameter of the change target. Note that changing the transmission speed and communication mode cannot be specified together with the change specification of the own station IP address or operation setting. If they are specified together, the change specification of only the own station IP address or operation setting is executed.  (1) Own station IP address change specification (b0) Specify whether to change the IP address of the own station. (When changing the IP address, specify a new address in (s)+3, (s)+4.) • 0: Not changed • 1: Changed (2) Operation setting change specification (b1) Specify whether to change the Ethernet operation setting. (When changing the setting, specify a new setting in (s)+5.) • 0: Not changed • 1: Changed (3) Transmission speed and communication mode change specification (b12 to b15) Specify the transmission speed and communication mode. • 0: Not changed • 1: Auto negotiation*2 • 2: 100Mbps/full duplex • 3: 100Mbps/half-duplex • 4: 10Mbps/full duplex • 5: 10Mbps/half-duplex	0000H to 5000H	User
+3 +4	Own station IP address	Specify the IP address of the own station. • (s)+3  • b8 to b15: 3rd octet • b0 to b7: 4th octet • (s)+4  • b8 to b15: 1st octet • b0 to b7: 2nd octet	00000001H to FFFFFFFEH	User

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+5	Ethernet operation setting	Specify the operation setting. b15 … b9 b8 b7 b6 b5 … b2 b1 b0 0 (3) 0 (2) 0 (1) 0 (1) Communication data code setting (b1) • 0: Binary • 1: ASCII (2) Online change enable/disable setting (b6) • 0: Disable all at once (SLMP) • 1: Enable all at once (SLMP) (3) Opening method setting (b8) • 0: Open with the program. • 1: Do not open with the program.	Left	User

*1 The instruction clears and reinitializes the address information of the external device held by the Ethernet module so that data communication can be restarted. (The initialization normal completion signal (X19) turns on.)

*2 When auto negotiation is specified, use 1000BASE-T.

Processing details

- These instructions reinitializes the Ethernet module specified by (U). If changing the IP address is specified, change the IP address of the Ethernet module of the own station.
- The execution status and the completion status of the UINI instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

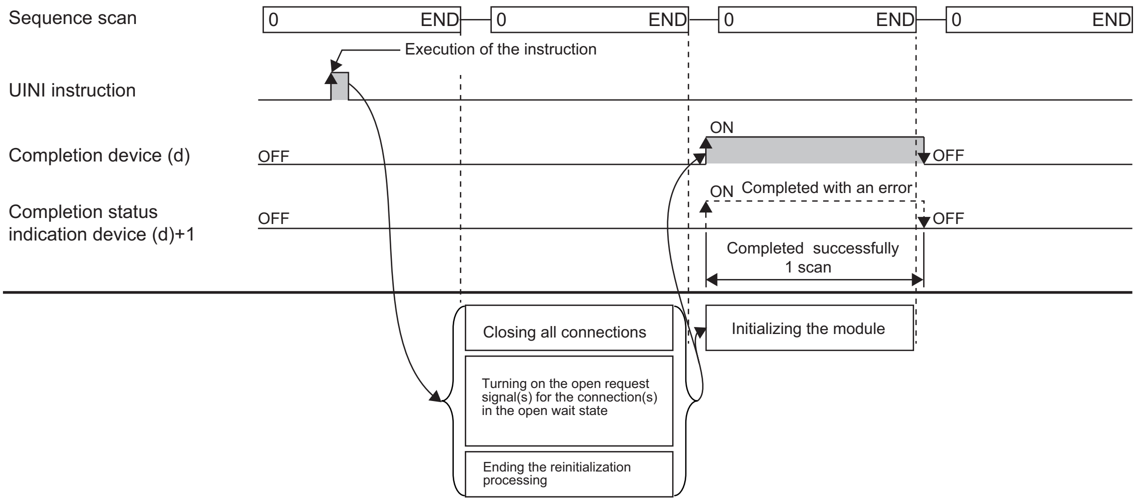
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the UINI instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the UINI instruction.



- The UINI instruction is executed on the rising edge (OFF to ON) of the reinitialize command.

Precautions

- When executed, the instruction forcibly closes all connections and initialize the internal information of the module.
- If the IP address is changed, communication is stopped. When an engineering tool is connected via Ethernet, the IP address of the Ethernet-equipped module should be corrected to the new IP address in the "Specify Connection Destination Connection" window before restarting communications.
- If the IP address is changed while "Use IP address" is selected in "Network No. and station number setting method" of the engineering tool, the network number and station number are not changed and the value using the IP address that is set by a parameter remains unchanged.

Operation error

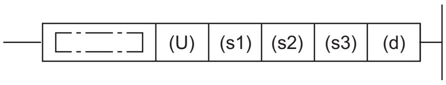
Error code ((s)+1)	Description
C000H to CFFFH	🕮 MELSEC iQ-R Ethernet User's Manual (Application)

16.5 Executing the Protocols Registered for the Predefined Protocol Support Function

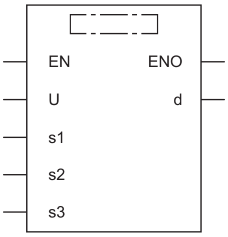
GP.ECPRTCL

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

This instruction executes the communication protocol registered using the engineering tool.

Ladder	ST
 (U) (s1) (s2) (s3) (d)	ENO:=GP_ECPRTCL(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.ECPRTCL	↑

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Connection No.	1 to 16	16-bit unsigned binary	ANY16
(s2)	Number of protocols to be executed continuously	1 to 8	16-bit unsigned binary	ANY16
(s3)	Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s3)	—	—	○*2	—	—	—	—	○	—	—	—	—
(d)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Resulting number of executed protocols	The number of protocols executed by the GP.ECPRTCL instruction is stored. Any protocol where an error occurred is also included in the execution number. If the setting of setting data or control data contains an error, "0" is stored.	0, 1 to 8	System
+1	Completion status	The completion status is stored upon completion of the instruction. If two or more protocols are executed, the execution result of the last protocol will be stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Execution protocol number 1	Specify the number of the protocol to be executed first.	1 to 128	User
+3	Execution protocol number 2	Specify the number of the protocol to be executed second.	0, 1 to 128	User
+4	Execution protocol number 3	Specify the number of the protocol to be executed third.	0, 1 to 128	User
+5	Execution protocol number 4	Specify the number of the protocol to be executed fourth.	0, 1 to 128	User
+6	Execution protocol number 5	Specify the number of the protocol to be executed fifth.	0, 1 to 128	User
+7	Execution protocol number 6	Specify the number of the protocol to be executed sixth.	0, 1 to 128	User
+8	Execution protocol number 7	Specify the number of the protocol to be executed seventh.	0, 1 to 128	User
+9	Execution protocol number 8	Specify the number of the protocol to be executed eighth.	0, 1 to 128	User
+10	Receive packet number 1 successful in collation match	If receiving is included in the communication type of the protocol that has been executed first, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the first protocol, 0 is stored.	0, 1 to 16	System
+11	Receive packet number 2 successful in collation match	If receiving is included in the communication type of the protocol that has been executed second, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the second protocol, "0" is stored. If the number of protocols executed is less than 2, 0 is stored.	0, 1 to 16	System
+12	Receive packet number 3 successful in collation match	If receiving is included in the communication type of the protocol that has been executed third, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the third protocol, 0 is stored. If the number of protocols executed is less than 3, 0 is stored.	0, 1 to 16	System

Operand: (s3)

Device	Item	Description	Setting range	Set by
+13	Receive packet number 4 successful in collation match	If receiving is included in the communication type of the protocol that has been executed fourth, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the fourth protocol, 0 is stored. If the number of protocols executed is less than 4, 0 is stored.	0, 1 to 16	System
+14	Receive packet number 5 successful in collation match	If receiving is included in the communication type of the protocol that has been executed fifth, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the fifth protocol, 0 is stored. If the number of protocols executed is less than 5, 0 is stored.	0, 1 to 16	System
+15	Receive packet number 6 successful in collation match	If receiving is included in the communication type of the protocol that has been executed sixth, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the sixth protocol, 0 is stored. If the number of protocols executed is less than 6, 0 is stored.	0, 1 to 16	System
+16	Receive packet number 7 successful in collation match	If receiving is included in the communication type of the protocol that has been executed seventh, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the seventh protocol, 0 is stored. If the number of protocols executed is less than 7, 0 is stored.	0, 1 to 16	System
+17	Receive packet number 8 successful in collation match	If receiving is included in the communication type of the protocol that has been executed eighth, the receive packet number successful in collation match is stored. If the communication type is "receive only", 0 is stored. If an error occurs during execution of the eighth protocol, 0 is stored. If the number of protocols executed is less than 8, 0 is stored.	0, 1 to 16	System

Processing details

- This instruction executes the protocol registered by the communication protocol support tool of the engineering tool for the Ethernet module specified by (U). Using the connection specified by (s1), the instruction executes the protocol in accordance with the control data stored in the device specified by (s3) and later.
- The instruction continuously executes as many protocols as specified by (s2) (a maximum of 8 protocols) at one time.
- The number of executed protocols is stored in the device specified by (s3)+0.
- The execution status and the completion status of the GP.ECPRTCL instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the GP.ECPRTCL instruction completes, and turns off during the next END processing.

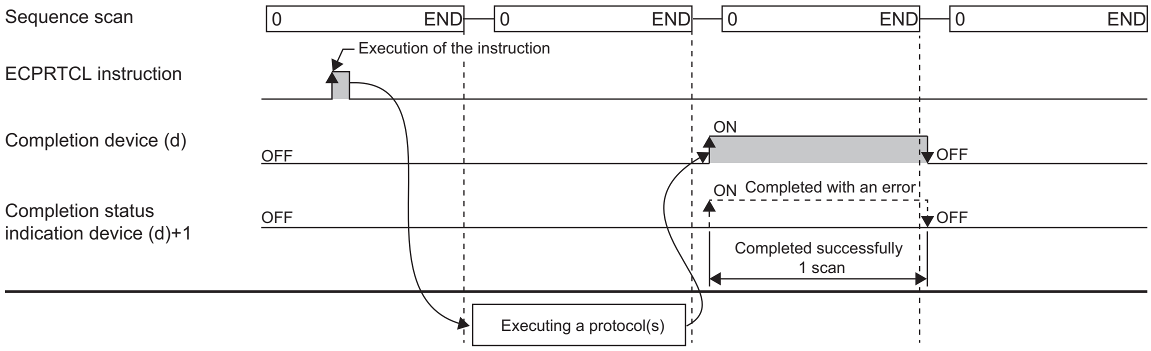
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the GP.ECPRTCL instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the GP.ECPRTCL instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the GP.ECPRTCL instruction.



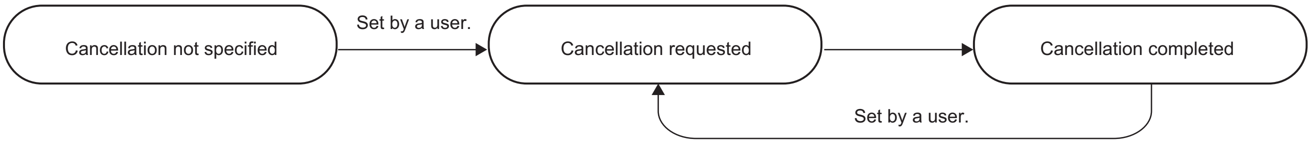
■Canceling protocol execution

A protocol can be canceled during its execution. The canceled protocol stops the processing and completes with an error. If multiple protocols are being executed sequentially, the current protocol is forcibly terminated, and the subsequent protocols are not executed.

Note that if the Ethernet-equipped module has started sending data, the send processing cannot be canceled even the protocol execution is canceled. The following table summarizes the cancellation timings and operation results.

Protocol operation	Cancellation timing (protocol execution status)	Operation result	
		Completion status	Handling of data
Sending data	'Protocol execution start' to 'Before transmission start (waiting for transmission)'	Completed with an error (protocol cancellation request error)	Data not being sent
	'Transmission start' to 'Before protocol execution completion'		Data being sent
Receiving data	Any time		Data not being received

To request a protocol cancellation, use the module label, "Protocol cancellation specification". Set 1 (Cancellation request) to the module label, "Protocol cancellation specification", of the cancellation-target connection. The label states transition as follows:



Protocol cancellation specification	Description
No cancellation specified	A state that a cancellation has never been requested since the the system was powered on or the CPU module was reset. Once a protocol cancellation is requested, the label never returns to this state.
Cancellation requested	A state that a protocol cancellation has been requested by a user. The label is in this state until the processing completes.
Cancellation completed	A state that the cancellation processing has been completed

Point

- If a cancellation is requested while no protocol is being executed, the module performs no processing.
- The module periodically checks for a cancellation request. It may take time until the protocol is actually canceled after the cancellation was requested.
- While the protocol cancellation specification state is "Cancellation requested", another GP.ECPRTCL instruction cannot be executed.

Precautions

- If an error occurs in the mth protocol while multiple protocols are being executed, the instruction does not execute the "m+1"th protocol and after and is completed with an error.
- The connections for which the GP.ECPRTCL instruction can be executed are only those for which "Communication protocol" is specified for the communication means.
- If a cancel request is received during execution of the mth protocol while multiple protocols are executed continuously, the Ethernet module stores the protocol number m being executed in the device specified by (s3)+0, the receive packet number successful in collation match for the already executed protocol in the device specified by 1 to (m-1), and the protocol cancel request error (error code: C404H) in the device specified by (s3)+1.
- When a protocol including no-conversion variables is executed, the total data length of the variables used in one packet may exceed 1920 bytes. In this case, the instruction may obtain CPU device values over several scans. Therefore, do not change the CPU device values specified in non-conversion variables from the start of the dedicated instruction to the end of execution.
- The intelligent function module device (buffer memory) is not affected by the sequence scan of the CPU module and therefore can process protocols faster than assigning programmable controller devices to variables.
- If protocol setting data is written during execution of the GP.ECPRTCL instruction, the protocol in execution is canceled upon completion of writing and the instruction is completed with a protocol setting data write error (error code: C430H).
- If the fixed buffer send/receive mode (such as send) of the specified connection has an invalid combination with the communication type of the protocol to be executed (such as receive only) while the connection number of the Ethernet module is one from 1 to 16, the GP.ECPRTCL instruction is completed with a connection number setting error (error code: C407H).
- If the receive waiting time is set to "0: Infinite wait", the GP.ECPRTCL instruction is not completed until the data specified in the protocol setting is received.

Operation error

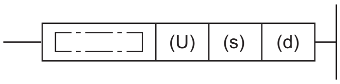
Error code ((s3)+1)	**Description**
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

16.6 Clearing Error Information

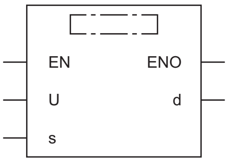
GP.ERRCLEAR, ZP.ERRCLEAR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions turn off the LEDs and clear error information.[1]

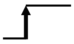
Ladder	ST
	ENO:=GP_ERRCLEAR(EN,U,s,d); ENO:=ZP_ERRCLEAR(EN,U,s,d);

FBD/LD



[1] To turn off the LEDs, use the RJ71EN71 with a firmware version of "05" or later.

■Execution condition

Instruction	Execution condition
GP.ERRCLEAR ZP.ERRCLEAR	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.ERRCLEAR	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.ERRCLEAR	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.ERRCLEAR	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.ERRCLEAR	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+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
+2	Clear target specification	Specify the error information to clear.	0000H to 0080H, 0100H, 0101H, 8101H, FFFFH	User
+3	Clear function specification	Specify the function to clear. • 0000H: Clear the error code. • FFFFH: Clear the error log.	0000H, FFFFH	User
+4	LED off specification	Specify the off-target LED (ERR LED, P ERR LED of PORT1, or P ERR LED of PORT2). • 0000H: Do not turn off any ERR LED. • 0001H: Turn off the P ERR LED of PORT1. • 8001H: Turn off the P ERR LED of PORT2. • FFFFH: Turn off all ERR LEDs. If 0001H or 8001H is specified and the P ERR LEDs of PORT1 and PORT2 are both off after the ERRCLEAR instruction is executed, the ERR LED automatically turns off. If either one of the P ERR LED is on, the ERR LED also remains on.	0000H, 0001H, 8001H, FFFFH	User
+5 to +7	System area	—	—	—

Processing details

- These instructions clear following error information with regard to the module specified by (U).

Target name		Clear target specification (s)+2	Clear function specification (s)+3	Specify the error information to clear.	Buffer memory address	
					Network type: Ethernet	Network type: Q-compatible Ethernet
Initial error		0000H	0000H	Initial error code	1900025	105
Open error		0001H to 0080H[*1]	0000H	Open error code of connection The open error code of connection No. 1 to 128 (0001H to 0080H) is cleared.	100 to 163, 2000100 to 2000163	124, 134, 144, 154, 164, 174, 184, 194, 22564, 22574, 22584, 22594, 22604, 22614, 22624, 22634
Error log[*2]		0100H	FFFFH	Error log	—	227 to 372
Communication status (PORT1)	Status by protocol	0101H	FFFFH	PORT1 communication status by protocol	5000 to 5188	376 to 511
Communication status (PORT2)[*3]	Status by protocol	8101H	FFFFH	PORT2 communication status by protocol	2005000 to 2005188	—
All error information		FFFFH	FFFFH	All the information above is cleared.	—	—

*1 Only 0001H to 0010H can be specified when the network type is Q-compatible Ethernet.

*2 The error log can be cleared only when the network type is Q-compatible Ethernet.

*3 The initial error can be cleared only when the network type is Ethernet.

Operation error

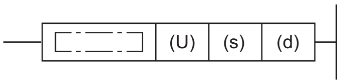
Error code ((s)+1)	Description
C000H to CFFFH	📖 MELSEC iQ-R Ethernet User's Manual (Application)

16.7 Reading Error Information

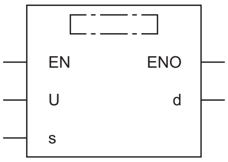
GP.ERRRD, ZP.ERRRD

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

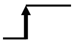
These instructions read error information.

Ladder	ST
	ENO:=GP_ERRRD(EN,U,s,d); ENO:=ZP_ERRRD(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.ERRRD ZP.ERRRD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.ERRRD	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	ZP.ERRRD	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.ERRRD	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.ERRRD	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+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
+2	Read information specification	Specify the error information to read.	0000H to 0080H	User
+3	Read target information specification	Specify the target of the error information to read. • 0000H: Latest error information	0000H	User
+4	Error information	The error information that has been read is stored. • 0000H: No error • Other than 0000H: Error code	—	System
+5 to +7	System area	—	—	—

Processing details

- These instructions read the following error information with regard to the module specified by (U).

Target name	Read information specification (s)+2	Read target information specification (s)+3	Error information to read	Buffer memory address	
				Network type: Ethernet	Network type: Q-compatible Ethernet
Initial error	0000H	0000H	Initial error code	1900025	105
Open error	0001H to 0080H*1	0000H	Open error code of connection The open error code of connection No. 1 to 128 (0001H to 0080H) is read.	100 to 163, 2000100 to 2000163	124, 134, 144, 154, 164, 174, 184, 194, 22564, 22574, 22584, 22594, 22604, 22614, 22624, 22634

*1 Only 0001H to 0010H can be specified when the network type is Q-compatible Ethernet.

Operation error

Error code ((s)+1)	Description
C000H to CFFFH	🕮 MELSEC iQ-R Ethernet User's Manual (Application)

17 CC-LINK IE CONTROLLER NETWORK INSTRUCTIONS

Point

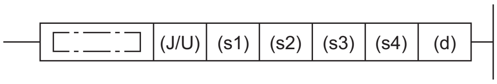
This chapter describes the instructions used commonly by MELSEC iQ-R series modules. For the instructions when MELSEC-Q series modules are used, refer to the manuals for each module used and create programs.
For precautions when using modules, refer to the following.
📖 MELSEC iQ-R Module Configuration Manual

17.1 Remote RUN

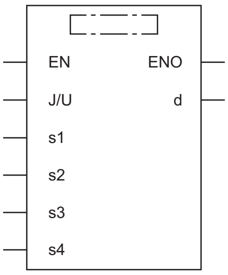
J(P).RRUN, G(P).RRUN, Z(P).RRUN

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

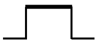
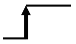
These instructions execute remote RUN for the programmable controller on another station.

Ladder	ST
	ENO:=J_RRUN(EN,J,s1,s2,s3,s4,d); ENO:=JP_RRUN(EN,J,s1,s2,s3,s4,d); ENO:=G_RRUN(EN,U,s1,s2,s3,s4,d); ENO:=GP_RRUN(EN,U,s1,s2,s3,s4,d); ENO:=Z_RRUN(EN,J/U,s1,s2,s3,s4,d); ENO:=ZP_RRUN(EN,J/U,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
J.RRUN G.RRUN Z.RRUN	
JP.RRUN GP.RRUN ZP.RRUN	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RRUN	(J): Target station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RRUN	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).RRUN	(J): Target station network number	1 to 239	String	ANY16_OR_STRING_SINGLE
	Z(P).RRUN	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Own station channel (☞ Page 1450 Range of available channel numbers) Specifies the channel to be used by the own station or node. Specify the channel that was specified, as being used by the own station, in the RSTOP instruction.	1 to 8, 11 to 18	16-bit unsigned binary	ANY16
(s2)		Specify the station number of the target station/target node. (1) Station number specification 1 to 120 (2) Group specification[*1] 81H to A0H: All stations of group numbers 1 to 32 (3) All-station specification[*1] FFH: All stations of target network numbers, except own station	1 to 120 0081H to 00A0H 00FFH	16-bit unsigned binary	ANY16
(s3)		Specify that CPU module in the target station that is to be accessed. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU[*2] • 03D1H: Addressed to standby system CPU[*2] • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	16-bit unsigned binary	ANY16
(s4)		Specify the operation and clear modes. b15 … b8 b7 … b4 b3 … b0 0 (2) (1) (1) Operation mode Specifies whether or not remote RUN be forcibly executed. • 1H: Not forcibly executed • 3H: Forcibly executed The forcible execution function forcibly executes remote RUN from another station when the station that executed remote STOP can no longer execute remote RUN. (2) Clear mode Specifies the CPU unit device status resulting after remote RUN is executed. • 0H: Not cleared, except local devices • 1H: Cleared, except the latch range • 2H: Cleared, including the latch range The clear mode setting specifies the clear, or initialization, processing for the CPU module device at the operation start of CPU module calculation by remote RUN. After completion of the specified clear processing, the CPU module executes RUN according to the initial device values of the engineering tool.	0001H 0003H 0011H 0013H 0021H 0023H	16-bit unsigned binary	ANY16
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[1] If the instruction is executed with group or all stations specification:
For the target station CPU type, specify either 0000H or 03FFH. If the target station is in CC-Link IE Field Network, group specification is not available. To check that the data has been written successfully in the target station, examine whether the CPU module of the destination has entered the RUN state.

[2] If the instruction is executed with the control system CPU (03D0H) or standby system CPU (03D1H) specified, it may complete with an error if system switching occurs in the target station (error code: 4244H, 4248H).
Retry to execute the instruction if it completed with an error due to the above error.

Point

- Remote RUN is enabled when the RUN/STOP/RESET switch of the target station CPU module is "RUN."
- If the target station CPU module is system protected, remote RUN will fail.
- If the target station CPU module has already been remotely stopped or paused from another station, RUN will fail if (s4) is "Not forcibly executed (0001H)."

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).RRUN	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RRUN	—	—	○	—	—	—	—	○	○	—	—	—	○
	Z(P).RRUN	—	—	○	—	—	—	—	○	—	—	○	○	○
(s1)		○[1]	—	○[2]	—	—	—	—	○	○	—	—	—	—
(s2)		○[1]	—	○[2]	—	—	—	—	○	○	—	—	—	—
(s3)		○[1]	—	○[2]	—	—	—	—	○	○	—	—	—	—
(s4)		○[1]	—	○[2]	—	—	—	—	○	○	—	—	—	—
(d)		○[1]	—	○[3]	—	—	—	—	—	—	—	—	—	—

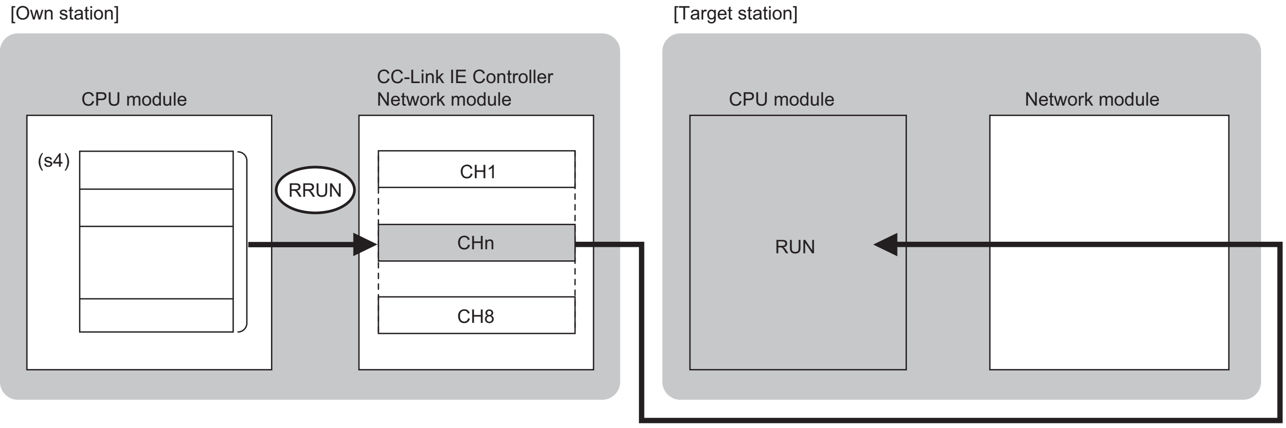
[1] FX and FY cannot be used.

[2] FD cannot be used.

[3] T, ST, C, and FD cannot be used.

Processing details

- Remote RUN is executed for the target station CPU module according to the (s4) specification details. The target station is specified with the target network number (J) or the start I/O number (U) of the own station or node and the target station number (s2). After completion of request to the target station, the completion device specified by (d) turns on.



CH: Channel

- The specifiable stations are those with the specified network number that are included in one of the following networks.
  - MELSECNET/H
  - MELSECNET/10
  - CC-Link IE Controller Network
  - CC-Link IE Field Network
  - Ethernet

- The execution status and the completion status of the RRUN instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the RRUN instruction completes, and turns off during the next END processing.

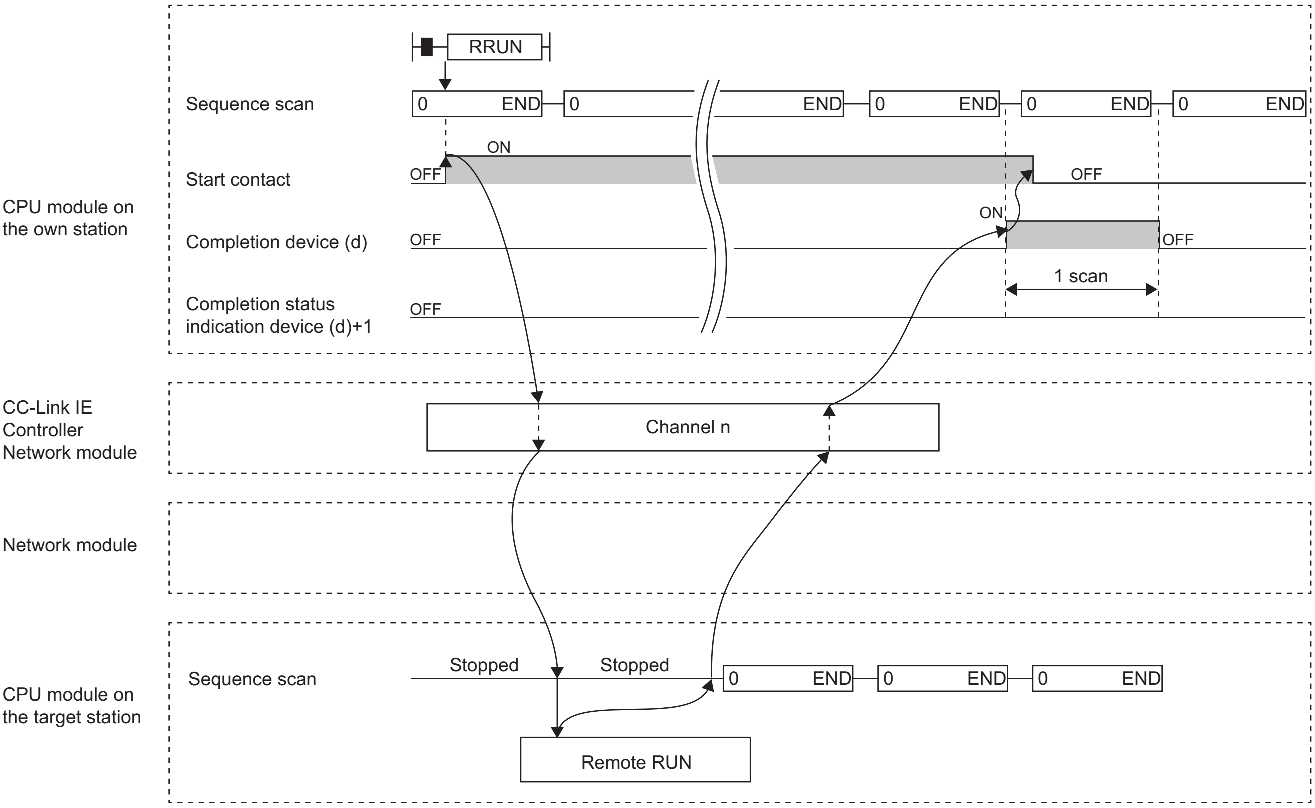
  - Completion status indication device (d)+1

This device turns on or off depending on the completion status of the RRUN instruction.

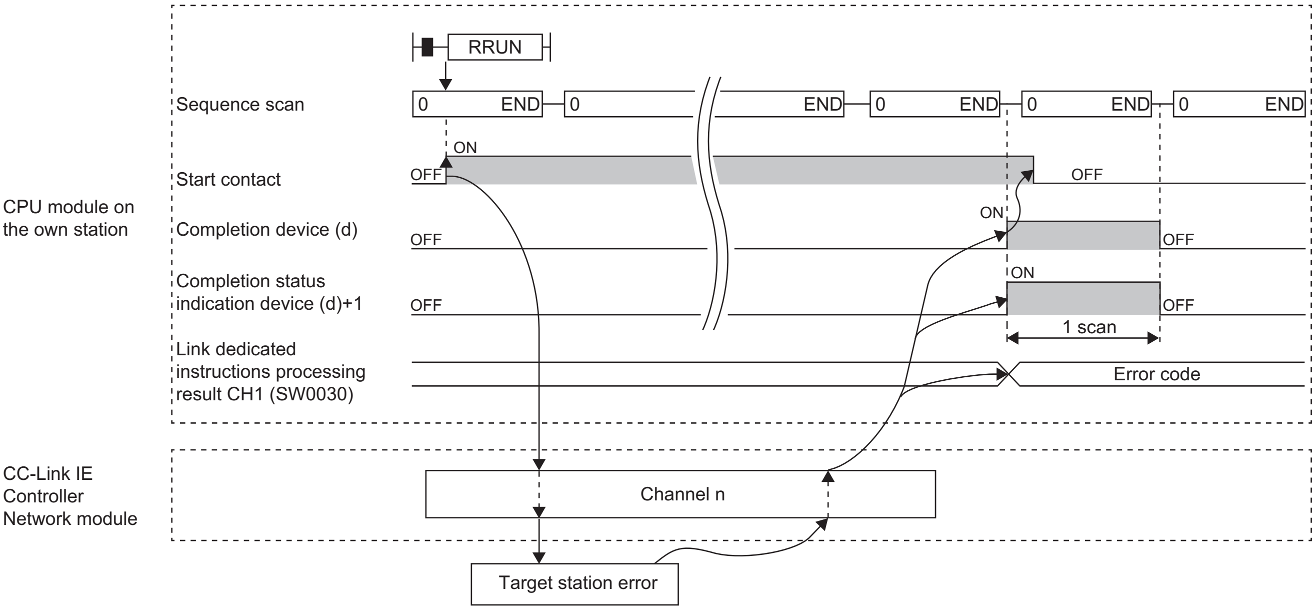
When completed normally: Unchanged from off.

When completed with an error: The device turns on during END processing of the scan where the RRUN instruction completes, and turns off during the next END processing.

- The following figure shows the RRUN instruction execution timing.
  - When completed successfully



  - When completed with an error



- If the J.RRUN, G.RRUN, or Z.RRUN is used to execute processing, processing of one time is successively followed by the next processing while the start-up contact is on.
- If the JP.RRUN, GP.RRUN, or ZP.RRUN is used to execute processing, processing is executed once at the start-up contact off-to-on rise.

Operation error

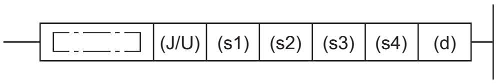
Error code (SW0030 to SW0037)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

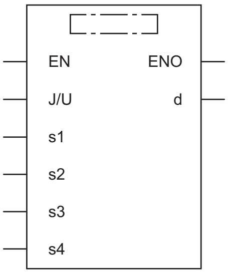
17.2 Remote STOP

J(P).RSTOP, G(P).RSTOP, Z(P).RSTOP

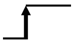
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions execute remote STOP for the programmable controller on another station.

Ladder	ST
	ENO:=J_RSTOP(EN,J,s1,s2,s3,s4,d); ENO:=JP_RSTOP(EN,J,s1,s2,s3,s4,d); ENO:=G_RSTOP(EN,U,s1,s2,s3,s4,d); ENO:=GP_RSTOP(EN,U,s1,s2,s3,s4,d); ENO:=Z_RSTOP(EN,J/U,s1,s2,s3,s4,d); ENO:=ZP_RSTOP(EN,J/U,s1,s2,s3,s4,d);

FBD/LD


■Execution condition

Instruction	Execution condition
J.RSTOP G.RSTOP Z.RSTOP	
JP.RSTOP GP.RSTOP ZP.RSTOP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RSTOP	(J): Target station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RSTOP	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).RSTOP	(J): Target station network number	1 to 239	String	ANY16_OR_STRING_SINGLE
	Z(P).RSTOP	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Own station channel (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	16-bit unsigned binary	ANY16
(s2)		Specify the station number of the target station/target node. (1) Station number specification 1 to 120 (2) Group specification[*1] 81H to A0H: All stations of group numbers 1 to 32 (3) All-station specification[*1] FFH: All stations of target network numbers, except own station	1 to 120 81H to A0H FFH	16-bit unsigned binary	ANY16
(s3)		Specify that CPU module in the target station that is to be accessed. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU[*2] • 03D1H: Addressed to standby system CPU[*2] • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	16-bit unsigned binary	ANY16
(s4)		Specifies the operation mode. (0001H (fixed))	0001H (fixed)	16-bit unsigned binary	ANY16
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

[*1] If the instruction is executed with group or all stations specification:
For the target station CPU type, specify either 0000H or 03FFH. If the target station is in CC-Link IE Field Network, group specification is not available. To check that the data has been written successfully in the target station, examine whether the CPU module of the destination has entered the STOP state.

[*2] If the instruction is executed with the control system CPU (03D0H) or standby system CPU (03D1H) specified, it may complete with an error if system switching occurs in the target station (error code: 4244H, 4248H).
Retry to execute the instruction if it completed with an error due to the above error.

Point

- Remote STOP is enabled when the RUN/STOP/RESET switch of the target station CPU module is "RUN."
- If the target station CPU module is system protected, remote STOP will fail.
- Resetting the target station CPU on which remote STOP was executed deletes information about the remote STOP.

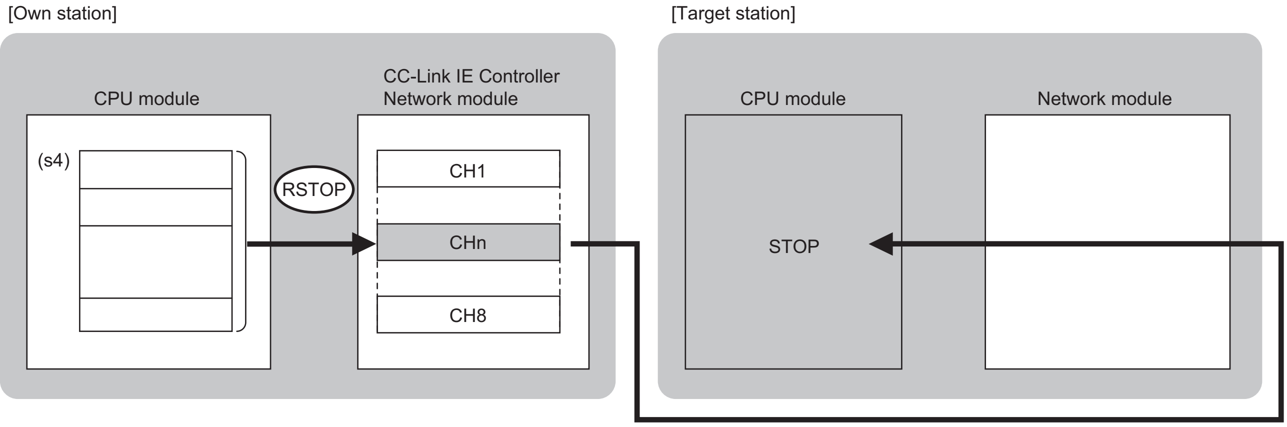
■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).RSTOP	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RSTOP	—	—	○	—	—	—	—	○	○	—	—	—	○
	Z(P).RSTOP	—	—	○	—	—	—	—	○	—	—	○	○	○
(s1)		○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—	—
(s2)		○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—	—
(s3)		○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—	—
(s4)		○[*1]	—	○[*2]	—	—	—	—	○	○	—	—	—	—
(d)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 FD cannot be used.
*3 T, ST, C, and FD cannot be used.

Processing details

- Remote STOP is executed for the target station CPU module. The target station is specified with the target network number (J) or the start I/O number (U) of the own station or node and the target station number (s2). After completion of request to the target station, the completion device specified by (d) turns on.



CH: Channel

- The specifiable stations are those with the specified network number that are included in one of the following networks.
  - MELSECNET/H
  - MELSECNET/10
  - CC-Link IE Controller Network
  - CC-Link IE Field Network
  - Ethernet
- The execution status and the completion status of the RSTOP instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the RSTOP instruction completes, and turns off during the next END processing.

- Completion status indication device (d)+1

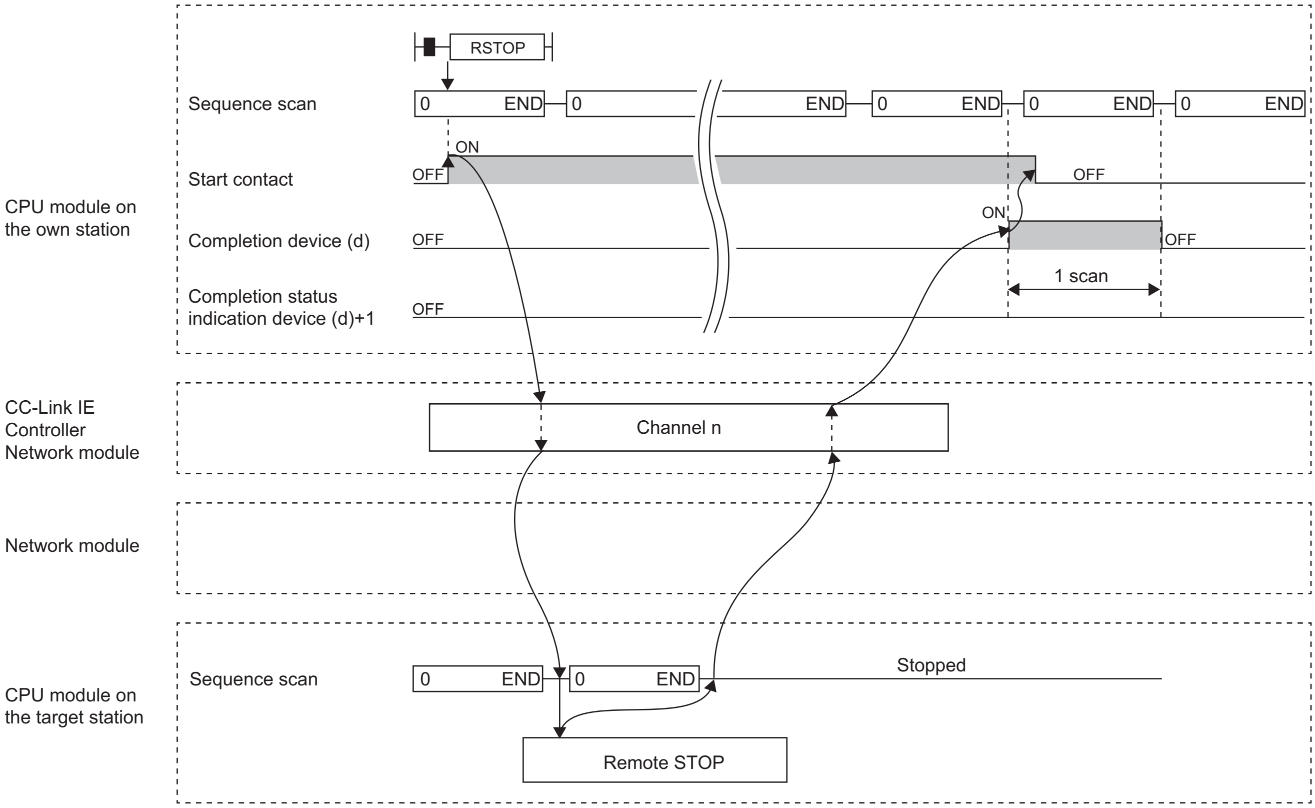
This device turns on or off depending on the completion status of the RSTOP instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the RSTOP instruction completes, and turns off during the next END processing.

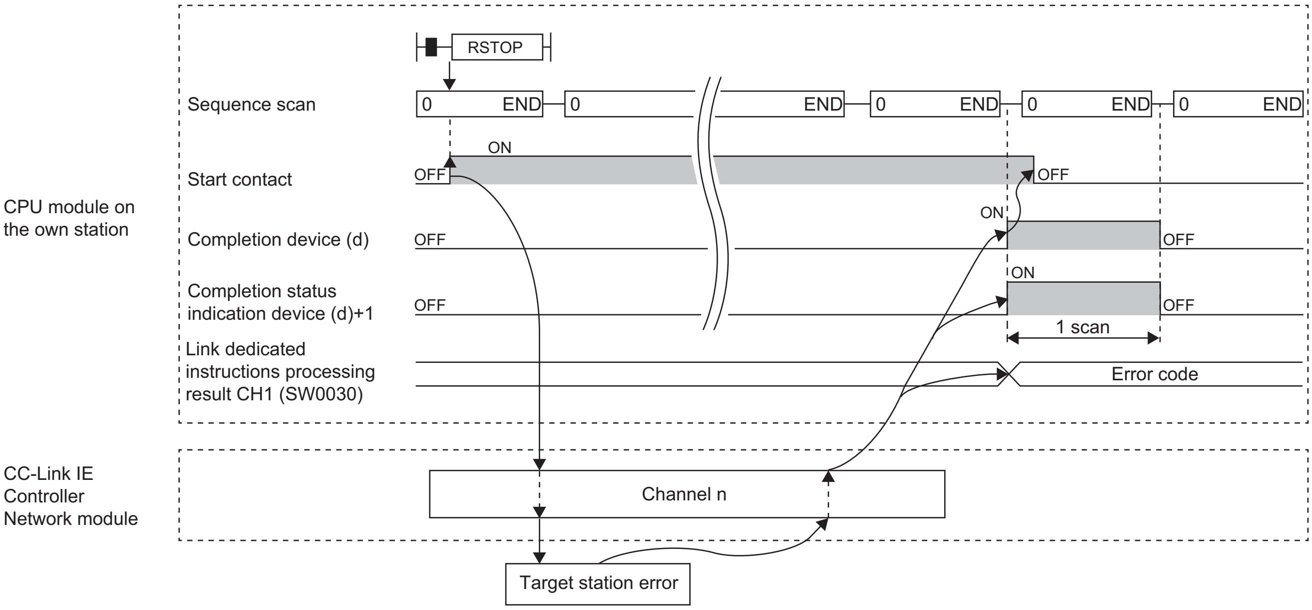
- The following figure shows the RSTOP instruction execution timing.

- When completed successfully



Depending on the system configuration and sequence scan time, several scans may be required until the sequence scan STOP completes.

- When completed with an error



- If the J.RSTOP, G.RSTOP, or Z.RSTOP is used to execute processing, processing of one time is successively followed by the next processing while the start-up contact is on.
- If the JP.RSTOP, GP.RSTOP, or ZP.RSTOP is used to execute processing, processing is executed once at the start-up contact off-to-on rise.

Operation error

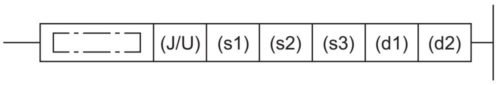
Error code (SW0030 to SW0037)	Description
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
E000H to EFFFH	🕮 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

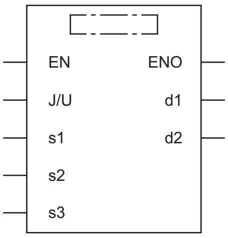
17.3 Reading Clock Data from the Programmable Controller on Another Station

J(P).RTMRD, G(P).RTMRD, Z(P).RTMRD

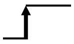
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instruction read clock data from the programmable controller on another station.

Ladder	ST
	ENO:=J_RTMRD(EN,J,s1,s2,s3,s4,d); ENO:=JP_RTMRD(EN,J,s1,s2,s3,s4,d); ENO:=G_RTMRD(EN,U,s1,s2,s3,s4,d); ENO:=GP_RTMRD(EN,U,s1,s2,s3,s4,d); ENO:=Z_RTMRD(EN,J/U,s1,s2,s3,s4,d); ENO:=ZP_RTMRD(EN,J/U,s1,s2,s3,s4,d);

FBD/LD


■Execution condition

Instruction	Execution condition
J.RTMRD G.RTMRD Z.RTMRD	
JP.RTMRD GP.RTMRD ZP.RTMRD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RTMRD	(J): Target station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RTMRD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).RTMRD	(J): Target station network number	1 to 239	String	ANY16_OR_STRING_SINGLE
	Z(P).RTMRD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s1)		Own station channel (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	16-bit unsigned binary	ANY16
(s2)		Target station number	1 to 120	16-bit unsigned binary	ANY16
(s3)		Specify that CPU module in the target station that is to be accessed. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU*1 • 03D1H: Addressed to standby system CPU*1 • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	16-bit unsigned binary	ANY16
(d1)		Own station start device for storing the read clock data.	—	Device name	ANY16_ARRAY (Number of elements: 4)
(d2)		Device of the own station, which 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 If the instruction is executed with the control system CPU (03D0H) or standby system CPU (03D1H) specified, it may complete with an error if system switching occurs in the target station (error code: 4244H, 4248H).
Retry to execute the instruction if it completed with an error due to the above error.

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).RTMRD	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RTMRD	—	—	○	—	—	—	—	○	○	—	—	—	○
	Z(P).RTMRD	—	—	○	—	—	—	—	○	—	—	○	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(s2)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(s3)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	○	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

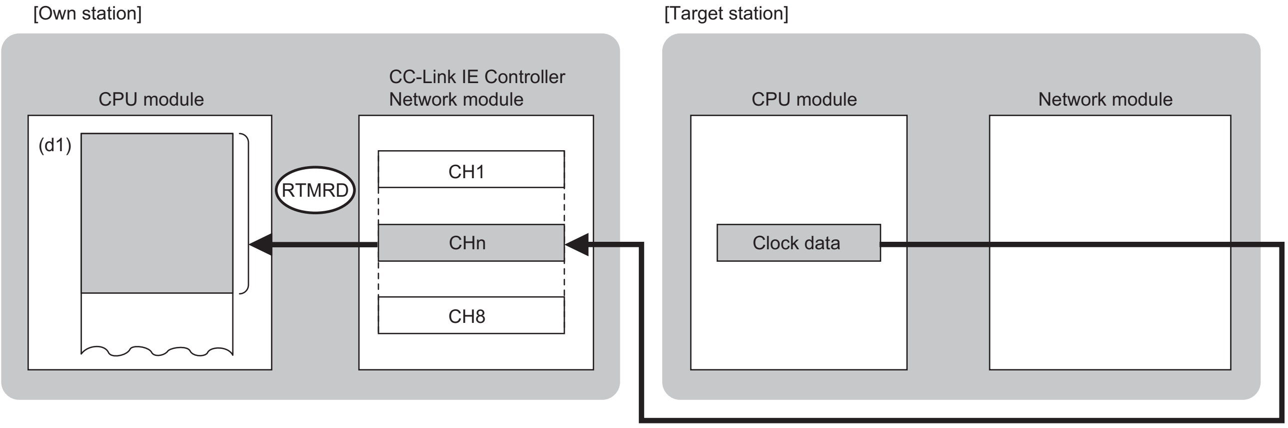
*1 FX and FY cannot be used.
*2 FD cannot be used.
*3 T, ST, C, and FD cannot be used.

■Clock data

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Clock data	Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Month (01H to 12H) The range of the 4 digits of the year is 1980 to 2079.	—	System
+1		Upper 8 bits: Day (01H to 31H) Lower 8 bits: Hour (00H to 23H)	—	System
+2		Upper 8 bits: Minute (00H to 59H) Lower 8 bits: Second (00H to 59H)	—	System
+3		Upper 8 bits: Year (19H and 20H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.)) The range of the 4 digits of the year is 1980 to 2079.	—	System

Processing details

- Clock data is read from the target station CPU module. The target station is specified with the target network number (J) or the start I/O number (U) of the own station or node and the target station number (s2). After completion of request to the target station, the completion device specified by (d2) turns on.



CH: Channel

- The specifiable stations are those with the specified network number that are included in one of the following networks.
  - MELSECNET/H
  - MELSECNET/10
  - CC-Link IE Controller Network
  - CC-Link IE Field Network
  - Ethernet
- The execution status and the completion status of the RTMRD instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the RTMRD instruction completes, and turns off during the next END processing.

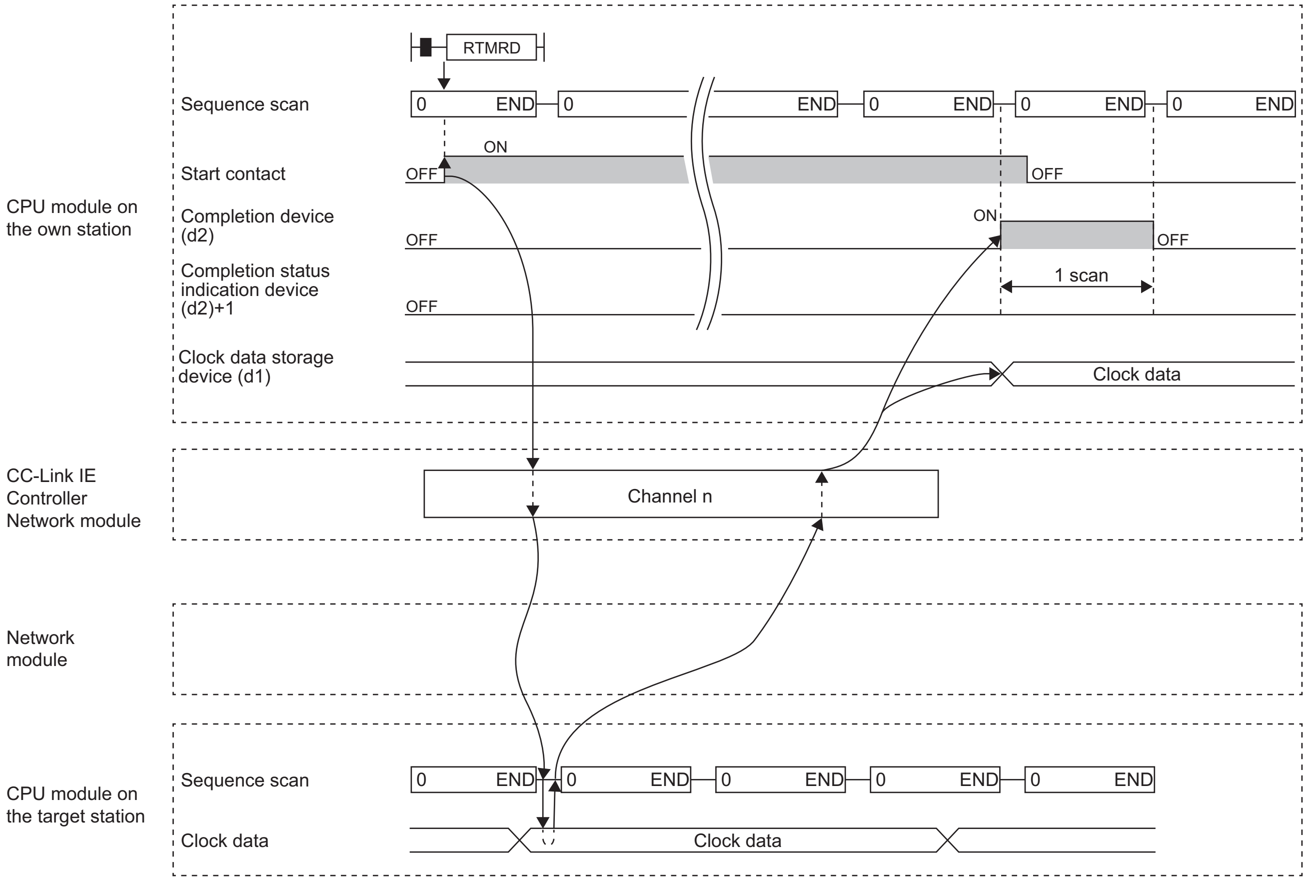
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the RTMRD instruction.

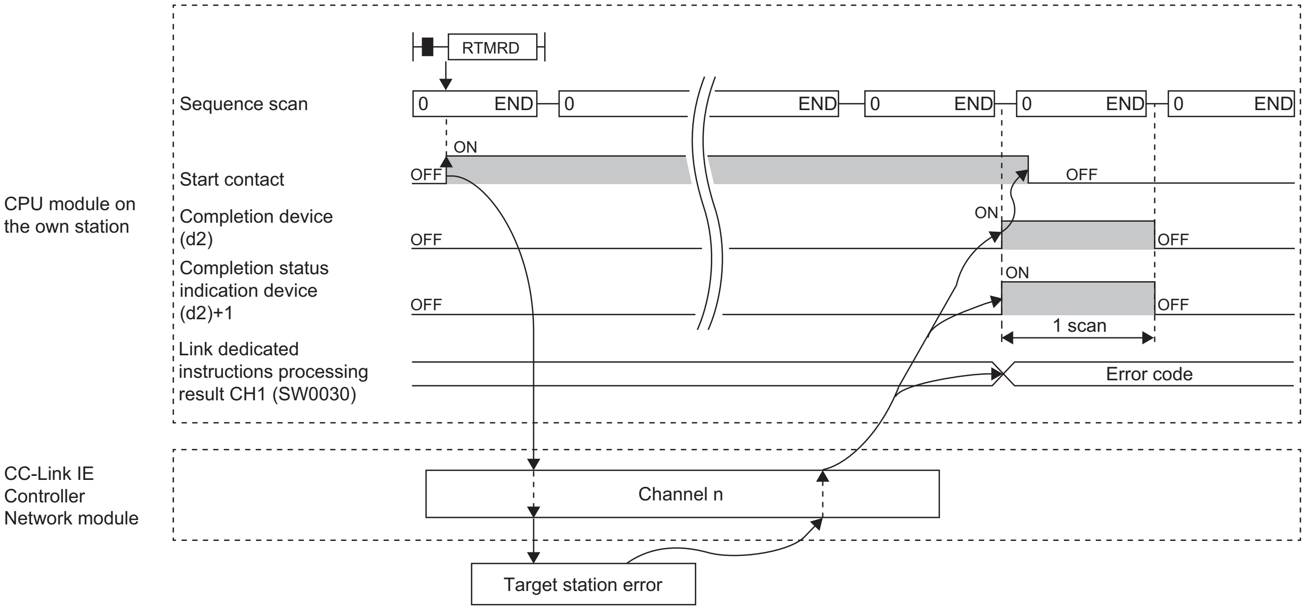
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the RTMRD instruction completes, and turns off during the next END processing.

- The following figure shows the RTMRD instruction execution timing.
  - When completed successfully



  - When completed with an error



- If the J.RTMRD, G.RTMRD, or Z.RTMRD is used to execute processing, processing of one time is successively followed by the next processing while the start-up contact is on.
- If the JP.RTMRD, GP.RTMRD, or ZP.RTMRD is used to execute processing, processing is executed once at the start-up contact off-to-on rise.

Operation error

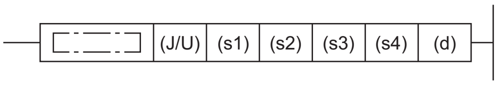
Error code (SW0030 to SW0037)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

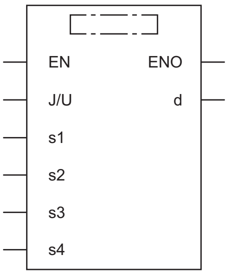
17.4 Writing Clock Data to the Programmable Controller on Another Station

J(P).RTMWR, G(P).RTMWR, Z(P).RTMWR

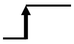
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instruction write clock data to the programmable controller on another station.

Ladder	ST
	ENO:=J_RTMWR(EN,J,s1,s2,s3,s4,d); ENO:=JP_RTMWR(EN,J,s1,s2,s3,s4,d); ENO:=G_RTMWR(EN,U,s1,s2,s3,s4,d); ENO:=GP_RTMWR(EN,U,s1,s2,s3,s4,d); ENO:=Z_RTMWR(EN,J/U,s1,s2,s3,s4,d); ENO:=ZP_RTMWR(EN,J/U,s1,s2,s3,s4,d);

FBD/LD


■Execution condition

Instruction	Execution condition
J.RTMWR G.RTMWR Z.RTMWR	
JP.RTMWR GP.RTMWR ZP.RTMWR	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).RTMWR	(J): Target station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).RTMWR	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).RTMWR	(J): Target station network number	1 to 239	String	ANY16_OR_STRING_ SINGLE
	Z(P).RTMWR	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_ SINGLE
(s1)		Own station channel (☞ Page 1450 Range of available channel numbers)	1 to 8, 11 to 18	16-bit unsigned binary	ANY16
(s2)		Specify the station number of the target station/target node. (1) Station number specification 1 to 120 (2) Group specification[*1] 81H to A0H: All stations of group numbers 1 to 32 (3) All-station specification[*1] FFH: All stations of target network numbers, except own station	1 to 120 81H to A0H FFH	16-bit unsigned binary	ANY16
(s3)		Specify that CPU module in the target station that is to be accessed. • 0000H: Addressed to target station CPU (control CPU) • 03D0H: Addressed to control system CPU[*2] • 03D1H: Addressed to standby system CPU[*2] • 03D2H: Addressed to system A CPU • 03D3H: Addressed to system B CPU • 03E0H: Addressed to multiple CPU No. 1 • 03E1H: Addressed to multiple CPU No. 2 • 03E2H: Addressed to multiple CPU No. 3 • 03E3H: Addressed to multiple CPU No. 4 • 03FFH: Addressed to target station CPU (control CPU)	0000H 03D0H to 03D3H 03E0H to 03E3H 03FFH	16-bit unsigned binary	ANY16
(s4)		Own station start device containing the clock data to be written	—	Device name	ANY16_ARRAY (Number of elements: 5)
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 If the instruction is executed with group or all stations specification:
For the target station CPU type, specify either 0000H or 03FFH. If the target station is in CC-Link IE Field Network, group specification is not available. To check that the data has been written successfully in the target station, examine whether the CPU module of the destination has entered the RUN state.

*2 If the instruction is executed with the control system CPU (03D0H) or standby system CPU (03D1H) specified, it may complete with an error if system switching occurs in the target station (error code: 4244H, 4248H).
Retry to execute the instruction if it completed with an error due to the above error.

■Applicable devices

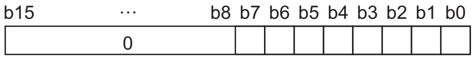
Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	J	U
(J/U)	J(P).RTMWR	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).RTMWR	—	—	○	—	—	—	—	○	○	—	—	—	○
	Z(P).RTMWR	—	—	○	—	—	—	—	○	—	—	○	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(s2)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(s3)		○*1	—	○*2	—	—	—	—	○	○	—	—	—	—
(s4)		—	—	○*2	—	—	—	—	○	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Clock data

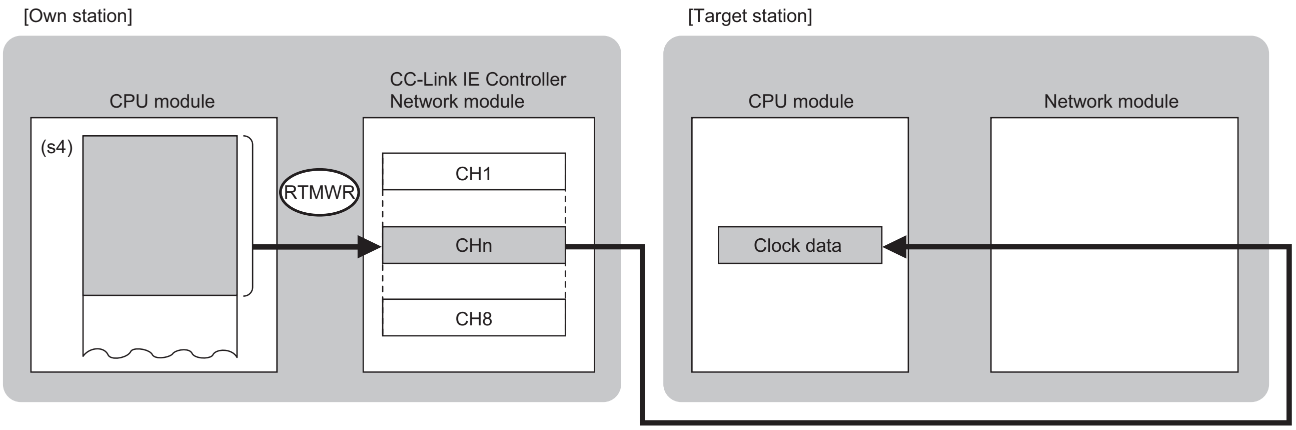
Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Change pattern	Specifies the bit pattern that indicates the item of the clock data to be changed. • 0: Not changed • 1: Changed  b7: Year (first two digits), b6: Day of week, b5: Second, b4: Minute, b3: Hour, b2: Day, b1: Month, b0: Year (last two digits)	0, 1	User
+1	Clock data	Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Month (01H to 12H) The range of the 4 digits of the year is 1980 to 2079.	Year: 00H to 99H Month: 01H to 12H	User
+2		Upper 8 bits: Day (01H to 31H) Lower 8 bits: Hour (00H to 23H)	Day: 01H to 31H Hour: 00H to 23H	User
+3		Upper 8 bits: Minute (00H to 59H) Lower 8 bits: Second (00H to 59H)	Minute: 00H to 59H Second: 00H to 59H	User
+4		Upper 8 bits: Year (19H and 20H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.)) The range of the 4 digits of the year is 1980 to 2079.	Year: 19H to 20H Day of week: 00H to 06H	User

Point

If the target station CPU module is system protected, writing the clock data will fail.

Processing details

- Clock data is written from the target station CPU module. The target station is specified with the target network number (J) or the start I/O number (U) of the own station or node and the target station number (s2). After completion of request to the target station, the completion device specified by (d) turns on.



CH: Channel

- The specifiable stations are those with the specified network number that are included in one of the following networks.
  - MELSECNET/H
  - MELSECNET/10
  - CC-Link IE Controller Network
  - CC-Link IE Field Network
  - Ethernet
- The execution status and the completion status of the RTMWR instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the RTMWR instruction completes, and turns off during the next END processing.

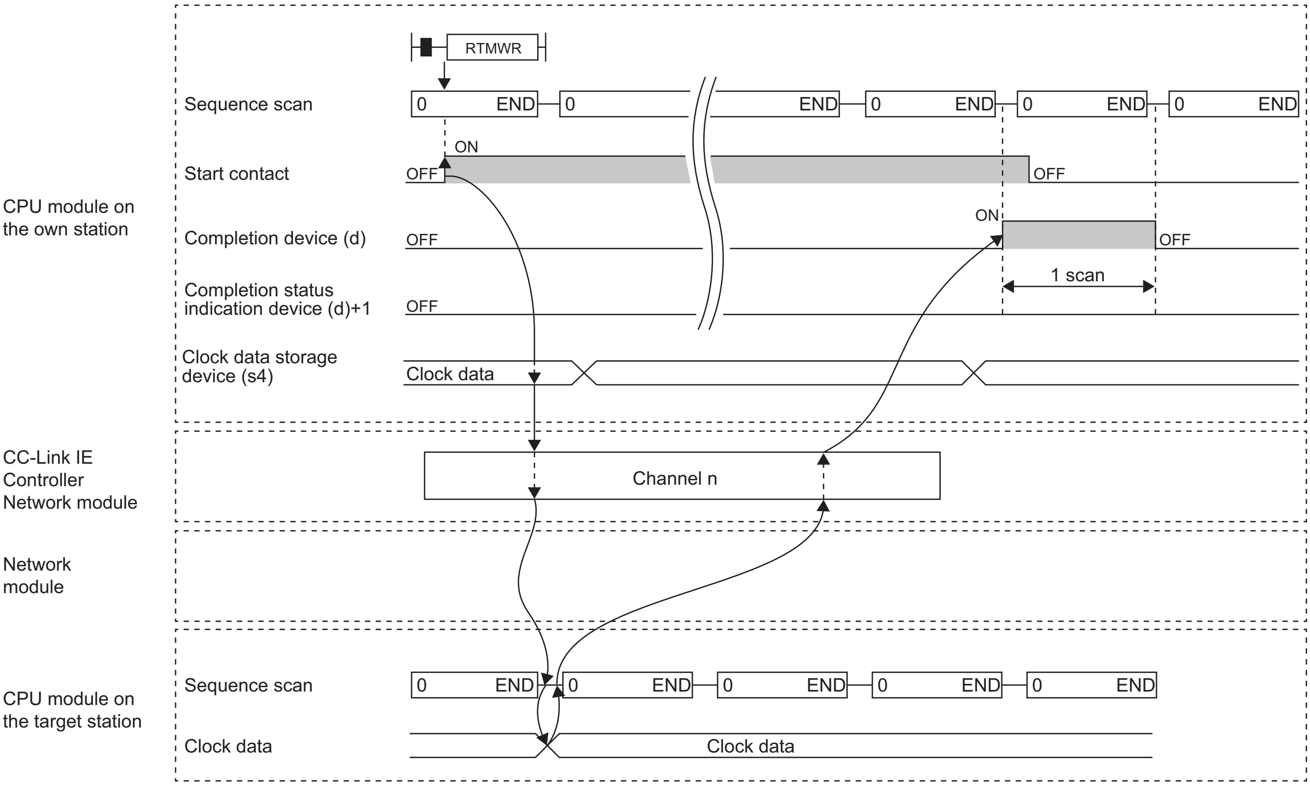
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the RTMWR instruction.

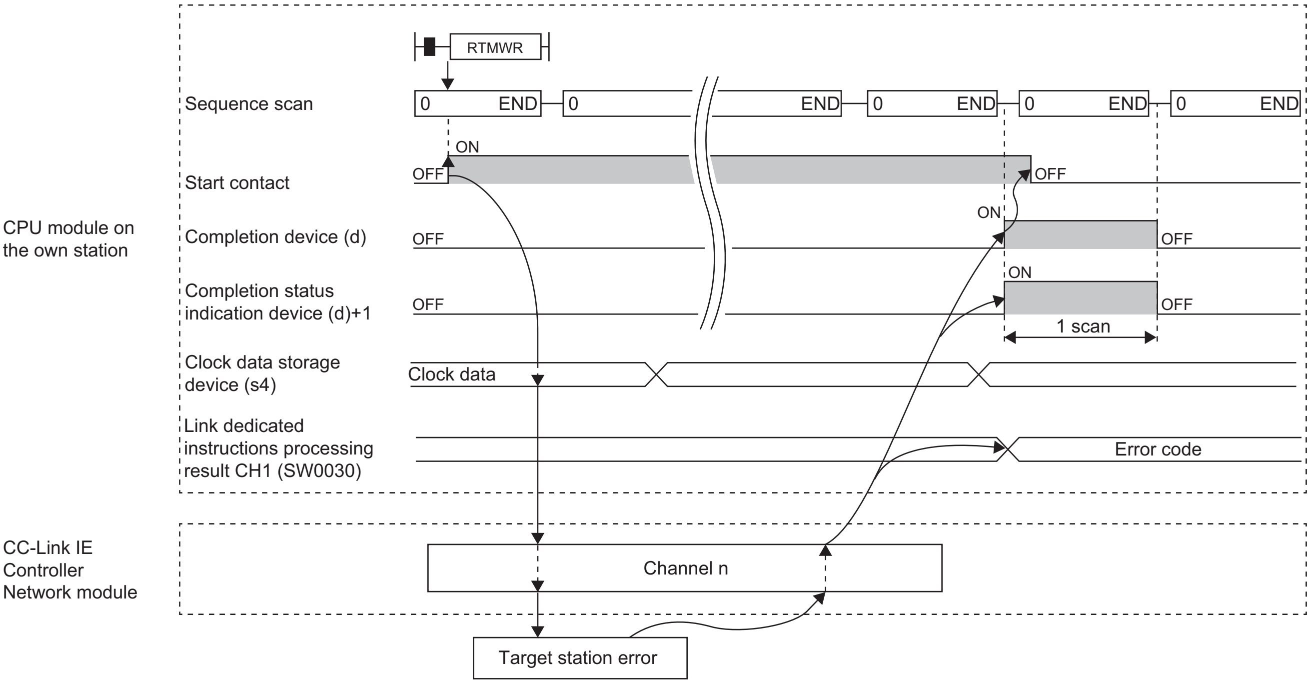
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the RTMWR instruction completes, and turns off during the next END processing.

- The following figure shows the RTMWR instruction execution timing.
  - When completed successfully



- When completed with an error



- If the J.RTMWR, G.RTMWR, or Z.RTMWR is used to execute processing, processing of one time is successively followed by the next processing while the start-up contact is on.
- If the JP.J.RTMWR, GP.J.RTMWR, or ZP.J.RTMWR is used to execute processing, processing is executed once at the start-up contact off-to-on rise.

Operation error

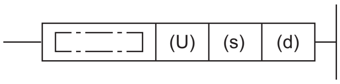
Error code (SW0030 to SW0037)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
6F00H to 6FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

17.5 Setting the Station Number to Own Station

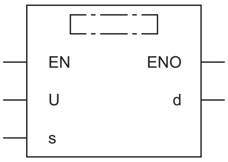
G(P).UINI, Z(P).UINI

RnCPU RnENCPU RnPCPU(Process) ✗RnPCPU(Redundant) RnSFCPU(Standard) ✗RnSFCPU(Safety)

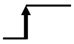
These instructions set the station number for the normal, or own, station whose station number has not yet been set.

Ladder	ST
	ENO:=G_UINI(EN,U,s,d); ENO:=GP_UINI(EN,U,s,d); ENO:=Z_UINI(EN,U,s,d); ENO:=ZP_UINI(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.UINI Z.UINI	
GP.UINI ZP.UINI	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	G(P).UINI	—	—	○	—	—	—	—	○	○	—	—	○
	Z(P).UINI	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*2	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

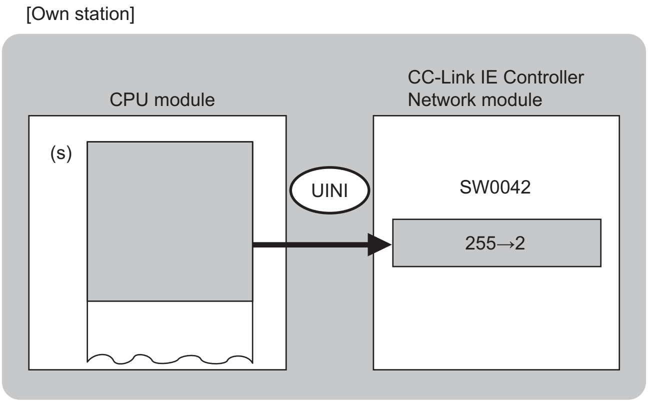
Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Not used	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Change target specification	0001H (fixed)	0001H	User
+3	Station number of own station	Specifies the station number to be set.	1 to 120	User
+4 to +9	Not used	—	—	—

Point

- The UINI instruction can be executed only once.
- If the UINI instruction is issued after the station number has been determined, it will complete with an error.
- If the UINI instruction completes with an error before the station number is determined, take corrective action to correct the error content before retrying to execute the instruction.

Processing details

- The station number of the normal, or own, station specified by (U) is set. After station number setting, the completion device specified by (d) turns on.



- The execution status and the completion status of the UINI instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

  - Completion device (d)

This device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

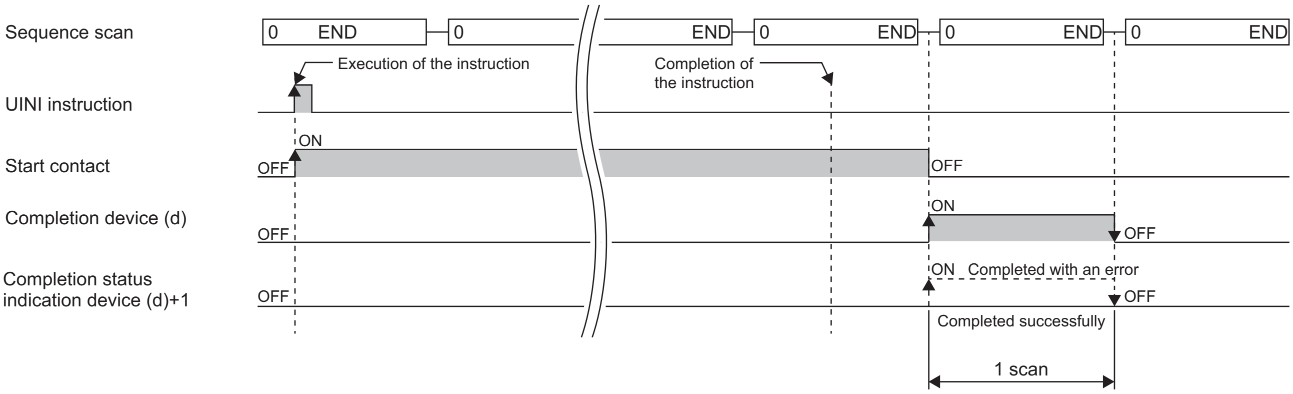
  - Completion status indication device (d)+1

This device turns on or off depending on the completion status of the UINI instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

- The following figure shows the UINI instruction execution timing.



Operation error

Error code ((S)+1)	Description
E000H to EFFFH	📖 MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)

18 CC-LINK IE FIELD NETWORK INSTRUCTIONS

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. For the instructions when MELSEC-Q series modules are used, refer to the manuals for each module used and create programs.

For precautions when using modules, refer to the following.

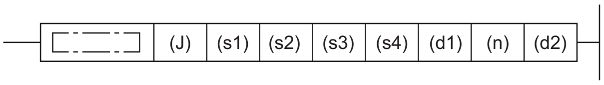
📖 MELSEC iQ-R Module Configuration Manual

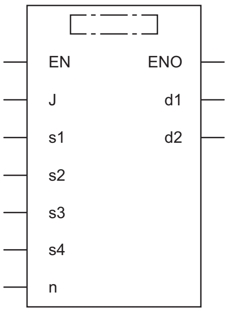
18.1 Reading Data from the Intelligent Device Station/ Remote Device Station (16-bit Address Specified)

JP.REMFR, ZP.REMFR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions read data from the buffer memory area of the intelligent device station or the remote device station (in units of words, 16-bit address specified).

Ladder	ST
	ENO:=JP_REMFR(EN,J,s1,s2,s3,s4,n,d1,d2); ENO:=ZP_REMFR(EN,J,s1,s2,s3,s4,n,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
JP.REMFR ZP.REMFR	↑

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J)	JP.REMFR	Target network number	1 to 239	16-bit unsigned binary	ANY16
	ZP.REMFR	Target network number	1 to 239	String	ANY16_OR_STRING_SINGLE
(s1)		Channel of the own station, which is used for the instruction*1	1 to 32	16-bit unsigned binary	ANY16
(s2)		Target station number	1 to 120	16-bit unsigned binary	ANY16
(s3)		Start I/O number of target station	■Head module: 00H to FEH*2 ■Modules other than the head module: 00H (fixed)	16-bit unsigned binary	ANY16
(s4)		Start address of the buffer memory area of the intelligent device station or the remote device station where the read-target data is stored	0000H to FFFFH (0 to 65535)	16-bit unsigned binary	ANY16
(d1)		Start device of the own station for storing the read data	—	Device name	ANY16*3
(n)		Number of read data points (in units of words)	1 to 240	16-bit unsigned binary	ANY16
(d2)		Device of the own station, which 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 A channel is used only for the REMFR, REMTO, REMFRD, and REMTOD instructions.
The same channel number used for the link dedicated instructions (such as the READ instruction) can be set.

*2 This is the start I/O number (first three digits in four-digit hexadecimal representation) of intelligent function modules.

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

Point

Use the REMFRD instruction if the start address of the buffer memory area (s4) exceeds FFFFH (65535).

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others
		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	$	(J)
(J)	JP.REMFR	—	—	—	—	—	—	—	—	—	—	—	○
	ZP.REMFR	—	—	—	—	—	—	—	—	—	—	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s3)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s4)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—
(n)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

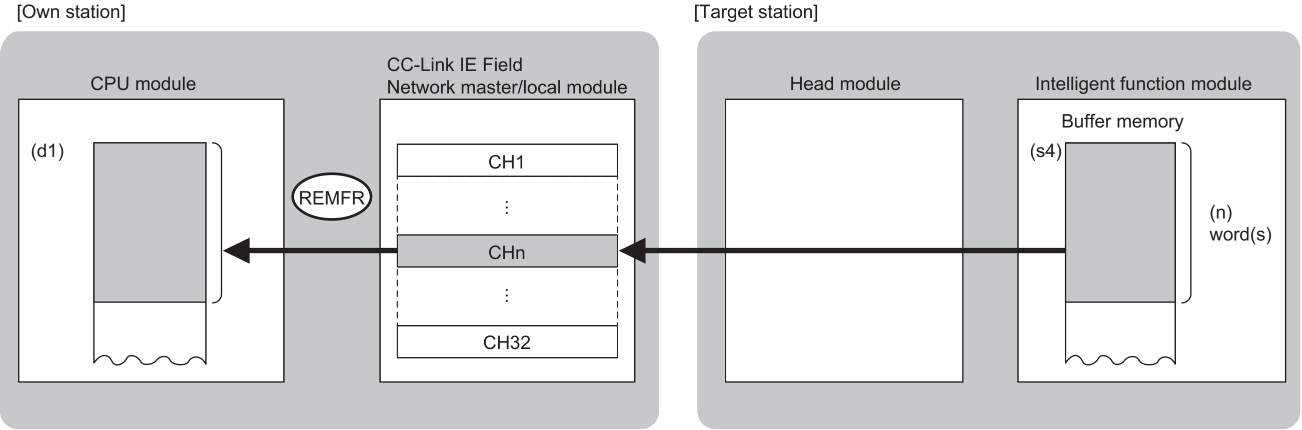
*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Processing details

- These instructions read the data from the start address (s4) of the buffer memory area of the intelligent device station or the remote device station to the word device ((d1) and later) of the own station for the specified number of words (n). Specify the target station by using (J), (s2), and (s3). The completion device (d2) turns on upon completion of the instruction.



CH: Channel

- The execution status and the completion status of the REMFR instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the REMFR instruction completes, and turns off during the next END processing.

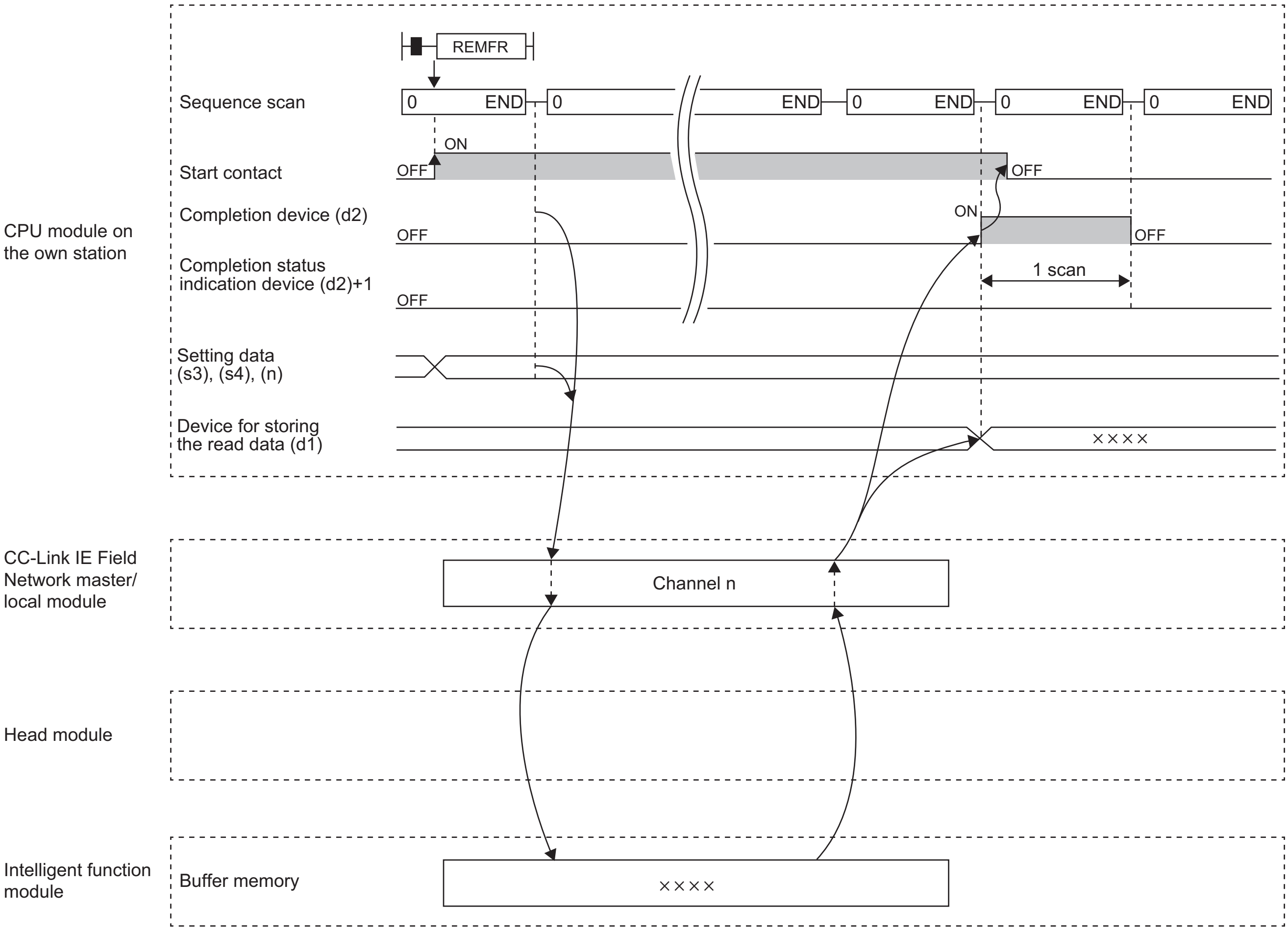
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the REMFR instruction.

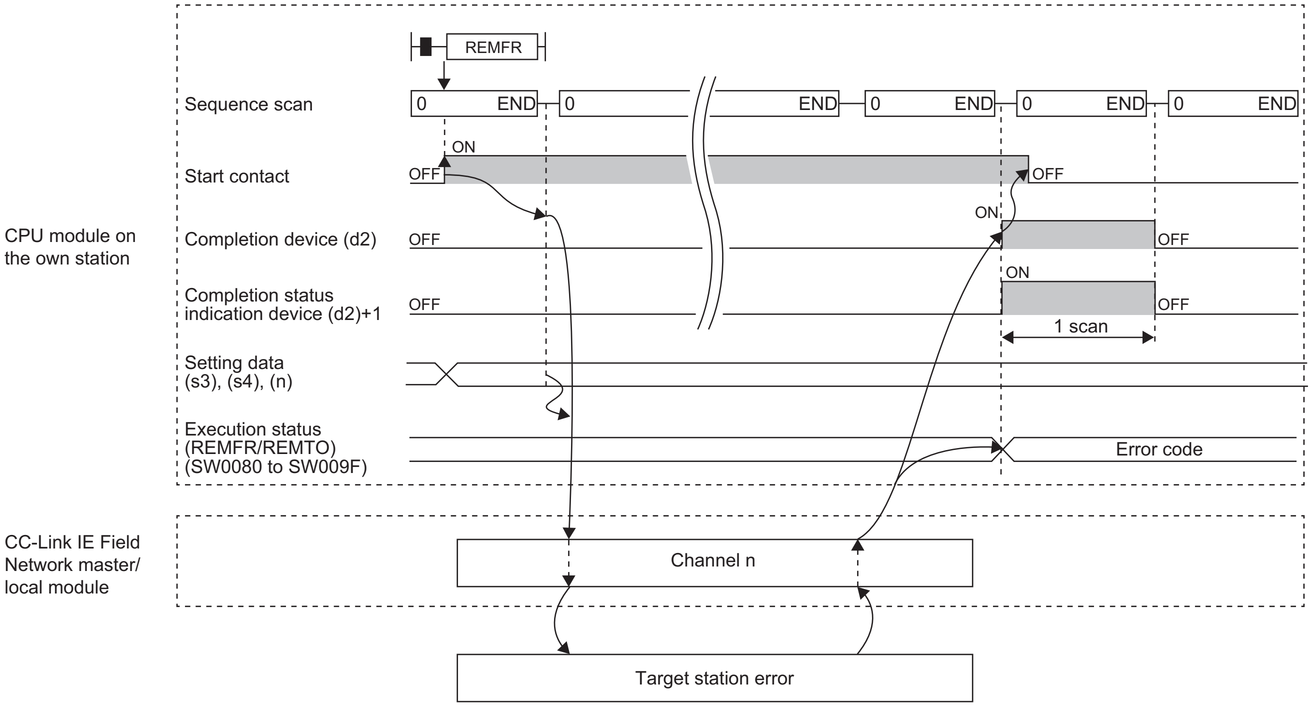
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the REMFR instruction completes, and turns off during the next END processing.

- The following figures show how the REMFR instruction operates when its execution has completed.
  - When completed successfully



  - When completed with an error



Point

For the REMFR instruction, the number of resends and the response wait timer can be set in the following link special register (SW) areas.

- Number of resends (REMFR/REMTO) (SW001A)
- Response wait timer (REMFR/REMTO) (SW001B)

If the number of resends (SW001A) is set, the REMFR instruction completes with an error after the following time has elapsed:

(Number of resends (REMFR/REMTO) + 1) × Response wait timer (REMFR/REMTO)

Note, however, that no retry is performed if the target station is disconnected. If the REMFR instruction completes with an error, execute the instruction again after the target station returns to the system.

Operation error

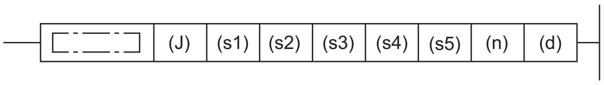
Error code (SW0080 to SW009F)	Description
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

18.2 Writing Data to the Intelligent Device Station/ Remote Device Station (16-bit Address Specified)

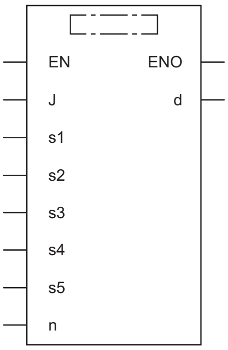
JP.REMTO, ZP.REMTO

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These instructions write data to the buffer memory area of the intelligent device station or the remote device station (in units of words, 16-bit address specified).

Ladder	ST
	ENO:=JP_REMTO(EN,J,s1,s2,s3,s4,s5,n,d); ENO:=ZP_REMTO(EN,J,s1,s2,s3,s4,s5,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.REMTO ZP.REMTO	↑

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J)	JP.REMTO	Target network number	1 to 239	16-bit unsigned binary	ANY16
	ZP.REMTO	Target network number	1 to 239	String	ANY16_OR_STRING_SINGLE
(s1)		Channel of the own station, which is used for the instruction*1	1 to 32	16-bit unsigned binary	ANY16
(s2)		Target station number	1 to 120	16-bit unsigned binary	ANY16
(s3)		Start I/O number of target station	■Head module: 00H to FEH*2 ■Modules other than the head module: 00H (fixed)	16-bit unsigned binary	ANY16
(s4)		Start address of the buffer memory area of the intelligent device station or the remote device station to which the data is written	0000H to FFFFH (0 to 65535)	16-bit unsigned binary	ANY16
(s5)		Start device of the own station, where the write-target data is stored	—	Device name	ANY16*3
(n)		Number of write data points (in units of words)	1 to 240	16-bit unsigned binary	ANY16
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 A channel is used only for the REMFR, REMTO, REMFRD, and REMTOD instructions.
The same channel number used for the link dedicated instructions (such as the READ instruction) can be set.

*2 This is the start I/O number (first three digits in four-digit hexadecimal representation) of intelligent function modules.

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

Point

Use the REMTOD instruction if the start address of the buffer memory area (s4) exceeds FFFFH (65535).

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (J)
		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	$	
(J)	JP.REMTO	—	—	—	—	—	—	—	—	—	—	—	○
	ZP.REMTO	—	—	—	—	—	—	—	—	—	—	○	○
(s1)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s3)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s4)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s5)		—	—	○*2	—	—	—	—	○	—	—	—	—
(n)		○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d)		○*1	—	○*3	—	—	—	—	—	—	—	—	—

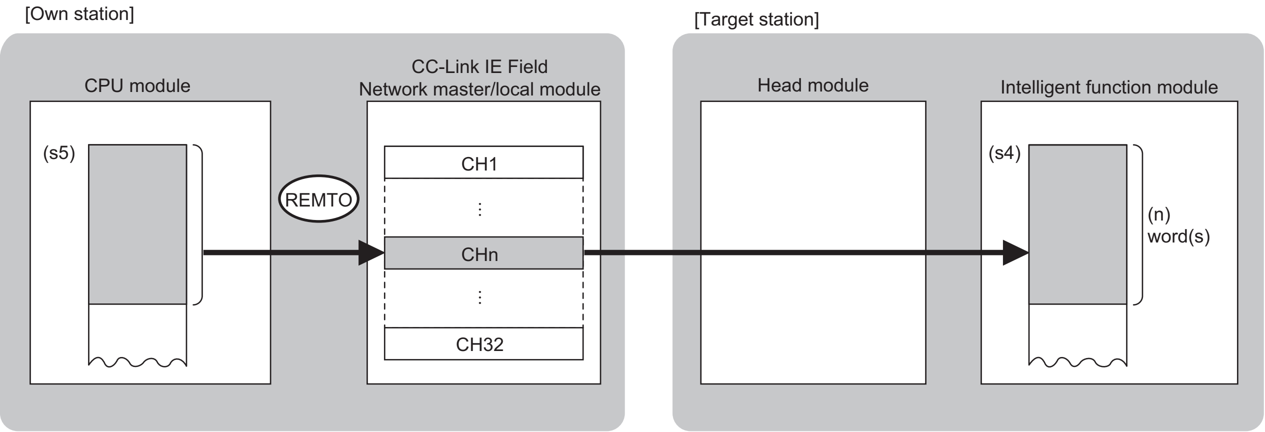
*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Processing details

- This instruction writes the data from the start device (s5) of the own station to the buffer memory area ((s4) and later) of the intelligent device station or the remote device station for the specified number of words (n). Specify the target station by using (J), (s2), and (s3). The completion device (d) turns on upon completion of the instruction.



CH: Channel

- The execution status and the completion status of the REMTO instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the REMTO instruction completes, and turns off during the next END processing.

- Completion status indication device (d)+1

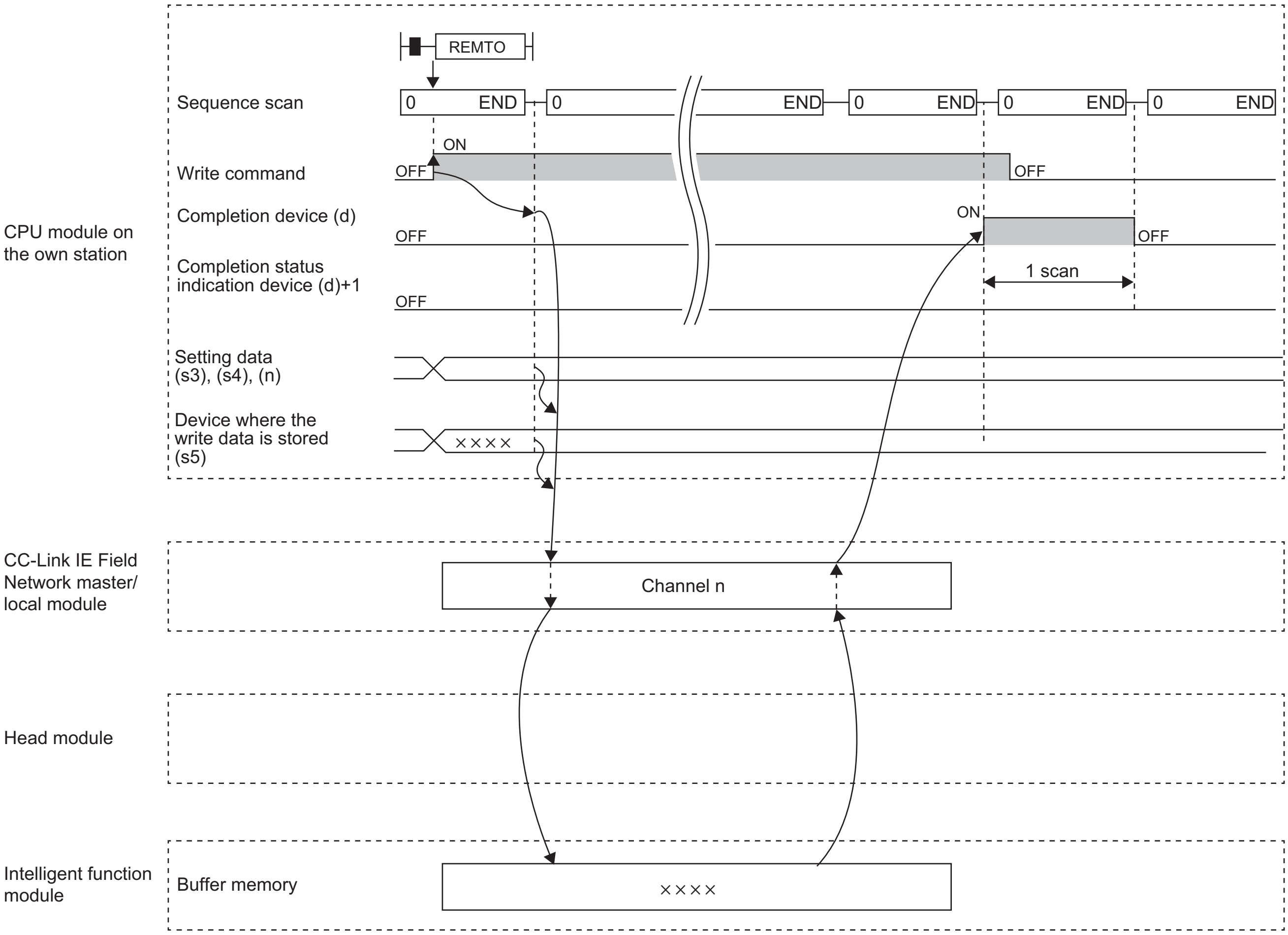
This device turns on or off depending on the completion status of the REMTO instruction.

When completed successfully: The device remains off.

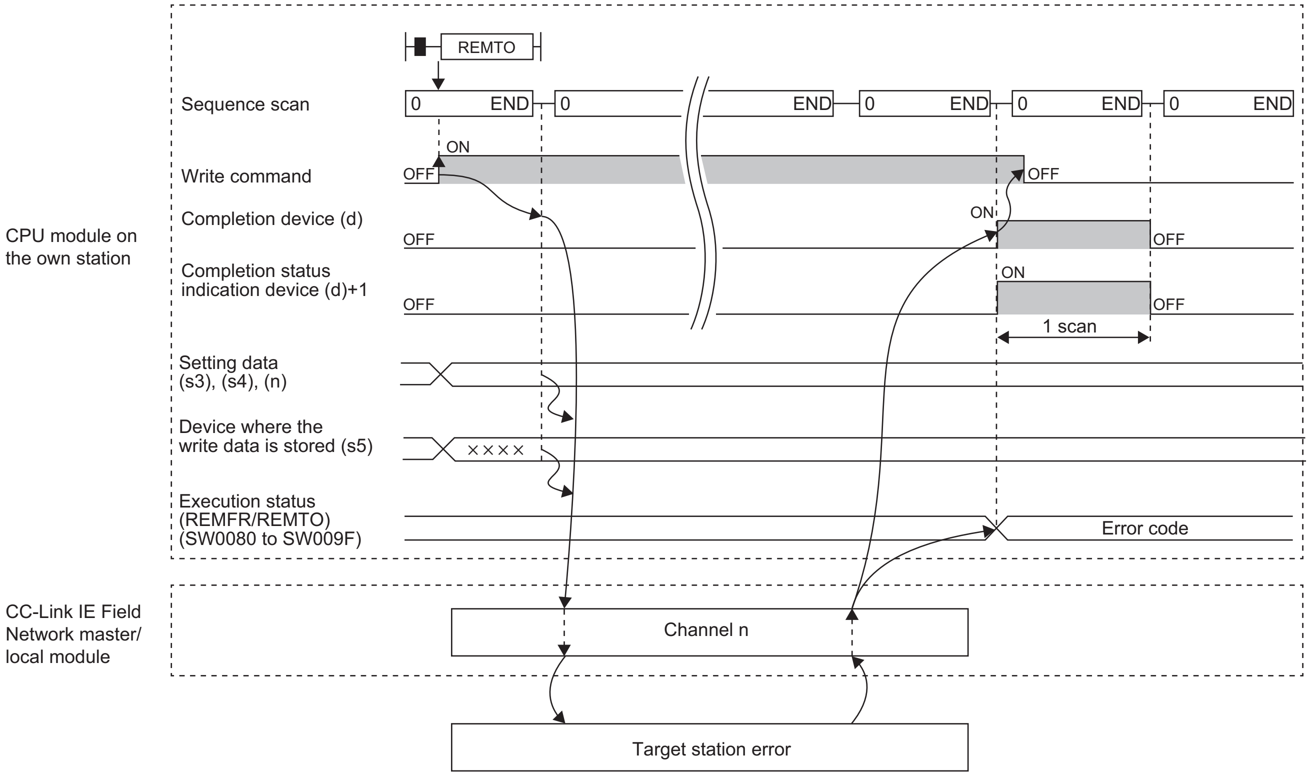
When completed with an error: The device turns on during END processing of the scan where the REMTO instruction completes, and turns off during the next END processing.

- The following figures show how the REMTO instruction operates when its execution has completed.

- When completed successfully



- When completed with an error



Point

For the REMTO instruction, the number of resends and the response wait timer can be set in the following link special register (SW) areas.

- Number of resends (REMFR/REMTO) (SW001A)
- Response wait timer (REMFR/REMTO) (SW001B)

If the number of resends (SW001A) is set, the REMTO instruction completes with an error after the following time has elapsed:

(Number of resends (REMFR/REMTO) + 1) × Response wait timer (REMFR/REMTO)

Note, however, that no retry is performed if the target station is disconnected. If the REMTO instruction completes with an error, execute the instruction again after the target station returns to the system.

Operation error

Error code (SW0080 to SW009F)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

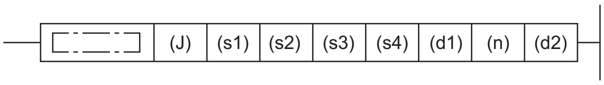
18.3 Reading Data from the Intelligent Device Station/ Remote Device Station (32-bit Address Specified)

JP.REMFRD

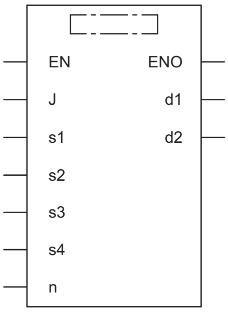
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

- The RnCPU and RnENCPU with firmware version "22" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RnPCPU (process) and the RnPCPU (redundant) with firmware version "04" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RnSFCPU (standard) with firmware version "06" or later supports this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RJ71GF11-T2 and RJ71EN71 with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)

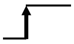
This instruction reads data from the buffer memory area of the intelligent device station or the remote device station (in units of words, 32-bit address specified).

Ladder	ST
	ENO:=JP_REMFRD(EN,J,s1,s2,s3,s4,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.REMFRD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(J)	Target network number	1 to 239	16-bit unsigned binary	ANY16
(s1)	Channel of the own station, which is used for the instruction[1]	1 to 32	16-bit unsigned binary	ANY16
(s2)	Target station number[4]	1 to 120	16-bit unsigned binary	ANY16
(s3)	Start I/O number of target station	■Head module: 00H to FEH[2] ■Modules other than the head module: 00H (fixed)	16-bit unsigned binary	ANY16
(s4)	Start address of the buffer memory area of the intelligent device station or the remote device station where the read-target data is stored	0000H to FFFFFFFFH (0 to 4294967295)	32-bit unsigned binary	ANY32
(d1)	Start device of the own station for storing the read data	—	Device name	ANY16[3]
(n)	Number of read data points (in units of words)	1 to 240	16-bit unsigned binary	ANY16
(d2)	Device of the own station, which 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 A channel is used only for the REMFR, REMTO, REMFRD, and REMTOD instructions.
The same channel number used for the link dedicated instructions (such as the READ instruction) can be set.

*2 This is the start I/O number (first three digits in four-digit hexadecimal representation) of intelligent function modules.

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

*4 When a station in a redundant system is targeted, specify the station number in the control system by using the module FB (RedundantSystem_GetAddress).
📖 MELSEC iQ-R Ethernet/CC-Link IE Function Block Reference

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J)
	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	$	
(J)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(s2)	○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(s3)	○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(s4)	○[1]	—	○[2]	—	—	○	—	○	○	—	—	—
(d1)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(n)	○[1]	—	○[2]	—	—	—	—	○	○	—	—	—
(d2)	○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

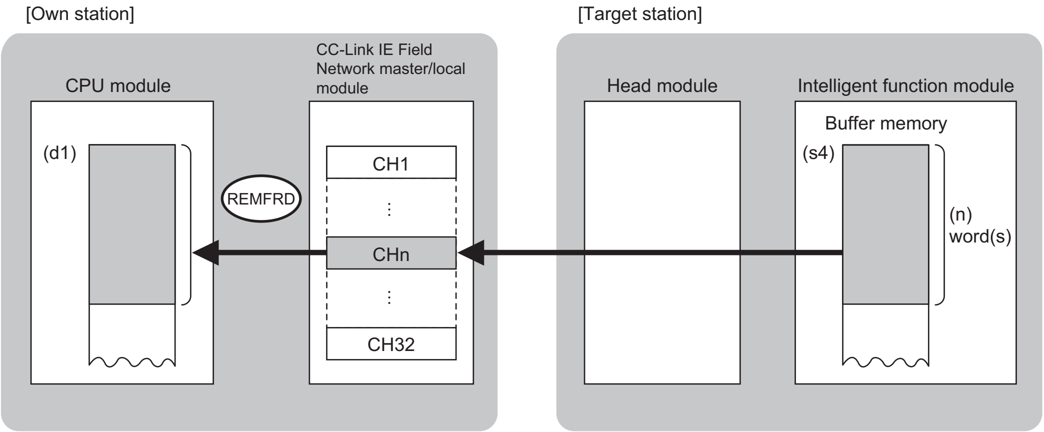
*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Processing details

- This instruction reads the data from the start address (s4) of the buffer memory area of the intelligent device station or the remote device station to the word device ((d1) and later) of the own station for the specified number of words (n). Specify the target station by using (J), (s2), and (s3). The completion device (d2) turns on upon completion of the instruction.



CH: Channel

- The execution status and the completion status of the REMFRD instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the REMFRD instruction completes, and turns off during the next END processing.

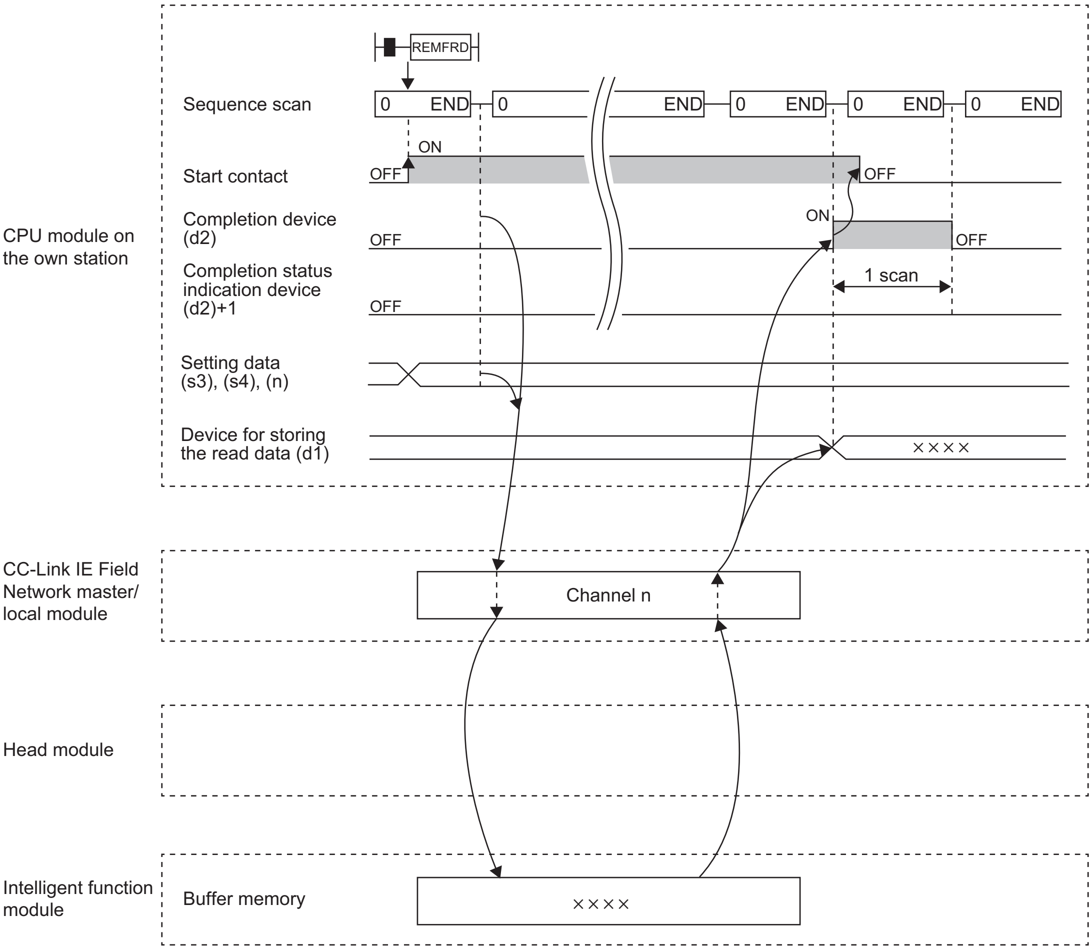
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the REMFRD instruction.

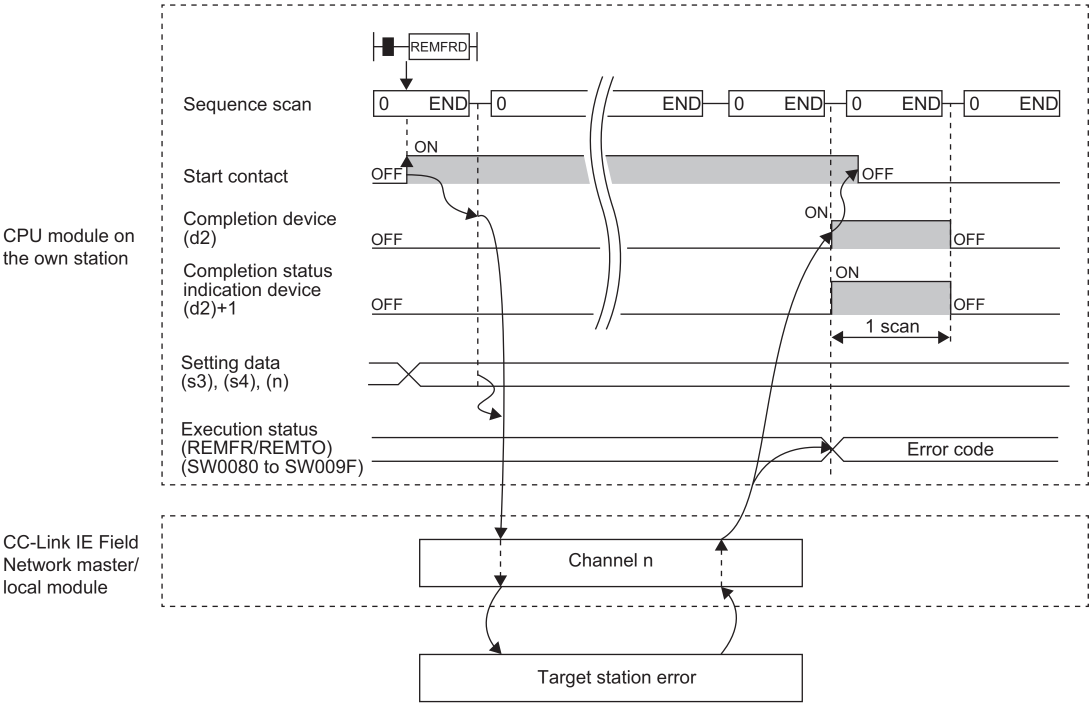
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the REMFRD instruction completes, and turns off during the next END processing.

- The following figures show how the REMFRD instruction operates when its execution has completed.
  - When completed successfully



  - When completed with an error



Point

For the REMFRD instruction, the number of resends and the response wait timer can be set in the following link special register (SW) areas.

- Number of resends (REMFR/REMTO) (SW001A)
- Response wait timer (REMFR/REMTO) (SW001B)

If the number of resends (SW001A) is set, the REMFRD instruction completes with an error after the following time has elapsed:

(Number of resends (REMFR/REMTO) + 1) × Response wait timer (REMFR/REMTO)

Note, however, that no retry is performed if the target station is disconnected. If the REMFRD instruction completes with an error, execute the instruction again after the target station returns to the system.

Operation error

Error code (SW0080 to SW009F)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

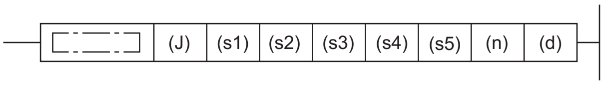
18.4 Writing Data to the Intelligent Device Station/ Remote Device Station (32-bit Address Specified)

JP.REMTOD

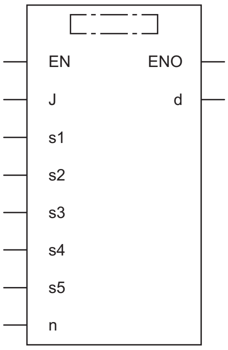
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

- The RnCPU and RnENCPU with firmware version "22" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RnPCPU (process) and the RnPCPU (redundant) with firmware version "04" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RnSFCPU (standard) with firmware version "06" or later supports this instruction. (Use an engineering tool with version "1.025B" or later.)
- The RJ71GF11-T2 and RJ71EN71 with firmware version "12" or later support this instruction. (Use an engineering tool with version "1.025B" or later.)

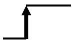
This instruction writes data to the buffer memory area of the intelligent device station or the remote device station (in units of words, 32-bit address specified).

Ladder	ST
	ENO:=JP_REMTOD(EN,J,s1,s2,s3,s4,s5,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
JP.REMTOD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(J)	Target network number	1 to 239	16-bit unsigned binary	ANY16
(s1)	Channel of the own station, which is used for the instruction*1	1 to 32	16-bit unsigned binary	ANY16
(s2)	Target station number*4	1 to 120	16-bit unsigned binary	ANY16
(s3)	Start I/O number of target station	■Head module: 00H to FEH*2 ■Modules other than the head module: 00H (fixed)	16-bit unsigned binary	ANY16
(s4)	Start address of the buffer memory area of the intelligent device station or the remote device station to which the data is written	0000H to FFFFFFFFH (0 to 4294967295)	32-bit unsigned binary	ANY32
(s5)	Start device of the own station, where the write-target data is stored	—	Device name	ANY16*3
(n)	Number of write data points (in units of words)	1 to 240	16-bit unsigned binary	ANY16
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 A channel is used only for the REMFR, REMTO, REMFRD, and REMTOD instructions.
The same channel number used for the link dedicated instructions (such as the READ instruction) can be set.

*2 This is the start I/O number (first three digits in four-digit hexadecimal representation) of intelligent function modules.

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

*4 When a station in a redundant system is targeted, specify the station number in the control system by using the module FB (RedundantSystem_GetAddress).
📖 MELSEC iQ-R Ethernet/CC-Link IE Function Block Reference

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J)
	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	$	
(J)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s2)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s3)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(s4)	○*1	—	○*2	—	—	○	—	○	○	—	—	—
(s5)	—	—	○*2	—	—	—	—	○	—	—	—	—
(n)	○*1	—	○*2	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

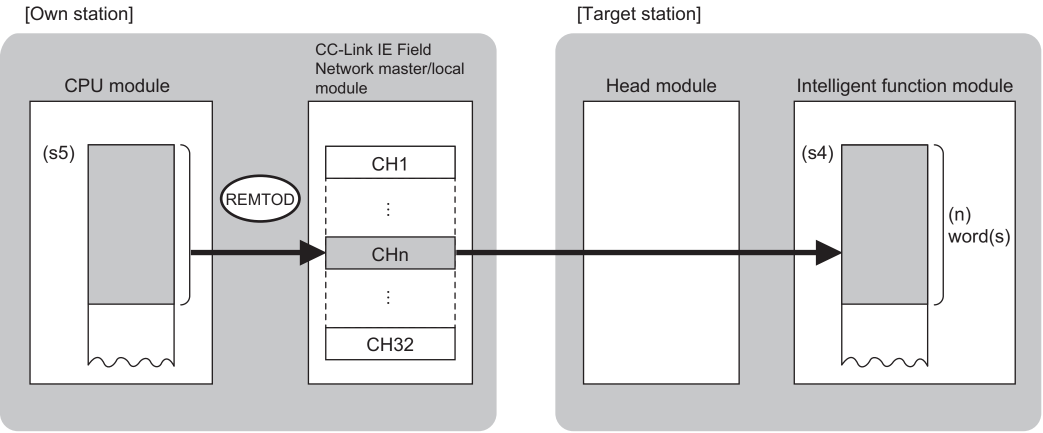
*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

Processing details

- This instruction writes the data from the start device (s5) of the own station to the buffer memory area ((s4) and later) of the intelligent device station or the remote device station for the specified number of words (n). Specify the target station by using (J), (s2), and (s3). The completion device (d) turns on upon completion of the instruction.



CH: Channel

- The execution status and the completion status of the REMTOD instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the REMTOD instruction completes, and turns off during the next END processing.

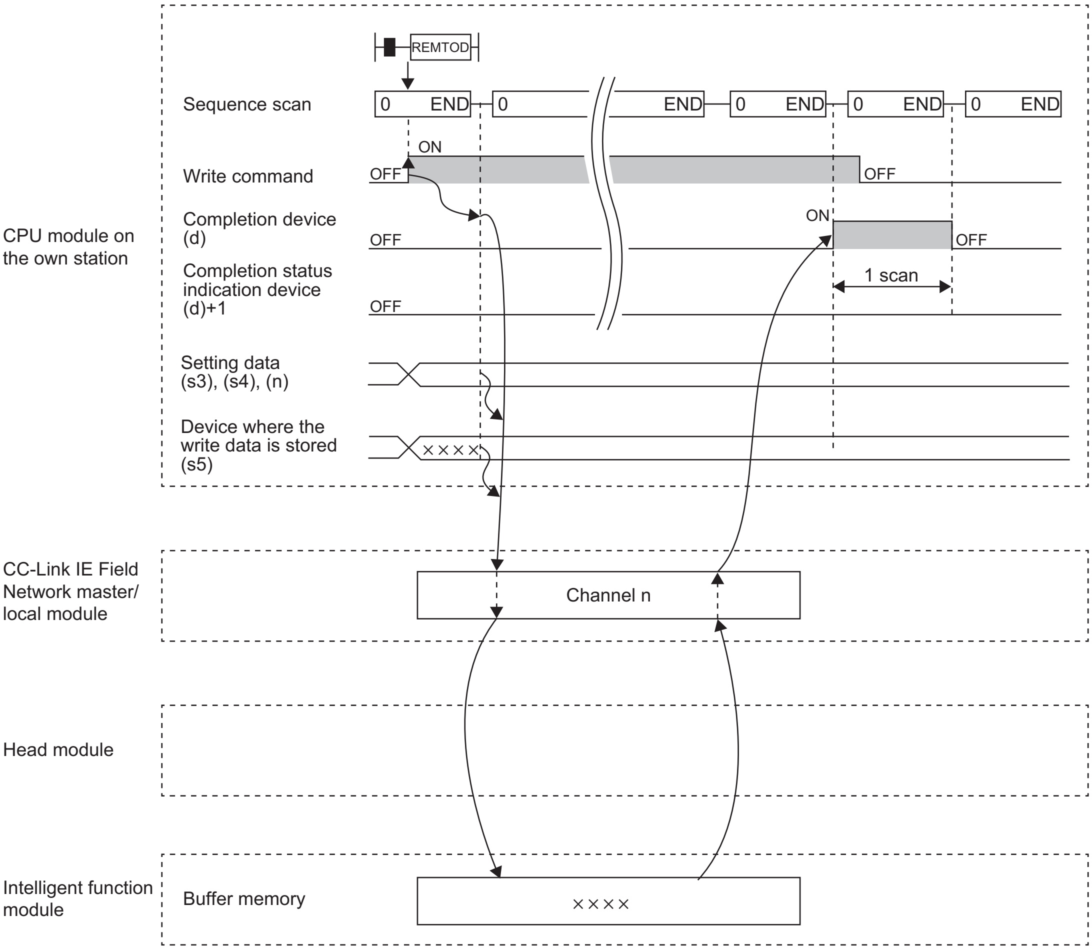
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the REMTOD instruction.

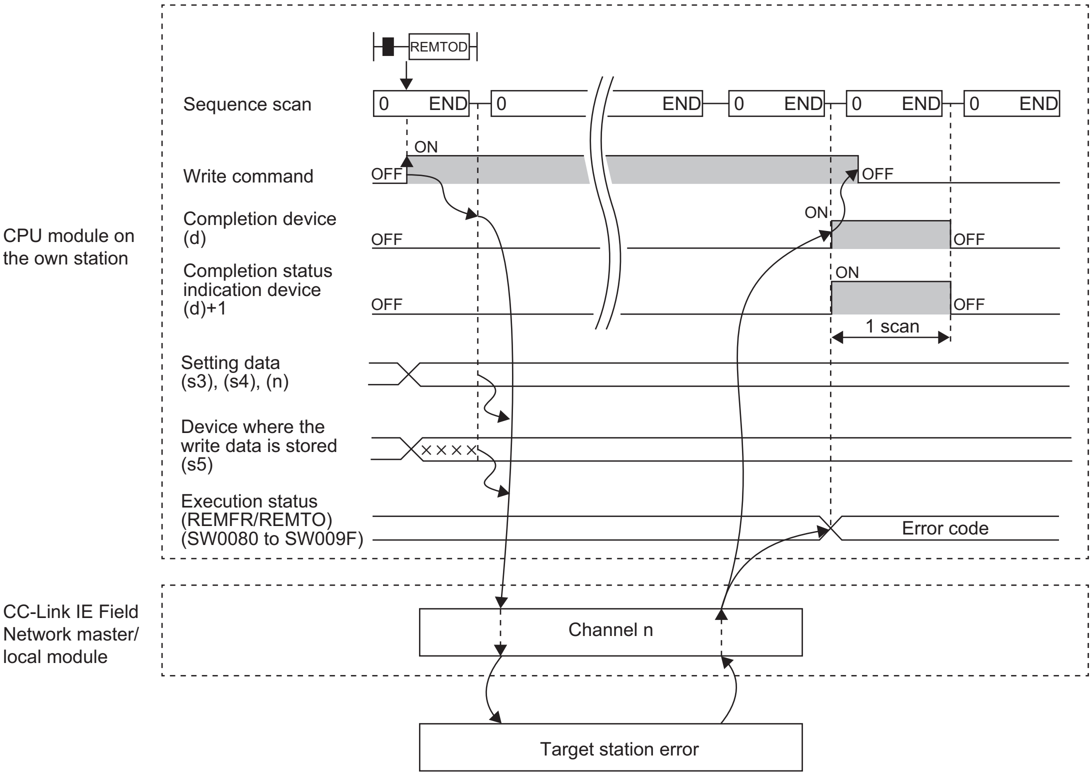
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the REMTOD instruction completes, and turns off during the next END processing.

- The following figures show how the REMTOD instruction operates when its execution has completed.
  - When completed successfully



  - When completed with an error



Point

For the REMTOD instruction, the number of resends and the response wait timer can be set in the following link special register (SW) areas.

- Number of resends (REMFR/REMTO) (SW001A)
- Response wait timer (REMFR/REMTO) (SW001B)

If the number of resends (SW001A) is set, the REMTOD instruction completes with an error after the following time has elapsed:

(Number of resends (REMFR/REMTO) + 1) × Response wait timer (REMFR/REMTO)

Note, however, that no retry is performed if the target station is disconnected. If the REMTOD instruction completes with an error, execute the instruction again after the target station returns to the system.

Operation error

Error code (SW0080 to SW009F)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

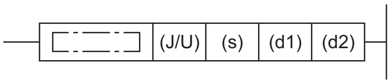
18.5 Reading the Module Model Information of an Intelligent Device Station

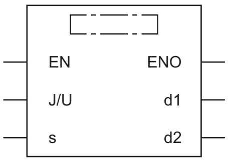
J(P).SINFTYRD, G(P)SINFTYRD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

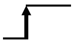
- The RnENCPU (network part), RJ71GF11-T2, and RJ71EN71 with firmware version "12" or later support these instructions. (Use an engineering tool with version "1.025B" or later.)

These instructions read the model names of modules and units used in an intelligent device station.

Ladder	ST
	ENO:=J_SINFTYRD(EN,J,s,d1,d2); ENO:=JP_SINFTYRD(EN,J,s,d1,d2); ENO:=G_SINFTYRD(EN,U,s,d1,d2); ENO:=GP_SINFTYRD(EN,U,s,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
J.SINFTYRD G.SINFTYRD	
JP.SINFTYRD GP.SINFTYRD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).SINFTYRD	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).SINFTYRD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)		Start device of the own station for storing the read data (A continuous area of 890 words is required.)	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] 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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).SINFTYRD	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).SINFTYRD	—	—	○	—	—	—	—	○	○	—	—	—	○
(s)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d2)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—	—

[*1] FX and FY cannot be used.

[*2] FD cannot be used.

[*3] T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	Error completion type	b15 ··· b8 b7 ··· b0 0 (2) (1) 0 (1) Error completion type (bit 7) Specify whether to set data at completion with an error. • 0: Do not set data at completion with an error in (s)+12 and later. • 1: Set data at completion with an error in (s)+12 and later. (2) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units	0000H 0080H 0100H 0180H	User
+1	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel[*1]	Specify the channel to be used by own station.	1 to 32	User
+3 to +4	—	Not used	—	System
+5	Target station number	Specify the station number of the target station.	1 to 120	User
+6	—	Not used	—	System
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s)+7. ■When "0" is specified in bit 8 of (s)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s)+0 • 0: 10s • 1 to 65535: 1 to 65535 × 100ms	0 to 65535	User
+9 to +10	—	Not used	—	System
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s)+12 and later is stored. Note that the data in (s)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	—	System
+17	Error-detected station number	The number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	—	System

*1 A channel used only for the SINFTYRD and SINFSTRD instructions. The same channel used for the remote instructions (such as REMFR and REMTO instructions) and the link dedicated instructions (such as the READ and SEND instructions) can be set.

Processing details

- These instructions read the model names of modules and units used in an intelligent device station, and store the data to the device specified by (d1) and later.

Operand	Item
(d1)+0 to (d1)+9	Main base unit: power supply information 1
(d1)+10 to (d1)+19	Main base unit: power supply information 2
(d1)+20 to (d1)+29	1st extension base unit: power supply information 1
(d1)+30 to (d1)+39	1st extension base unit: power supply information 2
⋮	⋮
(d1)+140 to (d1)+149	7th extension base unit: power supply information 1
(d1)+150 to (d1)+159	7th extension base unit: power supply information 2
(d1)+160 to (d1)+169	Main base unit information
(d1)+170 to (d1)+179	1st extension base unit information
⋮	⋮
(d1)+230 to (d1)+239	7th extension base unit information
(d1)+240 to (d1)+249	CPU slot: product information
(d1)+250 to (d1)+259	I/O slot 0: product information
(d1)+260 to (d1)+269	I/O slot 1: product information
⋮	⋮
(d1)+880 to (d1)+889	I/O slot 63: product information

- The result of instruction execution is stored in the 1st word and a model name is stored in the 2nd to 10th words of each item. The following table lists the values to be stored in the corresponding words.

Condition	1st word	2nd to 10th words
The read target module or unit has a model name.	0	Model name held by a module or a unit
The read target module or unit does not have a model name.	1	Character string consisting of module type and the number of points
The read target slot is empty.	-1*2	0000H
The read target module is being changed online.		

*2 If the read target module has a hardware failure, -1 may be stored in the 1st word.

- When the read target module has a model name, the model name is stored in the 2nd word and later according to the following rules.
  - The name occupies nine words.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - The model name held by a module or a unit is stored. (Note that it may differ from the one written on the rating plate.)

Ex.

The following table lists examples of model names stored.

Target module	Character string example (module type)
Remote head module	RJ72GF15-T2
I/O module	INPUT_16
Network module	RJ71EN71(E+E)

Point

If the model name in the I/O assignment setting and that of the mounted module differ, the model name held by the mounted module is stored.

- When the read target module does not have a model name, the character string is stored in the 2nd word and later according to the following rules.
  - The name occupies nine words.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - A character string consisting of a combination of "character string indicating the module type" and "character string indicating the number of points" is stored.

Ex.

The following table lists examples of character strings stored (module type).

Target module	Model name example
Input module	INPUT_16
Output module	OUTPUT_32
I/O combined module	MIXED_64
Intelligent function module	INTELLIGENT_128

The following table lists examples of character strings stored (number of points).

Number of points	Character string example (number of points)
16 points	_16
32 points	_32
48 points	_48
64 points	_64
128 points	_128
256 points	_256
512 points	_512
1024 points	_1024

Point

If the number of points in the I/O assignment setting and that of the mounted module differ, the number of points of the mounted module is stored.

Operation error

Error code ((s)+1)	Description
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

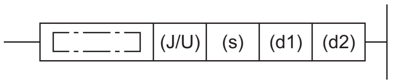
18.6 Reading the Module Status Information of an Intelligent Device Station

J(P).SINFSTRD, G(P)SINFSTRD

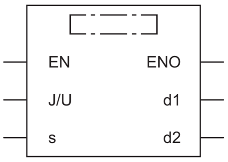
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

- The RnENCPU (network part), RJ71GF11-T2, and RJ71EN71 with firmware version "12" or later support these instructions. (Use an engineering tool with version "1.025B" or later.)

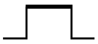
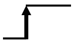
These instructions read the status information of modules and units used in an intelligent device station.

Ladder	ST
	ENO:=J_SINFSTRD(EN,J,s,d1,d2); ENO:=JP_SINFSTRD(EN,J,s,d1,d2); ENO:=G_SINFSTRD(EN,U,s,d1,d2); ENO:=GP_SINFSTRD(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.SINFSTRD G.SINFSTRD	
JP.SINFSTRD GP.SINFSTRD	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).SINFSTRD	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).SINFSTRD	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s)		Start device of the own station where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)		Start device of the own station for storing the read data (A continuous area of 356 words is required.)	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

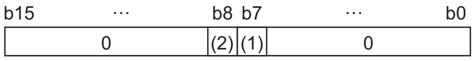
■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		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	$	J	U
(J/U)	J(P).SINFSTRD	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).SINFSTRD	—	—	○	—	—	—	—	○	○	—	—	—	○
(s)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○[*2]	—	—	—	—	○	—	—	—	—	—
(d2)		○[*1]	—	○[*3]	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 FD cannot be used.
*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	Error completion type	 (1) Error completion type (bit 7) Specify whether to set data at completion with an error. • 0: Do not set data at completion with an error in (s)+12 and later. • 1: Set data at completion with an error in (s)+12 and later. (2) Arrival check time setting (bit 8) • 0: 1s units • 1: 100ms units	0000H 0080H 0100H 0180H	User
+1	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Own station channel[*1]	Specify the channel to be used by own station.	1 to 32	User
+3 to +4	—	Not used	—	System
+5	Target station number	Specify the station number of the target station.	1 to 120	User
+6	—	Not used	—	System
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s)+7. ■When "0" is specified in bit 8 of (s)+0 • 0: 10s • 1 to 32767: 1 to 32767s ■When "1" is specified in bit 8 of (s)+0 • 0: 10s • 1 to 65535: 1 to 65535 × 100ms	0 to 65535	User
+9 to +10	—	Not used	—	System
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s)+12 and later is stored. Note that the data in (s)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. (No information is stored if an error is detected in the own station.) • 1 to 239 (Network number)	—	System
+17	Error-detected station number	The station number of the station where an error was detected is stored. (No information is stored if an error is detected in the own station.) • 125: Master station • 1 to 120: Local station, intelligent device station, submaster station	—	System

*1 A channel used only for the SINFTYRD and SINFSTRD instructions. The same channel used for the remote instructions (such as REMFR and REMTO instructions) and the link dedicated instructions (such as the READ and SEND instructions) can be set.

Processing details

- These instructions read the status information of modules and units used in an intelligent device station, and store the data to the device specified by (d1) and later.

Operand	Item
(d1)+0 to (d1)+3	Main base unit: power supply information 1
(d1)+4 to (d1)+7	Main base unit: power supply information 2
(d1)+8 to (d1)+11	1st extension base unit: power supply information 1
(d1)+12 to (d1)+15	1st extension base unit: power supply information 2
⋮	⋮
(d1)+56 to (d1)+59	7th extension base unit: power supply information 1
(d1)+60 to (d1)+63	7th extension base unit: power supply information 2
(d1)+64 to (d1)+67	Main base unit information
(d1)+68 to (d1)+71	1st extension base unit information
⋮	⋮
(d1)+92 to (d1)+95	7th extension base unit information
(d1)+96 to (d1)+99	CPU slot: product information
(d1)+100 to (d1)+103	I/O slot 0: product information
(d1)+104 to (d1)+107	I/O slot 1: product information
⋮	⋮
(d1)+352 to (d1)+355	I/O slot 63: product information

- The following table lists the values to be stored in the corresponding words.

○: Supported, —: Not supported[*2]

Device	Bit	Item	Description	Power supply module	Base unit	Modules mounted on the CPU slot and slot No.0 to No.63
1st word	b0 to b3	Reserved	Fixed to 0	—	—	—
	b4 to b7	Module type[*3]	0000:Input module, or not set (default) 0010:Output module 0100:I/O combined module (different numbers) 0110:I/O combined module (same number) 1000:Intelligent function module Further details can be checked in b0 to b3 of the 3rd word.	○	○	○
	b8 to b10	Series[*3]	000:— (default) 010:MELSEC-Q series module 011:MELSEC iQ-R series module 111:Unknown	○	○	○
	b11 to b15	Reserved	Fixed to 0	—	—	—
2nd word	b0 to b1	Error status[*3]	00: No error (default) 01: Minor error 10: Moderate error 11: Major error	○	○	○
	b2	Module ready status[*3]	0: Not ready (default) 1: Ready	—	—	○
	b3 to b5	Reserved	Fixed to 0	—	—	—
	b6 to b7	Inter-module synchronization[*3]	00: Not synchronized (default) 01: Preparing for synchronization 10: Synchronized 11: Synchronization error	—	—	○
	b8 to b9	Reserved	Fixed to 0	—	—	—
	b10	External power supply[*3]	0: Normal operation (default) 1: Power off	—	—	○
	b11	Fuse[*3]	0: Normal operation (default) 1: Fuse blown	—	—	○
	b12	Reserved	Fixed to 0	—	—	—
	b13	Online module change	0: Not being performed (default) 1: Module being changed online	—	—	○
	b14	Reserved	Fixed to 0	—	—	—
	b15	Module access	0: Disabled (default) 1: Enabled	○	○	○[*4]

Device	Bit	Item	Description	Power supply module	Base unit	Modules mounted on the CPU slot and slot No.0 to No.63
3rd word	b0 to b3	Module type details[3]	0000:Not set (default) 0001:Remote head module 0010:Power supply module 0011:Base unit 0100:Input module 0101:Output module 0110:I/O combined module (different numbers) 0111:I/O combined module (same number) 1000:Intelligent function module 1111:Module or unit other than above	○	○	○
	b4 to b7	Reserved	Fixed to 0	—	—	—
	b8	Safety function setting[3]	0: Invalid (default) 1: Valid	—	—	○
	b9	Reserved	Fixed to 0	—	—	—
	b10	Safety operation status[3]	0: Not operating (default) 1: Operating	—	—	○
	b11	Reserved	Fixed to 0	—	—	—
	b12	Redundant function setting[3]	0: Invalid (default) 1: Valid	—	—	○[5]
	b13	Reserved	Fixed to 0	—	—	—
	b14 to b15	Redundant operation status[3]	00: Not operating (default) 01: Standby system 10: Control system	—	—	○[5]
4th word	—	Latest error code	0000:No error (default) Other than 0000:Error code	—	—	○

[2] If a module or a unit does not support the information, a default value is stored.

[3] A value is stored only when the value stored in b15 of the 2nd word (Module access) is 1 (enabled). When the value stored in b15 of the 2nd word is 0 (disabled), a default value is stored.

[4] When a module is mounted on a slot and accessible, 1 is stored. When a module occupying two or more slots is mounted, a value is stored only for a slot where a module is actually mounted.

[5] A value is stored only for a remote head module.

Operation error

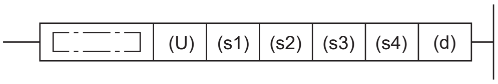
Error code ((s)+1)	Description
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

18.7 Setting Parameters

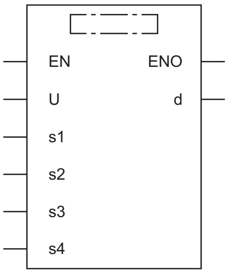
G(P).CCPASET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

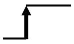
These instructions set the parameters in the CC-Link IE Field Network master and local modules.

Ladder	ST
	ENO:=G_CCPASET(EN,U,s1,s2,s3,s4,d); ENO:=GP_CCPASET(EN,U,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.CCPASET	
GP.CCPASET	

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 own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)	Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(s2)	Own station start device containing the network configuration settings	—	Device name	ANY16[1]
(s3)	Own station start device containing the reserved station specification	—	Device name	ANY16[1]
(s4)	Own station start device containing the error invalid station setting	—	Device name	ANY16[1]
(d)	Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(s3)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(s4)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(d)	○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range[1]**	**Set by**
+0	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+1	Setting flag	Specifies whether the setting data in (s2) to (s4) are valid/invalid b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 0 | (12) | (11) | (10) | (9) | (8) | (7) | (6) | 0 | (5) | (4) | (3) | (2) | (1) (1) Presence of network configuration setting data (bit 0) • 0: Invalid[2] • 1: Valid (2) Presence of reserved station setting data (bit 1) • 0: Invalid[2] • 1: Valid (3) Presence of error invalid station setting data (bit 2) • 0: Invalid[2] • 1: Valid (4) Presence of submaster function (bit 3) • 0: Invalid[2] • 1: Valid (5) Presence of IP packet transfer function (bits 4, 5) • 00: Invalid • 01: Valid (6) Data link faulty station setting (bit 8) • 0: Cleared • 1: Held (7) CPU STOP time output setting (bit 9) • 0: Held • 1: Cleared (8) Link scan mode (bit 10) • 0: Sequence scan unsynchronized • 1: Sequence scan synchronized (9) Network topology setting (bit 11) • 0: Line topology, star topology, or mixed topology of line and star • 1: Ring topology (10)CPU error time output setting (bit 12) • 0: Cleared • 1: Held (11)Master station return time operation setting (bit 13) Enabled only when the station type is master. • 0: Returning as the master operating station • 1: Returning as the submaster operating station (12)Submaster setting (bit 14) Enabled only when the station type is submaster. • 0: Operating with the parameters of the master station • 1: Operating with the parameters of the own, or submaster, station	Left	User
+2	Total number of slave stations	Sets the total number of slave stations connected.	1 to 121[3]	User

Operand: (s1)

Device	Item	Description	Setting range[1]	Set by
+3	Constant link scan time	Sets the constant link scan time. • 0: No operator • 1 to 200: Constant link scan time	0 to 200	User
+4 +5	IP address	Sets the IPv4 network address (i.e. IP address) when the IP packet transfer function is used. (s1)+5 | (s1)+4 (s1)+4: Dummy (s1)+5: Network address	00000001H to FFFFFFFFH	User

[1] If a value outside the setting range is set, the instruction will complete with an error.

[2] If "Invalid" is set, the default parameter will be used.

[3] Set value 121 is available only if the submaster function is used.

Point

For the startup in the local station, only Data link faulty station setting (bit 8 of (s1)+1) and CPU STOP time output setting (bit 9 of (s1)+1) will be valid. Any other changed settings will be ignored during execution of the instruction, without causing an error.

■Network configuration setting data (for master station setting only)

Operand: (s2)

Device	Item		Description	Setting range[1]	Set by
+0	For 1st module	Slave station setting information	Sets the station type, the number of occupied stations, and the station number. b15 … b12 b11 … b8 b7 … b0 (3) | (2) | (1) (1) Station number • 1 to 120: Station number • 0: Master station (2) Number of occupied stations • 1: Fixed (3) Station type • 0: Remote I/O station • 1: Remote device station • 2: Intelligent device station • 3: Local station (master-slave system) • 4: Submaster station • F: Master station	Left	User
+1		RX/RY offset	Sets the offset value from the head of RX/RY (in units of 16 points).[2]	0 to 16368	User
+2		Number of RX/RY points	Sets the number of RX/RY, LB points.[2] • If the station type is master or local: 0 to 2048 points • If the station type is intelligent device: 0 to 2048 points (in units of 16 points) • If the station type is remote I/O: 0 to 64 points (in units of 16 points) • If the station type is remote device: 0 to 128 points (in units of 16 points)	0 to 2048 0 to 64 0 to 128	User
+3		RWr/RWw offset	Sets the offset value from the start of RWr/RWw (in units of 4 points).[2]	0 to 8188	User
+4		Number of RWr/RWw points	Sets the number of RWr/RWw, LW points.[2] • If the station type is master or local: 0 to 256 points (in units of 16 points) • If the station type is intelligent device: 0 to 1024 points (in units of 4 points) • If the station type is remote device: 0 to 64 points (in units of 4 points)	0 to 256 0 to 1024 0 to 64	User
+5 to +594	⋮				User

Operand: (s2)					
Device	**Item**		**Description**	**Setting range**[*1]	**Set by**
+595	For 120th module	Slave station setting information	Sets the station type.	Same as (s2)+0	User
+596		RX/RY offset	Sets the offset value from the head of RX/RY (in units of 16 points).[*2]	0 to 16368	User
+597		Number of RX/RY points	Sets the number of RX/RY points.[*2] • If the station type is master or local: 0 to 2048 points • If the station type is intelligent device: 0 to 2048 points (in units of 16 points) • If the station type is remote I/O: 0 to 64 points (in units of 16 points) • If the station type is remote device: 0 to 128 points (in units of 16 points)	0 to 2048 0 to 64 0 to 128	User
+598		RWr/RWw offset	Sets the offset value from the start of RWr/RWw (in units of 4 points).[*2]	0 to 8188	User
+599		Number of RWr/RWw points	Sets the number of RWr/RWw points.[*2] • If the station type is master or local: 0 to 256 points (in units of 16 points) • If the station type is intelligent device: 0 to 1024 points (in units of 4 points) • If the station type is remote device: 0 to 64 points (in units of 4 points)	0 to 256 0 to 1024 0 to 64	User
+600 to +604	For 121st module	Same as above			User

*1 If a value outside the setting range is set for the appropriate station information setting, the instruction will complete with an error.

*2 The specified number of points is regarded as the number of RX/RY or RWr/RWw points.
Repeat this setting for the total number of slave stations that was set in the control data.

■Reserved station setting data (for master station setting only)

<table>
<tr><th colspan="5">Operand: (s3)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td>+0 to +7</td><td>Station 1 to 120 specification[*1]</td><td>Specify the reserved station.
• 0: Not specified
• 1: Specified
Each number in the table indicates a station number.
<table>
<tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>(s3)+0</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>(s3)+1</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td></tr>
<tr><td>(s3)+2</td><td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td></tr>
<tr><td>(s3)+3</td><td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td></tr>
<tr><td>(s3)+4</td><td>80</td><td>79</td><td>78</td><td>77</td><td>76</td><td>75</td><td>74</td><td>73</td><td>72</td><td>71</td><td>70</td><td>69</td><td>68</td><td>67</td><td>66</td><td>65</td></tr>
<tr><td>(s3)+5</td><td>96</td><td>95</td><td>94</td><td>93</td><td>92</td><td>91</td><td>90</td><td>89</td><td>88</td><td>87</td><td>86</td><td>85</td><td>84</td><td>83</td><td>82</td><td>81</td></tr>
<tr><td>(s3)+6</td><td>112</td><td>111</td><td>110</td><td>109</td><td>108</td><td>107</td><td>106</td><td>105</td><td>104</td><td>103</td><td>102</td><td>101</td><td>100</td><td>99</td><td>98</td><td>97</td></tr>
<tr><td>(s3)+7</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>120</td><td>119</td><td>118</td><td>117</td><td>116</td><td>115</td><td>114</td><td>113</td></tr>
</table>
The default parameter specifies that all stations are 0 (not specified).</td><td>Left</td><td>User</td></tr>
</table>

*1 Make the setting for the station number that was set with the appropriate setting information setting data.

Point

If the specified total number of slave stations does not match the individual station setting data, the total number of individual stations specified in the total number of slave stations take precedence. Individual station information which is specified exceeding the total number of slave stations is ignored. Note that 1 is added to the total number of slave stations when the submaster function enabled/disabled ((s1)+1 bit 3) is enabled (1).

■Error invalid station setting data (for master station setting only)

<table>
<tr><th colspan="5">Operand: (s4)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td>+0 to +7</td><td>Station 1 to 120 specification[1]</td><td>Specify the error invalid station.
• 0: Not specified
• 1: Specified
Each number in the table indicates a station number.
<table>
<tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>(s4)+0</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>(s4)+1</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td></tr>
<tr><td>(s4)+2</td><td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td></tr>
<tr><td>(s4)+3</td><td>64</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td></tr>
<tr><td>(s4)+4</td><td>80</td><td>79</td><td>78</td><td>77</td><td>76</td><td>75</td><td>74</td><td>73</td><td>72</td><td>71</td><td>70</td><td>69</td><td>68</td><td>67</td><td>66</td><td>65</td></tr>
<tr><td>(s4)+5</td><td>96</td><td>95</td><td>94</td><td>93</td><td>92</td><td>91</td><td>90</td><td>89</td><td>88</td><td>87</td><td>86</td><td>85</td><td>84</td><td>83</td><td>82</td><td>81</td></tr>
<tr><td>(s4)+6</td><td>112</td><td>111</td><td>110</td><td>109</td><td>108</td><td>107</td><td>106</td><td>105</td><td>104</td><td>103</td><td>102</td><td>101</td><td>100</td><td>99</td><td>98</td><td>97</td></tr>
<tr><td>(s4)+7</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>120</td><td>119</td><td>118</td><td>117</td><td>116</td><td>115</td><td>114</td><td>113</td></tr>
</table>
The default parameter specifies that all stations are 0 (not specified).</td><td>Left</td><td>User</td></tr>
</table>

[1] Make the setting for the station number that was set with the appropriate setting information setting data.
If both the error invalid and reserved stations are specified for the same station, the reserved station setting will take priority.

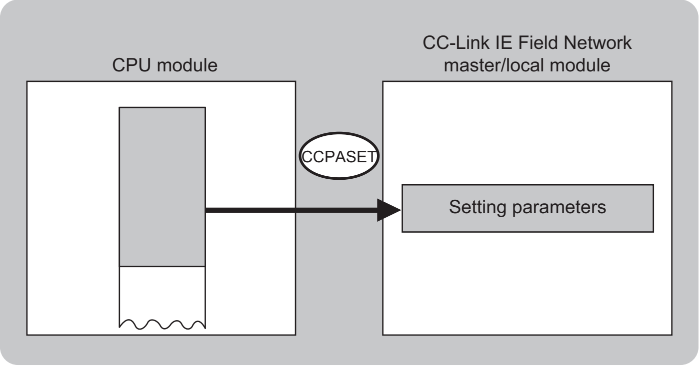
Point

Any station numbers outside the range specified by the number of slave stations or other than those specified in the individual station information settings are ignored even if they are specified as reserved stations or temporary error invalid stations.

Processing details

- This instruction sets the parameters in the CC-Link IE Field Network master and local modules.

[Own station]



- The execution status and the completion status of the G(P).CCPASET instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the G(P).CCPASET instruction completes, and turns off during the next END processing.

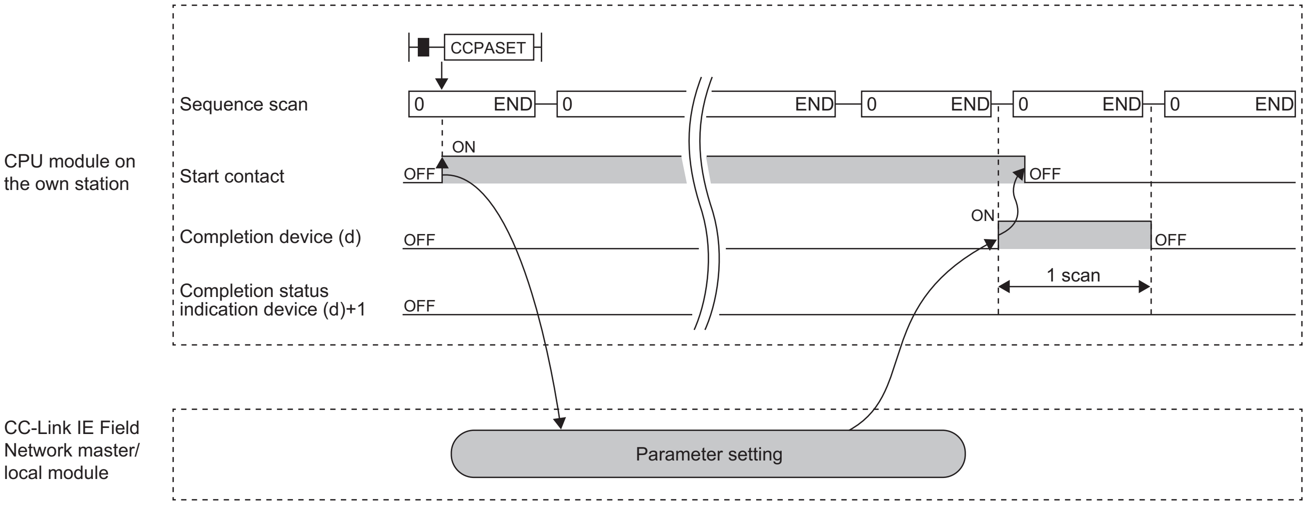
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).CCPASET instruction.

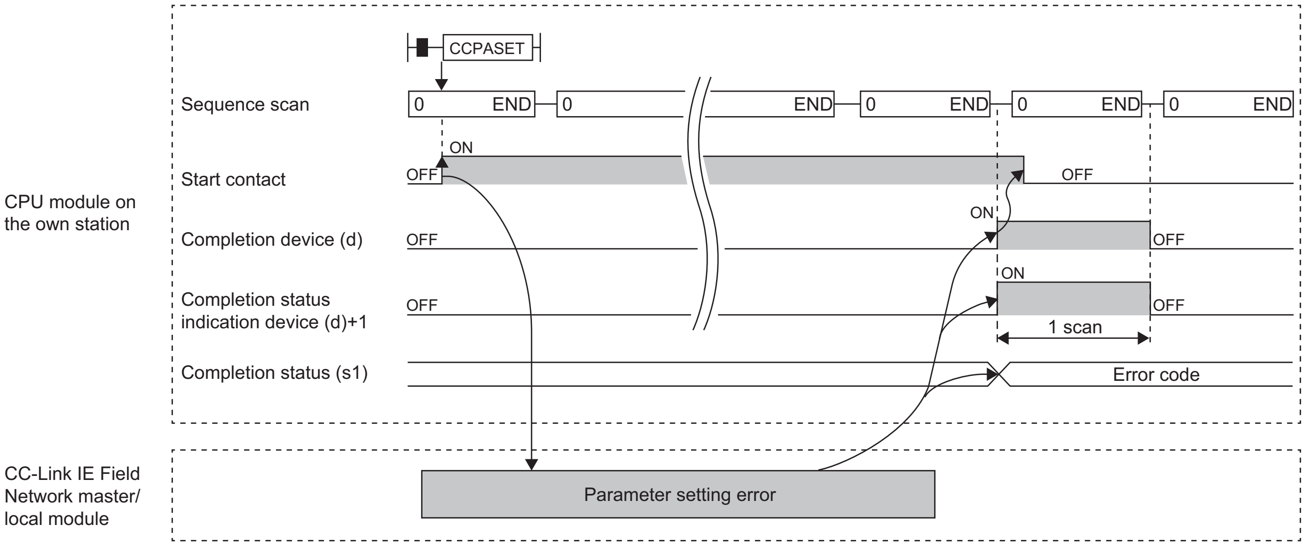
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).CCPASET instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).CCPASET instruction operates when completing its execution.
  - When completed successfully



  - When completed with an error



Operation error

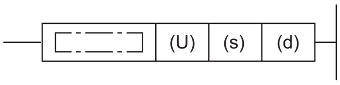
Error code ((s1)+0)	Description
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

18.8 Setting the Station Number to Own Station

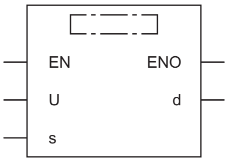
G(P).UINI, Z(P).UINI

RnCPU RnENCPU RnPCPU (Process) ✗ RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

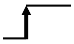
These instructions set the station number for the local, or own, station whose station number has not yet been set.

Ladder	ST
	ENO:=G_UINI(EN,U,s,d); ENO:=GP_UINI(EN,U,s,d); ENO:=Z_UINI(EN,U,s,d); ENO:=ZP_UINI(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.UINI Z.UINI	
GP.UINI ZP.UINI	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
	Z(P).UINI	Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d)		Device of the own station, which turns on for one scan upon completion of the instruction. When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others
		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	$	(U)
(U)	G(P).UINI	—	—	○	—	—	—	—	○	○	—	—	○
	Z(P).UINI	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○[2]	—	—	—	—	○	—	—	—	—
(d)		○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	—	Not used	—	System
+1	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Change target specification	0001H (fixed)	0001H	User
+3	Station number of own station	Specifies the station number to be set.	1 to 120	User
+4 to +9	—	Not used	—	System

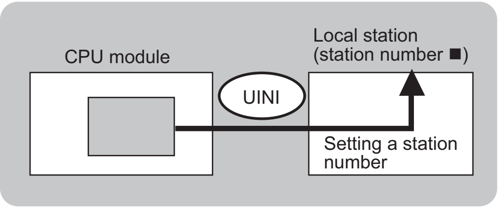
Point

- The UINI instruction can be executed only once.
- If the UINI instruction is issued after the station number has been determined, it will complete with an error.
- If the UINI instruction completes with an error before the station number is determined, take corrective action to correct the error content before retrying to execute the instruction.

Processing details

- Sets the station number for the local station.

[Own station]



- The execution status and the completion status of the UINI instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

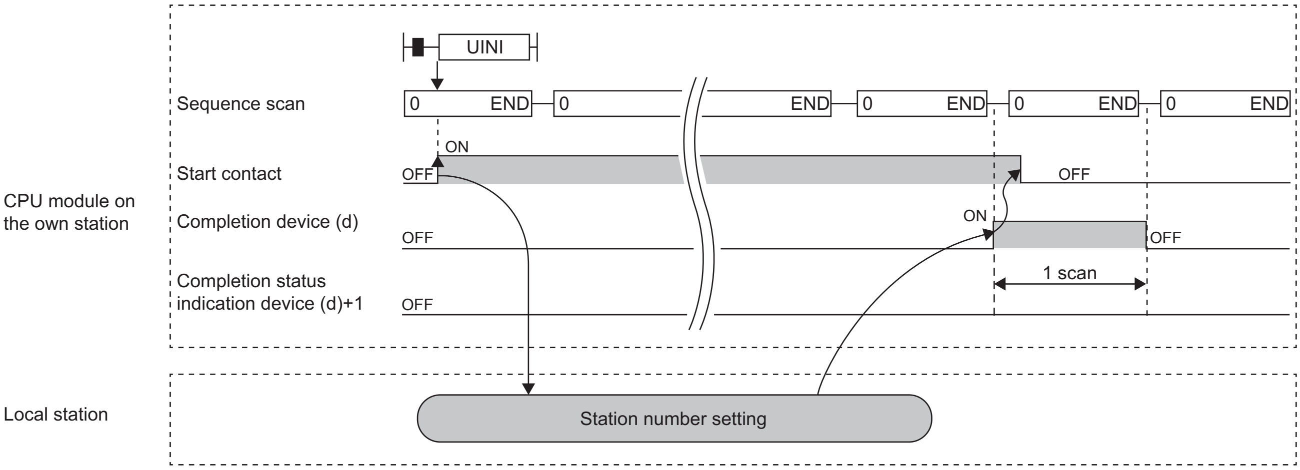
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the UINI instruction.

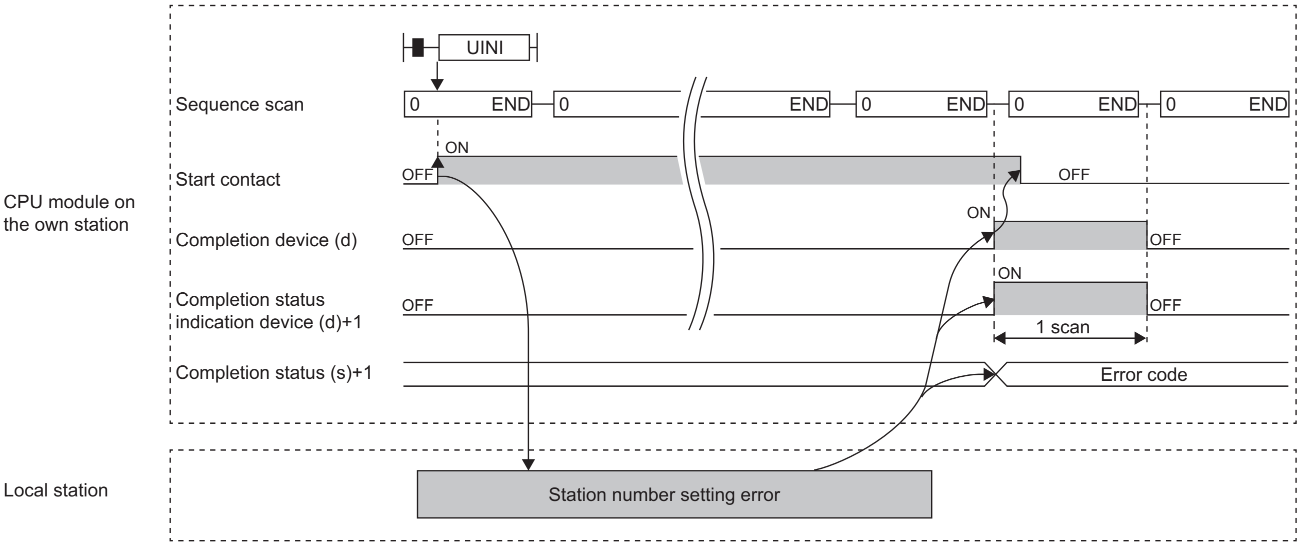
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the UINI instruction completes, and turns off during the next END processing.

- The following figure shows the operation at completion of the UINI instruction.
  - When completed successfully



  - When completed with an error



Operation error

Error code ((s)+1)	Description
D000H to DFFFH	📖 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

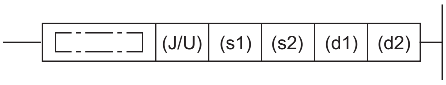
18.9 Sending an SLMP Frame

J(P).SLMPREQ, G(P).SLMPREQ

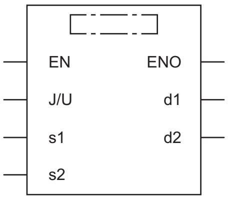
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

- The RnENCPU (network part), RJ71GF11-T2, and RJ71EN71 with firmware version "18" or later support these instructions. (Use an engineering tool with version "1.035M" or later.)

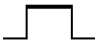
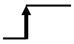
These instructions send an SLMP frame to the SLMP-compatible device in the same network. Data of the target station can be read/written and operated.

Ladder	ST
	ENO:=J_SLMPREQ(EN,J,s1,s2,d1,d2); ENO:=JP_SLMPREQ(EN,J,s1,s2,d1,d2); ENO:=G_SLMPREQ(EN,U,s1,s2,d1,d2); ENO:=GP_SLMPREQ(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
J.SLMPREQ G.SLMPREQ	
JP.SLMPREQ GP.SLMPREQ	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(J/U)	J(P).SLMPREQ	(J): Own station network number	1 to 239	16-bit unsigned binary	ANY16
	G(P).SLMPREQ	(U): Start I/O number (first three digits in four-digit hexadecimal representation) of own station or own node	00H to FEH	16-bit unsigned binary	ANY16
(s1)		Own station start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(s2)		Own station start device where request data is stored	—	Device name	ANY16[1]
(d1)		Own station start device for storing response data	—	Device name	ANY16[1]
(d2)		Device of the own station, which 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] 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

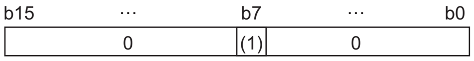
Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	J	U
(J/U)	J(P).SLMPREQ	—	—	—	—	—	—	—	—	—	—	—	○	—
	G(P).SLMPREQ	—	—	○	—	—	—	—	○	○	—	—	—	○
(s1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○*2	—	—	—	—	○	—	—	—	—	—
(d2)		○*1	—	○*3	—	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Error completion type	 (1)Error completion type (bit 7) Specify whether to set clock data when completed with an error. • 0: Clock data at error occurrence is not stored in (s1)+12 and later. • 1: Clock data at error occurrence is stored in (s1)+12 and later.	0000H 0080H	User
+1	Completion status	The instruction completion status is stored. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Not used	—	—	—
+3	Request destination module I/O number	Specify an access target module. • 03FFH: Fixed	03FFH	User
+4	Request destination network number	Specify an access target network number. • 1 to 239: Network number	1 to 239	User
+5	Request destination station number	Specify an access target station number. • 1 to 120: Local station, intelligent device station, remote device station, submaster station • 125: Master station • 126: Master operating station	1 to 120, 125, 126	User
+6	Not used	—	—	—
+7	Number of resends	■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+8. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+8	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent by the number of resends specified in (s1)+7. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
+9	Request data length	Specify the data size of the request data to be specified by (s2) in bytes. The request data includes (s2)+0 (not used).	6 to 1964	User
+10	Response data length	The data size of the response data to be stored in (d1) is stored in bytes. The response data includes (d1)+0 (end code).	—	System

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+11	Clock setting flag	The validity status (valid or invalid) of the data in (s1)+12 and later is stored. Note that the data in (s1)+12 and later is not cleared even when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+12	Clock data (Set only in an abnormal state)	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+13		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)	—	System
+14		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)	—	System
+15		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))	—	System
+16	Error detection network number	The network number of the station in which an error was detected is stored. • 1 to 239 (Network number)	—	System
+17	Error-detected station number	The number of the station where an error was detected is stored. • 1 to 120: Local station, intelligent device station, remote device station, submaster station • 125: Master station • 126: Master operating station	—	System

■Request data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Not used	—	—	—
+1	Command	Sets the command of the SLMP frame.*1	Refer to the manual of the target station.	User
+2	Subcommand	Sets the subcommand of the SLMP frame.*1		User
+3 to +□	SLMP frame data	Stores the SLMP frame data after the subcommand.*1		User

*1 For details on the SLMP frame, refer to the following.
📖 SLMP Reference Manual

■Response data

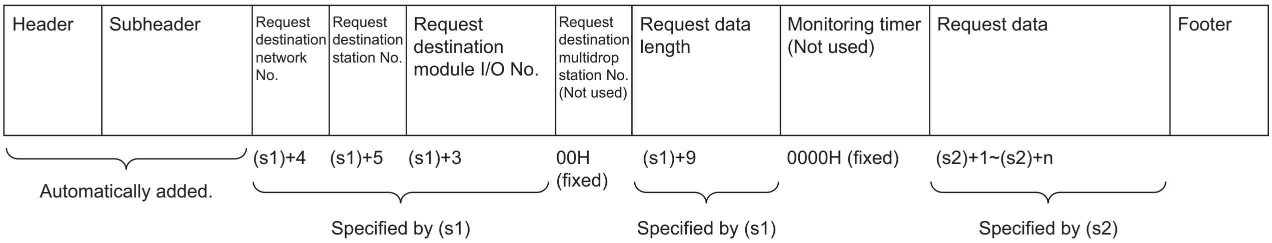
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	End code	Stores the result of command processing. In normal end, 0 is stored. In abnormal end, an error code set by the external device is stored.	—	System
+1 to +□	Response data	Sets execution results for the request data. (Some commands do not return response data.)	—	System

Processing details

- These instructions send the request data in the device specified by (s2) or later to the target station specified with the control data. When a response message is received from the target station, it is stored in the device specified by (d1).

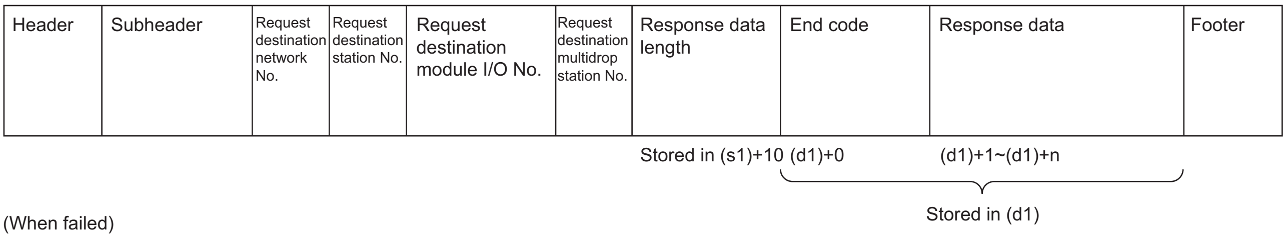
The following figures show the request data and the response data in normal/abnormal end.

●Request data

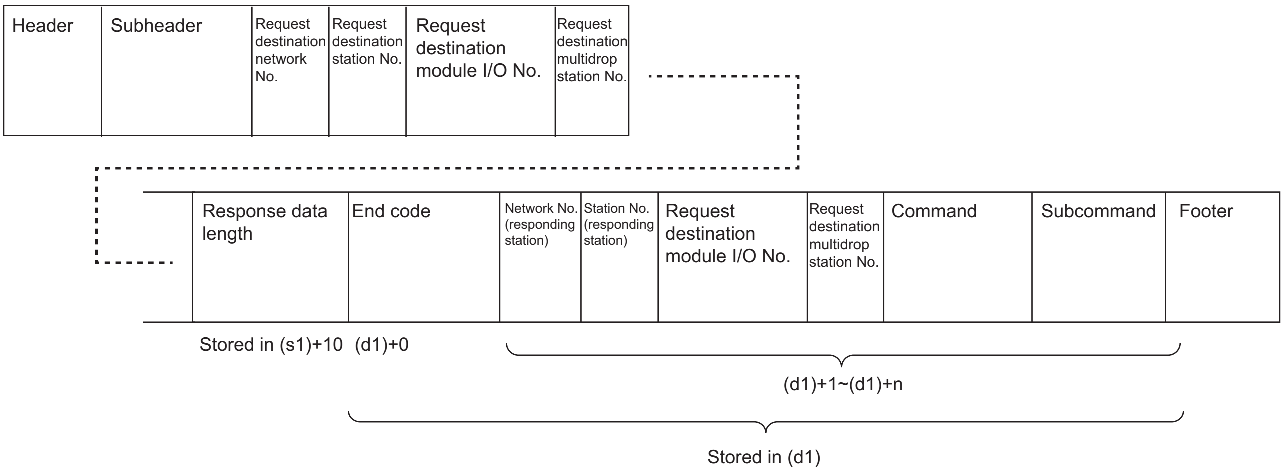


●Response data

(When completed)



(When failed)

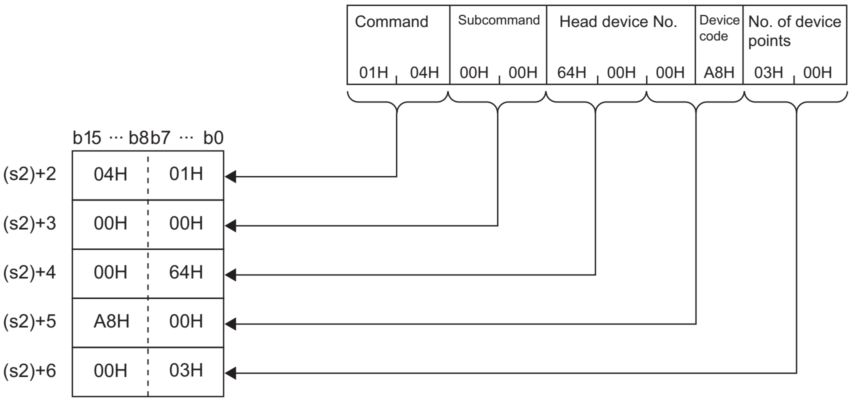


- The J(P).SLMPREQ and G(P).SLMPREQ instructions communicate in binary code.
- A serial number is automatically given to the subheader.

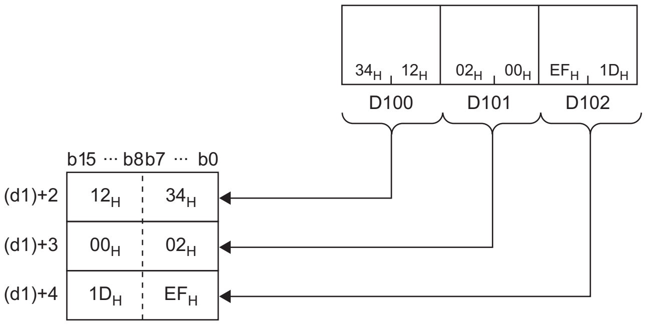
Ex.

Sending "Read (command: 0401H)" (read in units of words) which reads the value in D100 to D102

●Request data



●Response data



Operation error

Error code (SW0080 to SW009F)	Description
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
D000H to DFFFH	🕮 MELSEC iQ-R CC-Link IE Field Network User's Manual (Application)

Precautions

Stations in other networks cannot be set as target stations. (Access via a relay station is not allowed.)

19 CC-LINK INSTRUCTIONS

Point

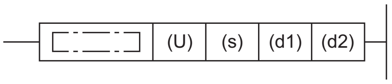
This chapter describes the instructions used commonly by MELSEC iQ-R series modules. For the instructions when MELSEC-Q series modules are used, refer to the manuals for each module used and create programs.
For precautions when using modules, refer to the following.
📖 MELSEC iQ-R Module Configuration Manual

19.1 Reading Data from the Target Station

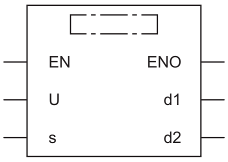
G(P).RIRD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

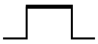
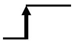
These instructions read data of the specified number of points from the buffer memory area of the target station or the device of its CPU module.

Ladder	ST
	ENO:=G_RIRD(EN,U,s,d1,d2); ENO:=GP_RIRD(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RIRD	
GP.RIRD	

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
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d1)	Start device for storing the read data	—	Device name	ANY16[*1]
(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

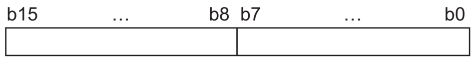
[*1] When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

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	$	
(U)[*1]	—	—	○	—	—	—	—	○	○	—	—	○
(s)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(d1)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(d2)	○[*2]	—	○[*4]	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.
*4 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	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
+1	Station number	Specify the station number of the target station.	0 to 64	User
+2	Access/attribute code	Specifies the type of the buffer memory or device from which to read data.  b15 … b8 b7 … b0 • b8 to b15: Access code • b0 to b7: Attribute code	Refer to "Access/attribute code."	User
+3	Buffer memory address or device number	Specifies the start address of the buffer memory or the start device from which to read data.	Refer to the manual of the target station.[*1]	User
+4	Number of read points	Specifies, in units of words, the number of points to be read.	1 to 480[*2] 1 to 32[*3]	User

*1 When specifying a random accuracy buffer, specify the address with the start of the random accuracy buffer defined as 0.
*2 Specify a value within the capacity of the target station buffer memory area or device.
*3 If data is to be read from a CPU module device if the target station CPU module is other than RCPU, QCPU (Q mode), QCPU (A mode), LCPU, QnACPU, or AnUCPU, the setting range is 1 to 32 words.

■Access/attribute code

- When data is read from the buffer memory within the CC-Link module

Contents of buffer memory		Access code	Attribute code
Buffer in intelligent device		00H	04H
Buffer in master, local, or standby master station	Random access buffer	20H	04H
	Remote input	21H	04H
	Remote output	22H	04H
	Remote register	24H	04H
	Link special relay	63H	04H
	Link special register	64H	04H

- When data is read from a CPU module device

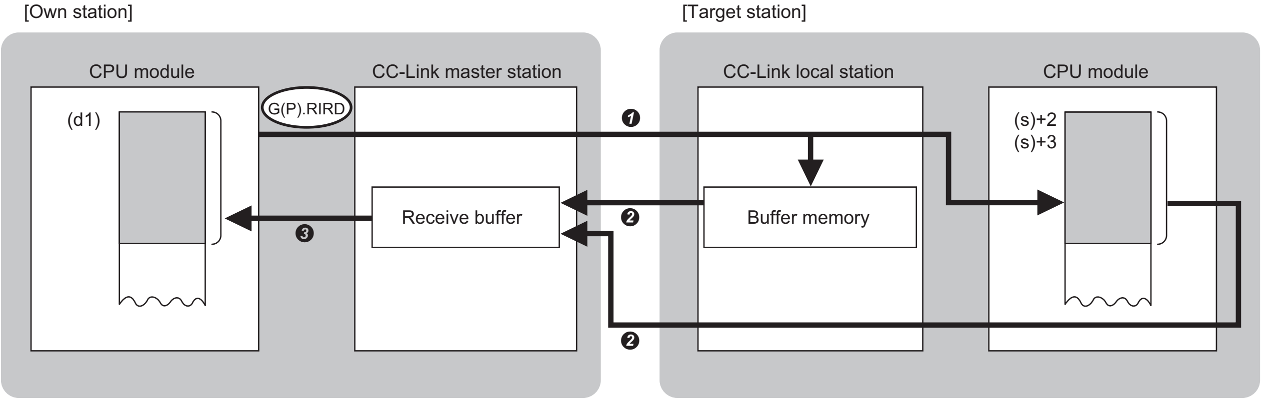
Device category*1	Name	Date type		Unit	Access code	Attribute code
		Bit	Word			
Input relay	X	○	—	Hexadecimal	01H	05H
Output relay	Y	○	—	Hexadecimal	02H	05H
Internal relay	M	○	—	Decimal	03H	05H
Latch relay	L	○	—	Decimal	83H	05H
Link relay	B	○	—	Hexadecimal	23H	05H
Timer (contact)	T	○	—	Decimal	09H	05H
Timer (coil)	T	○	—	Decimal	0AH	05H
Timer (current value)	T	—	○	Decimal	0CH	05H
Retentive timer (contact)	ST	○	—	Decimal	89H	05H
Retentive timer (coil)	ST	○	—	Decimal	8AH	05H
Retentive timer (current value)	ST	—	○	Decimal	8CH	05H
Counter (contact)	C	○	—	Decimal	11H	05H
Counter (coil)	C	○	—	Decimal	12H	05H
Counter (current value)	C	—	○	Decimal	14H	05H
Data register*2	D	—	○	Decimal	04H	05H
Link register*2	W	—	○	Hexadecimal	24H	05H
File register	R	—	○	Decimal	84H	05H
Link special relay	SB	○	—	Hexadecimal	63H	05H
Link special register	SW	—	○	Hexadecimal	64H	05H
Special relay	SM	○	—	Decimal	43H	05H
Special register	SD	—	○	Decimal	44H	05H

*1 Only the above devices can be specified. To read data from a bit device, specify it with 0 or a multiple of 16.

*2 None of D65536 and the subsequent extended data registers and of W10000 and the subsequent extended link registers are accepted.

Processing details

- This instruction reads data of the specified number of points from the buffer memory area of the target station or the device of its CPU module.
- From the master station, the instruction can be executed for the local, standby master, or intelligent device station. From the local or standby master station, the instruction can be executed for the master, local, or standby master station.
- The following figure shows how the G(P).RIRD instruction operates during execution.



➊ Data is read from the buffer memory area specified by (s)+2 and (s)+3 that is included in the station specified by (s)+1 or from the device of the CPU module.
➋ The read data is stored in the receive buffer of the master station.
➌ The read data is stored in the device specified by (d1) and later, and the device specified by (d2) is turned on.

- The completion status of the G(P).RIRD instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the G(P).RIRD instruction completes, and turns off during the next END processing.

  - Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the G(P).RIRD instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).RIRD instruction completes, and turns off during the next END processing.

Precautions

- The monitoring time and the number of retries can be set with the following link special registers (SWs).
  - Monitoring time setting (SW0009)
  - Setting of the number of retries for dedicated instruction (SW000B)
- The G(P).RIRD instruction can be concurrently executed from the master station for two or more local, standby master, or intelligent device stations. Two or more dedicated instructions cannot be concurrently executed for a single station. Two or more dedicated instructions, including those other than G(P).RIRD, cannot be concurrently executed from a local station even for another station. If the next dedicated instruction is issued before completion of the preceding one that has started, the next one will be ignored. Create the program so that the next dedicated instruction will start after the completion device turns on, because processing of a dedicated instruction takes several scans until its completion.

Operation error

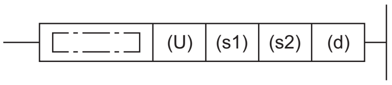
Error code ((s)+0)	**Description**
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
B000H to BFFFH	🕮 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

19.2 Writing Data to the Target Station

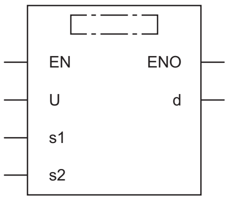
G(P).RIWT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

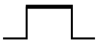
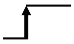
These instructions write data of the specified number of points to the buffer memory area of the target station or the device of its CPU module.

Ladder	ST
	ENO:=G_RIWT(EN,U,s1,s2,d); ENO:=GP_RIWT(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RIWT	
GP.RIWT	

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	Refer to the control data.	Device name	ANY16[1]
(s2)	Start device for storing the data to be written	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)*1	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○*3	—	—	—	—	○	—	—	—	—
(s2)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d)	○*2	—	○*4	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.

*2 FX and FY cannot be used.

*3 FD cannot be used.

*4 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	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
+1	Station number	Specify the station number of the target station.	0 to 64	User
+2	Access/attribute code	Specifies the type of the buffer memory or device from which to read data. b15 … b8 b7 … b0 • b8 to b15: Access code • b0 to b7: Attribute code	Refer to "Access/attribute code."	User
+3	Buffer memory address or device number	Specifies the start address of the buffer memory area or the start device to which to write data.	Refer to the manual of the target station.*1	User
+4	Number of write points	Specifies, in units of words, the number of points to be written.	1 to 480*2 1 to 32*3	User

*1 When specifying a random accuracy buffer, specify the address with the start of the random accuracy buffer defined as 0.

*2 Specify a value within the capacity of the target station buffer memory area or device.

*3 If data is to be written to a CPU module device if the target station CPU module is other than RCPU, QCPU (Q mode), QCPU (A mode), LCPU, QnACPU, or AnUCPU, the setting range is 1 to 32 words.

■Access/attribute code

- When data is written to the buffer memory within the CC-Link module

Contents of buffer memory		Access code	Attribute code
Buffer in intelligent device		00H	04H
Buffer in master or local station	Random access buffer	20H	04H
	Remote input	21H	04H
	Remote output	22H	04H
	Remote register	24H	04H
	Link special relay	63H	04H
	Link special register	64H	04H

- When data is written to the device of the CPU module

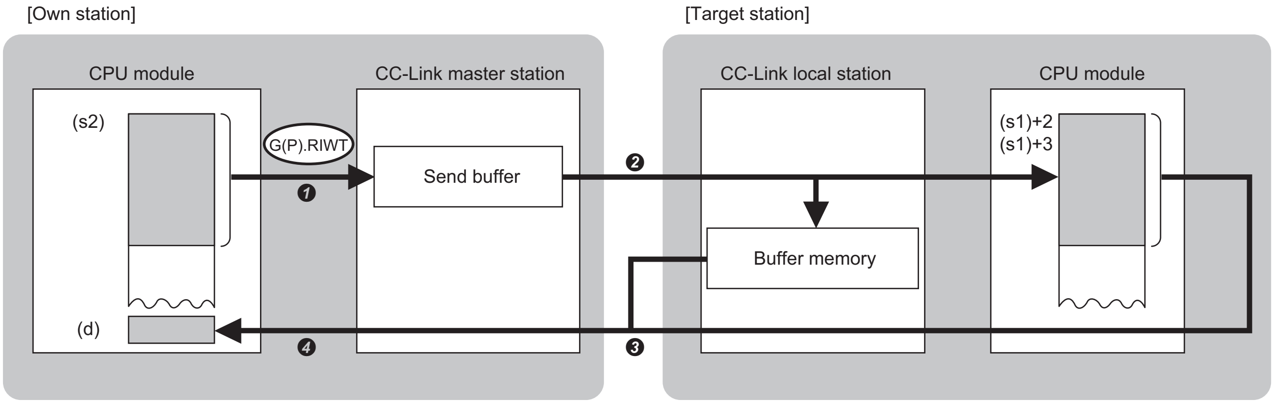
Device category[*1]	Name	Date type		Unit	Access code	Attribute code
		Bit	Word			
Input relay	X	○	—	Hexadecimal	01H	05H
Output relay	Y	○	—	Hexadecimal	02H	05H
Internal relay	M	○	—	Decimal	03H	05H
Latch relay	L	○	—	Decimal	83H	05H
Link relay	B	○	—	Hexadecimal	23H	05H
Timer (contact)	T	○	—	Decimal	09H	05H
Timer (coil)	T	○	—	Decimal	0AH	05H
Timer (current value)	T	—	○	Decimal	0CH	05H
Retentive timer (contact)	ST	○	—	Decimal	89H	05H
Retentive timer (coil)	ST	○	—	Decimal	8AH	05H
Retentive timer (current value)	ST	—	○	Decimal	8CH	05H
Counter (contact)	C	○	—	Decimal	11H	05H
Counter (coil)	C	○	—	Decimal	12H	05H
Counter (current value)	C	—	○	Decimal	14H	05H
Data register[*2]	D	—	○	Decimal	04H	05H
Link register[*2]	W	—	○	Hexadecimal	24H	05H
File register	R	—	○	Decimal	84H	05H
Link special relay	SB	○	—	Hexadecimal	63H	05H
Link special register	SW	—	○	Hexadecimal	64H	05H
Special relay	SM	○	—	Decimal	43H	05H
Special register	SD	—	○	Decimal	44H	05H

[*1] Only the above devices can be specified. To write data to a bit device, specify it with 0 or a multiple of 16.

[*2] None of D65536 and the subsequent extended data registers and of W10000 and the subsequent extended link registers are accepted.

Processing details

- This instruction writes data of the specified number of points from the buffer memory area of the target station or the device of its CPU module.
- From the master station, the instruction can be executed for the local, standby master, or intelligent device station. From the local or standby master station, the instruction can be executed for the master, local, or standby master station.
- The following figure shows how the G(P).RIWT instruction operates during execution.



❶ The data specified by (s2) is stored in the send buffer of the master station.
❷ Data is stored in the buffer memory area specified by (s1)+2 and (s1)+3 that is included in the station specified by (s1)+1 or to the device of the CPU module.
❸ The response indicating the write completion returns from the target station to the master station.
❹ The device specified by (d) is turned on.

- The completion status of the G(P).RIWT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the G(P).RIWT instruction completes, and turns off during the next END processing.

  - Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).RIWT instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the G(P).RIWT instruction completes, and turns off during the next END processing.

Precautions

- The monitoring time and the number of retries can be set with the following link special registers (SWs).
  - Monitoring time setting (SW0009)
  - Setting of the number of retries for dedicated instruction (SW000B)
- The G(P).RIWT instruction can be concurrently executed from the master station for two or more local or intelligent device stations. Two or more dedicated instructions cannot be concurrently executed for a single station. Two or more dedicated instructions, including those other than G(P).RIWT, cannot be concurrently executed from a local station even for another station. If the next dedicated instruction is issued before completion of the preceding one that has started, the next one will be ignored. Create the program so that the next dedicated instruction will start after the completion device turns on, because processing of a dedicated instruction takes several scans until its completion.

Operation error

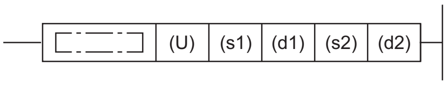
Error code ((s1)+0)	**Description**
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
B000H to BFFFH	🕮 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

19.3 Reading Data from the Buffer Memory of the Specified Intelligent Device Station

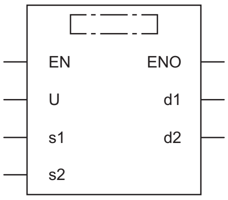
G(P).RIRCV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

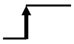
These instructions automatically perform a handshake with the specified intelligent device station and read data from its buffer memory. The instructions can be used for communications with a module supporting a handshake signal such as AJ65BT-R2N.

Ladder	ST
	ENO:=G_RIRCV(EN,U,s1,s2,d1,d2); ENO:=GP_RIRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RIRCV	
GP.RIRCV	

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	Refer to the control data.	Device name	ANY16[*1]
(d1)	Start device for storing the read data	—	Device name	ANY16[*1]
(s2)	Start device for storing the interlock signal	—	Device name	ANY16[*1]
(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 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

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	$	
(U)*1	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d1)	—	—	○*3	—	—	—	—	○	—	—	—	—
(s2)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d2)	○*2	—	○*4	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.
*4 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	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
+1	Station number	Specify the station number of the target station.	1 to 64	User
+2	Access/attribute code	Specifies 0004H.	0004H	User
+3	Buffer memory address	Specifies the start address of the buffer memory area from which to read data.	Refer to the manual of the target station.	User
+4	Number of read points	Specifies, in units of words, the number of points to be read.	1 to 480*1	User

*1 Specify a value within the capacity of the target station buffer memory area.

■Interlock signal

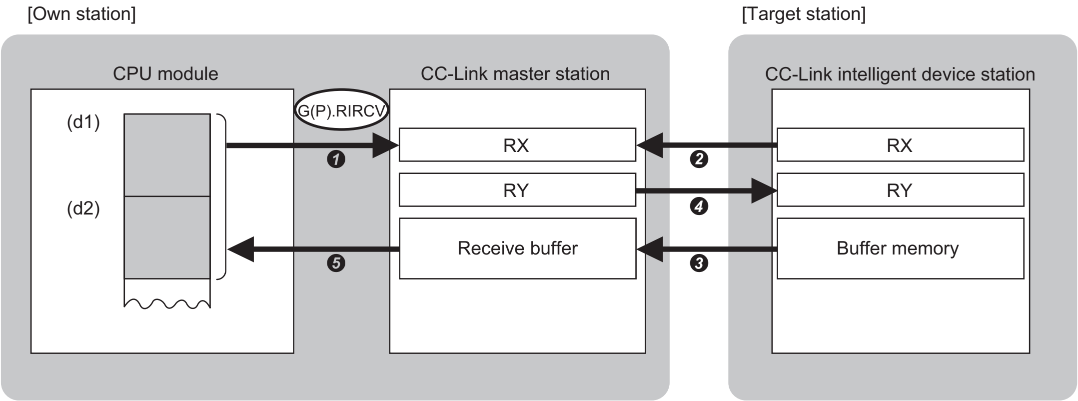
Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	b15 … b8 b7 … b0 0 | RY	RY: Specifies the number of the completion signal (RY) to be used for interlock.*1	00H to 7FH	User
+1	b15 … b8 b7 … b0 0 | RX	RY: Specifies the number of the read request signal (RY) to be used for interlock.*1	00H to 7FH	User
+2	Dummy	Specifies 0.	0	User

*1 For the interlock signal, refer to the following.
📖 Manual of the intelligent device station from which to read data

Processing details

- This instruction automatically performs handshake with the specified intelligent device station and reads data from its buffer memory.
- The instruction can be executed, from the master station, for an intelligent device station with a handshake signal (e.g. AJ65BT-R2N).
- The following figure shows how the G(P).RIRCV instruction operates during execution.



❶ A request is issued to read data from the buffer memory address specified by (s1)+3 that is included in the station specified by (s1)+1.
❷ The remote input (RX) specified by (s2)+1 that is in the station specified by (s1)+1 is monitored.
❸ Turning on the remote input (RX) causes the master station to read the data from the buffer memory of the target station. The read data is stored in the receive buffer of the master station.
❹ The master station turns on the remote output (RY) specified by (s2)+0. It turns off the remote output (RY) specified by (s2)+0 by turning on and off the remote output (RX) specified by (s2)+1.
❺ The data read from the target station is stored in the device specified by (d1) and later, and the device specified by (d2) is turned on.

- The completion status of the G(P).RIRCV instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the G(P).RIRCV instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the G(P).RIRCV instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).RIRCV instruction completes, and turns off during the next END processing.

Precautions

- The monitoring time and the number of retries can be set with the following link special registers (SWs).
  - Monitoring time setting (SW0009)
  - Setting of the number of retries for dedicated instruction (SW000B)
- The G(P).RIRCV instruction can be concurrently executed for two or more intelligent device stations. Two or more dedicated instructions, including those other than G(P).RIRD, cannot be concurrently executed for a single station. If the next dedicated instruction is issued before completion of the preceding one that has started, the next one will be ignored. Create the program so that the next dedicated instruction will start after the completion device turns on, because processing of a dedicated instruction takes several scans until its completion.

Operation error

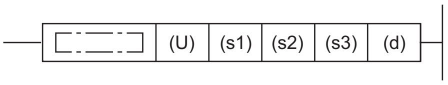
Error code ((s1)+0)	**Description**
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)
B000H to BFFFH	📖 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

19.4 Writing Data to the Buffer Memory of the Specified Intelligent Device Station

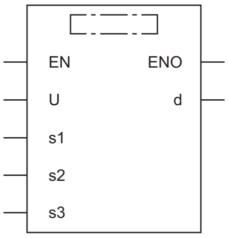
G(P).RISEND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

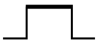
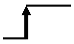
These instructions automatically perform handshake with the specified intelligent device station and writes data to its buffer memory. The instructions can be used for communications with a module supporting a handshake signal such as AJ65BT-R2N.

Ladder	ST
	ENO:=G_RISEND(EN,U,s1,s2,s3,d); ENO:=GP_RISEND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RISEND	
GP.RISEND	

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	Refer to the control data.	Device name	ANY16[*1]
(s2)	Start device for storing the data to be written	—	Device name	ANY16[*1]
(s3)	Start device for storing the interlock signal	—	Device name	ANY16[*1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[1]	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s3)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[4]	—	—	—	—	—	—	—	—	—

[1] Index modification is not available.

[2] FX and FY cannot be used.

[3] FD cannot be used.

[4] T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	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
+1	Station number	Specify the station number of the target station.	1 to 64	User
+2	Access/attribute code	Specifies 0004H.	0004H	User
+3	Buffer memory address	Specifies the start address of the buffer memory area to which to write data.	Refer to the manual of the target station.	User
+4	Number of write points	Specifies, in units of words, the number of points to be written.	1 to 480[1]	User

[1] Specify a value within the capacity of the target station buffer memory area.

■Interlock signal

Operand: (s3)

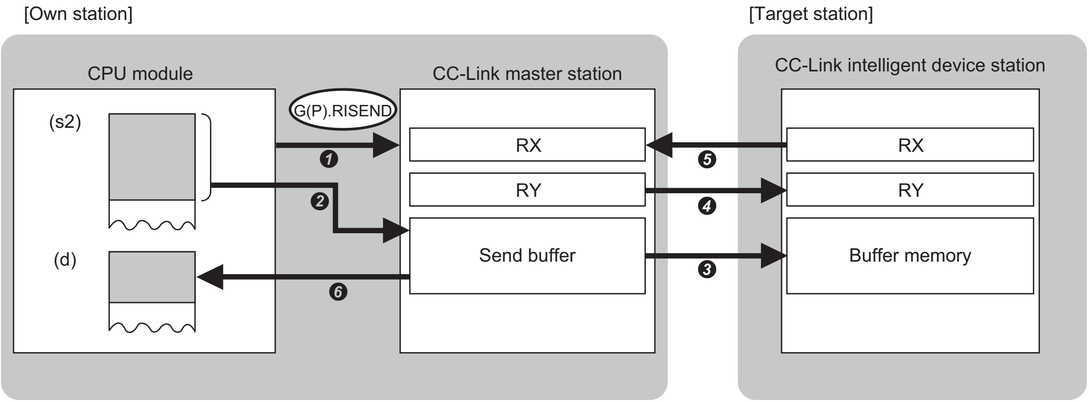
Device	Item	Description	Setting range	Set by
+0	b15 … b8 b7 … b0 0 RY	RY: Specifies the number of the write request signal (RY) to be used for interlock.[1]	00H to 7FH	User
+1	b15 … b8 b7 … b0 RWr[1] RX	RY: Specifies the number of the write completion signal (RY) to be used for interlock.[1]	00H to 7FH	User
		RWr: Specifies the number of the device in which to store the error code (RWr). If the error code storage device does not exist, specify FFH.	00H to 0FH, FFH[2]	User
+2	Completion mode	Specifies how the completion signal behaves. • 0: Using Device 1 During completion, the RX signal specified by (s2)+1 turns on. • 1: Using Device 2 For normal completion, the RX signal specified by (s2)+1 turns on. For error completion, both RX and RX+1 turn on simultaneously.	0, 1	User

[1] For the interlock signal, refer to the following.
📖 Manual of the intelligent device station from which to read data

[2] The error code stored in the error code storage device is the same as the completion status of control data ((s1)+0).

Processing details

- This instruction automatically performs handshake with the specified intelligent device station and writes data to its buffer memory.
- The instruction can be executed, from the master station, for an intelligent device station with a handshake signal (e.g. AJ65BT-R2N).
- The following figure shows how the G(P).RISEND instruction operates during execution.



❶ A request is issued to write data to the buffer memory address specified by (s1)+3 that is included in the station specified by (s1)+1.
❷ The data from the device specified by (s2) and later is stored in the send buffer of the master station.
❸ The master station writes the data of the send buffer in the buffer memory of the target station.
❹ The master station turns on the remote output (RY) specified by (s3)+0.
❺ Upon completion of processing against the remote output (RY), the target station turns on the remote input (RX) specified by (s3)+1. It turns off the remote input (RX) specified by (s3)+1 by turning on and off the remote output (RY) specified by (s3)+1.
❻ The device specified by (d) is turned on.

- The completion status of the G(P).RISEND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the G(P).RISEND instruction completes, and turns off during the next END processing.

- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).RISEND instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the G(P).RISEND instruction completes, and turns off during the next END processing.

Precautions

- The monitoring time and the number of retries can be set with the following link special registers (SWs).
  - Monitoring time setting (SW0009)
  - Setting of the number of retries for dedicated instruction (SW000B)
- The G(P).RISEND instruction can be concurrently executed for two or more intelligent device stations. Two or more dedicated instructions, including those other than G(P).RIRD, cannot be concurrently executed for a single station. If the next dedicated instruction is issued before completion of the preceding one that has started, the next one will be ignored. Create the program so that the next dedicated instruction will start after the completion device turns on, because processing of a dedicated instruction takes several scans until its completion.

Operation error

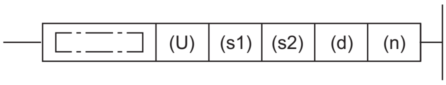
Error code ((s1)+0)	Description
4000H to 4FFFH	🕮 MELSEC iQ-R CPU Module User's Manual (Application)
B000H to BFFFH	🕮 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

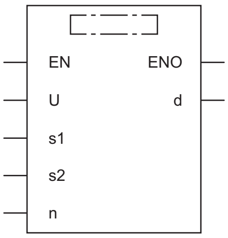
19.5 Reading Data from the Automatic Update Buffer

G(P).RIFR

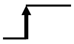
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions read data from the automatic update buffer or random access buffer. The instructions can be used for communications with a module having the automatic update buffer such as AJ65BT-R2N.

Ladder	ST
	ENO:=G_RIFR(,ENU,s1,s2,n,d); ENO:=GP_RIFR(EN,U,s1,s2,n,d);

FBD/LD


■Execution condition

Instruction	Execution condition
G.RIFR	
GP.RIFR	

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)	If data is read from an automatic update buffer: Station number of intelligent device station	1 to 64	16-bit unsigned binary	ANY16
	If data is read from a random access buffer: Specifies the random access buffer.	00FFH	16-bit unsigned binary	ANY16
(s2)	Offset value from the start of the automatic update or random access buffer that has been assigned to the target station	0 to parameter set value[1]	16-bit unsigned binary	ANY16
(d)	Start device for storing the read data	—	Device name	ANY16[2]
(n)	Number of read points	1 to 4096	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Value that was set in the CC-Link configuration window of the engineering tool.

[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.

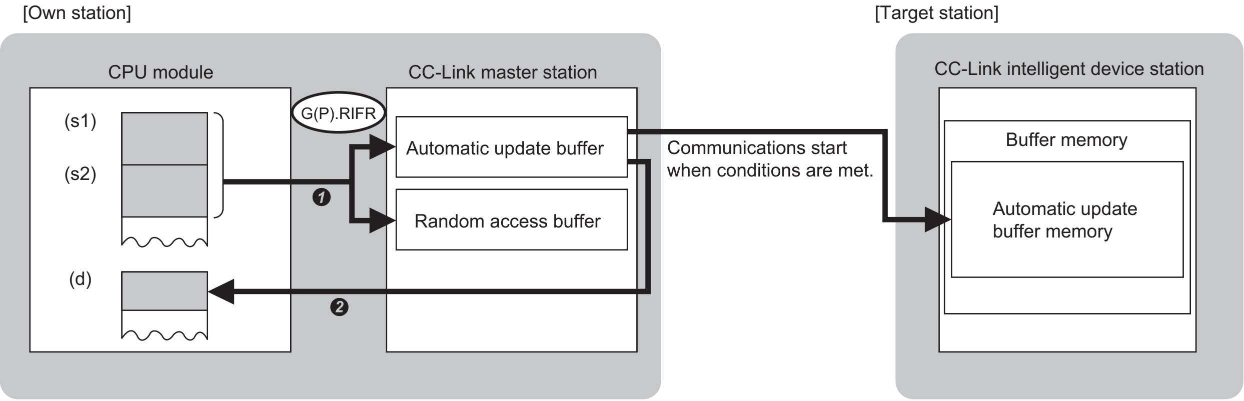
■Applicable devices

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	$	
(U)*1	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(s2)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(d)	—	—	○*3	—	—	—	—	○	—	—	—	—
(n)	○*2	—	○*3	—	—	—	—	○	○	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.

Processing details

- Data is read from the automatic update or random access buffer of the master station.
- The instruction can be executed, from the master station, for an intelligent device station that performs communication with an automatic update buffer (e.g. AJ65BT-R2N).
- The following figure shows how the G(P).RIFR instruction operates during execution.



❶ Data is read from the automatic update or random access buffer specified by (s1) and (s2) that is in the master station.
❷ The read data is stored in the device specified by (d) and later.

Precautions

- Assignment of the automatic update buffer can be achieved by CC-Link configuration setting of the module parameters.

Operation error

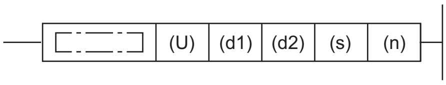
Error code (SD0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)

19.6 Writing Data to the Automatic Update Buffer

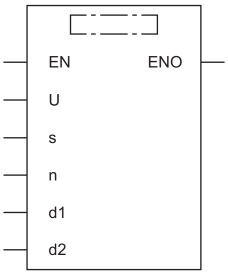
G(P).RITO

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

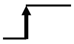
These instructions write data to the automatic update buffer or random access buffer. The instructions can be used for communications with a module having the automatic update buffer such as AJ65BT-R2N.

Ladder	ST
	ENO:=G_RITO(EN,U,s,n,d1,d2); ENO:=GP_RITO(EN,U,s,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RITO	
GP.RITO	

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
(d1)	If data is written to an automatic update buffer: Station number of target station	1 to 64	16-bit unsigned binary	ANY16
	If data is written to a random access buffer: Specifies the random access buffer.	00FFH	16-bit unsigned binary	ANY16
(d2)	Offset value from the start of the automatic update or random access buffer that has been assigned to the target station	0 to parameter set value[1]	16-bit unsigned binary	ANY16
(s)	Start device for storing the data to be written	—	Device name	ANY16[2]
(n)	Number of write points	1 to 4096	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Value that was set in the CC-Link configuration window of the engineering tool.

[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.

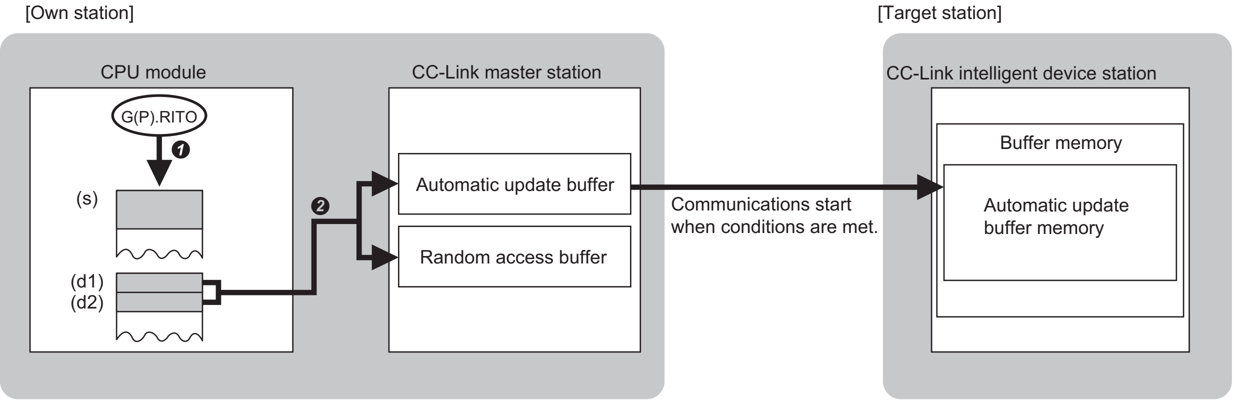
■Applicable devices

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	$	
(U)*1	—	—	○	—	—	—	—	○	○	—	—	○
(d1)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(d2)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(s)	—	—	○*3	—	—	—	—	○	—	—	—	—
(n)	○*2	—	○*3	—	—	—	—	○	○	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.

Processing details

- Data is written to the automatic update or random access buffer of the master station.
- The instruction can be executed, from the master station, for an intelligent device station that performs communication with an automatic update buffer (e.g. AJ65BT-R2N).
- The following figure shows how the G(P).RITO instruction operates during execution.



❶ Data is read from the device specified by (s) and later of the master station.
❷ The read data is written to the automatic update or random access buffer specified by (d1) and (d2).

Precautions

- Assignment of the automatic update buffer can be achieved by CC-Link configuration setting of the module parameters.

Operation error

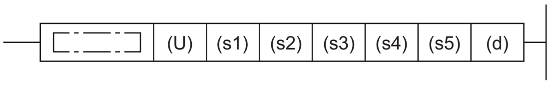
Error code (SD0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)

19.7 Setting Network Parameters

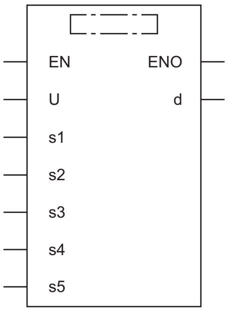
G(P).RLPASET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

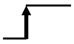
These instructions set network parameters in the master station and starts up the data link.

Ladder	ST
	ENO:=G_RLPASET(EN,U,s1,s2,s3,s4,s5,d); ENO:=GP_RLPASET(EN,U,s1,s2,s3,s4,s5,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RLPASET	
GP.RLPASET	

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	Refer to the control data.	Device name	ANY16[1]
(s2)[2]	Start device containing the slave station setting data	—	Device name	ANY16[1]
(s3)[2]	Start device containing the reserved station specification data	—	Device name	ANY16[1]
(s4)[2]	Start device containing the error invalid station specification data	—	Device name	ANY16[1]
(s5)[2]	Start device containing the automatic update buffer assignment data	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

*2 If this operand is omitted, specify the dummy device or label.

■Applicable devices

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	$	
(U)[1]	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s3)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s4)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(s5)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[4]	—	—	—	—	—	—	—	—	—

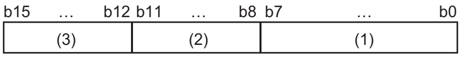
*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.
*4 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	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
+1	Setting flag	Specifies whether the setting data in (s2) to (s5) are valid/invalid • 0: Invalid[1] • 1: Valid b15 b14 b13 b12 b11 … b4 b3 b2 b1 b0 0 0 (6) (5) 0 0 (4) (3) (2) (1) (1) Slave station setting data (s2) (2) Reserved station specification data (s3) (3) Error invalid station specification data (s4) (4) Automatic update buffer assignment data (s5) (5) Data link faulty station setting • 0: Data in the remote input (RX) are cleared. • 1: Data in the remote input (RX) are held. (6) CPU STOP time output setting • 0: Data are refreshed to the remote output (RY). • 1: Data, 0, is sent to the remote output (RY).	—	User
+2	Total number of connected modules/ stations	Specifies the number of slave stations to be connected.	1 to 64	User
+3	Number of retries	Specifies the number of retries to be performed for a communication error station.	1 to 7	User
+4	Number of automatic return modules	Specifies the number of slave stations that can return by one link scan.	1 to 10	User
+5	Data link setting when CPU is down	Specifies whether to stop or continue the data link when the CPU module causes a stop error. • 0: Stops the data link. • 1: Continues the data link.	0, 1	User
+6	Scan mode setting	Specifies whether the link scan be unsynchronized or synchronized with the sequence scan. If it is unsynchronized, the output transmission delay time will shorten. • 0: Unsynchronized with sequence scan • 1: Synchronized with sequence scan	0, 1	User

*1 The default value will be used for any setting data specified as invalid. For the default values, refer to the following.
☞ Page 1663 Slave station setting data
☞ Page 1663 Reserved station specification data
☞ Page 1664 Error invalid station specification data
☞ Page 1664 Automatic update buffer assignment data

■Slave station setting data

<table>
<tr><th colspan="5">Operand: (s2)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td rowspan="4">+0 to +63</td><td rowspan="4">Station information settings[1]</td><td>Specifies the station number, the number of occupied stations, and the station type for each slave station.

The default value range is 0101H to 0140H, meaning that the station number is 1 to 64, the number of occupied stations is 1 station occupied, and the station type is remote I/O station for Ver. 1.</td><td>—</td><td>User</td></tr>
<tr><td>(1) Station number setting
1 to 64</td><td>01H to 40H</td><td>User</td></tr>
<tr><td>(2) Number of occupied stations setting
• 1H: 1 station occupied
• 2H: 2 stations occupied
• 3H: 3 stations occupied
• 4H: 4 stations occupied</td><td>1H to 4H</td><td>User</td></tr>
<tr><td>(3) Station type setting[2]
• 0H: Remote I/O station for Ver. 1
• 1H: Remote device station for Ver. 1
• 2H: Intelligent device station for Ver. 1
• 5H: Single remote device station for Ver. 2
• 6H: Single intelligent device station for Ver. 2
• 8H: Double remote device station for Ver. 2
• 9H: Double intelligent device station for Ver. 2
• BH: Quadruple remote device station for Ver. 2
• BH: Quadruple intelligent device station for Ver. 2
• EH: Octuple remote device station for Ver. 2
• FH: Quadruple intelligent device station for Ver. 2</td><td>0H to FH</td><td>User</td></tr>
</table>

[1] Repeat this setting for the total number of connected modules/stations.

[2] If a local station is specified, specify the intelligent device station.

■Reserved station specification data

<table>
<tr><th colspan="5">Operand: (s3)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td>+0 to +3</td><td>Reserved station specification</td><td>Specifies a reserved station in a station-by-station manner.[1]
• 0: Not specified
• 1: Specified
<table>
<tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>…</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>(s3)+0</td><td>16</td><td>15</td><td>14</td><td>13</td><td>…</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>(s3)+1</td><td>32</td><td>31</td><td>30</td><td>29</td><td>…</td><td>20</td><td>19</td><td>18</td><td>17</td></tr>
<tr><td>(s3)+2</td><td>48</td><td>47</td><td>46</td><td>45</td><td>…</td><td>36</td><td>35</td><td>34</td><td>33</td></tr>
<tr><td>(s3)+3</td><td>64</td><td>63</td><td>62</td><td>61</td><td>…</td><td>52</td><td>51</td><td>50</td><td>49</td></tr>
</table>
The default value is "0: Not specified" for all stations.</td><td>—</td><td>User</td></tr>
</table>

[1] For two or more stations occupied, specify only the slave station start number.

■Error invalid station specification data

<table>
<tr><th colspan="5">Operand: (s4)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td>+0 to +3</td><td>Error invalid station specification</td><td>Specifies an error invalid station in a station-by-station manner.[1][2]
• 0: Not specified
• 1: Specified
<table>
<tr><th></th><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>…</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>(s4)+0</td><td>16</td><td>15</td><td>14</td><td>13</td><td>…</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>(s4)+1</td><td>32</td><td>31</td><td>30</td><td>29</td><td>…</td><td>20</td><td>19</td><td>18</td><td>17</td></tr>
<tr><td>(s4)+2</td><td>48</td><td>47</td><td>46</td><td>45</td><td>…</td><td>36</td><td>35</td><td>34</td><td>33</td></tr>
<tr><td>(s4)+3</td><td>64</td><td>63</td><td>62</td><td>61</td><td>…</td><td>52</td><td>51</td><td>50</td><td>49</td></tr>
</table>
The default value is "0: Not specified" for all stations.</td><td>—</td><td>User</td></tr>
</table>

[1] For two or more stations occupied, specify only the slave station start number.

[2] If both the reserved and error invalid stations are specified for the same station, the reserved station specification will take priority.

■Automatic update buffer assignment data

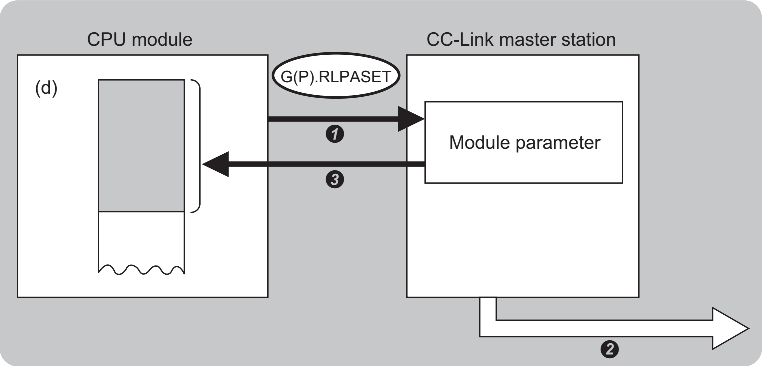
<table>
<tr><th colspan="5">Operand: (s5)</th></tr>
<tr><th>Device</th><th>Item</th><th>Description</th><th>Setting range</th><th>Set by</th></tr>
<tr><td>+0 to +77</td><td>Automatic update buffer assignment specification</td><td>Specifies the assigned buffer memory size (words) that is used for the transient transmission with the automatic update buffer that is performed to the local or intelligent device station.[1]
• 0: Not specified
• 1: Specified
<table>
<tr><td>(s5)+0</td><td>Dummy</td><td rowspan="3">For the 1st module</td></tr>
<tr><td>(s5)+1</td><td>Dummy</td></tr>
<tr><td>(s5)+2</td><td>Automatic update buffer size</td></tr>
<tr><td></td><td>⋮</td><td></td></tr>
<tr><td>(s5)+75</td><td>Dummy</td><td rowspan="3">For the 26th module</td></tr>
<tr><td>(s5)+76</td><td>Dummy</td></tr>
<tr><td>(s5)+77</td><td>Automatic update buffer size</td></tr>
</table>
The default value is 0080H.</td><td>0H (no setting)
0080H to 1000H[2]</td><td>User</td></tr>
</table>

[1] Beginning at the smallest station number, set the size for the slave stations for which the local or intelligent device station has been set with the slave station setting data ((s2)+0 to (s2)+63).

[2] The automatic update buffer size must be 1000H (4096) words or less in total. For the automatic update buffer size, specify the required size for each intelligent device station.

Processing details

- This instruction sets the network parameters in the master station and starts up the data link.
- This instruction can be executed only for the master station.
- The following figure shows how the G(P).RLPASET instruction operates during execution.



❶ The network parameters specified by (s1) to (s5) are written to the master station.
❷ The data link is started up.
❸ The device specified by (d) is turned on.

- The completion status of the G(P).RLPASET instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the G(P).RLPASET instruction completes, and turns off during the next END processing.

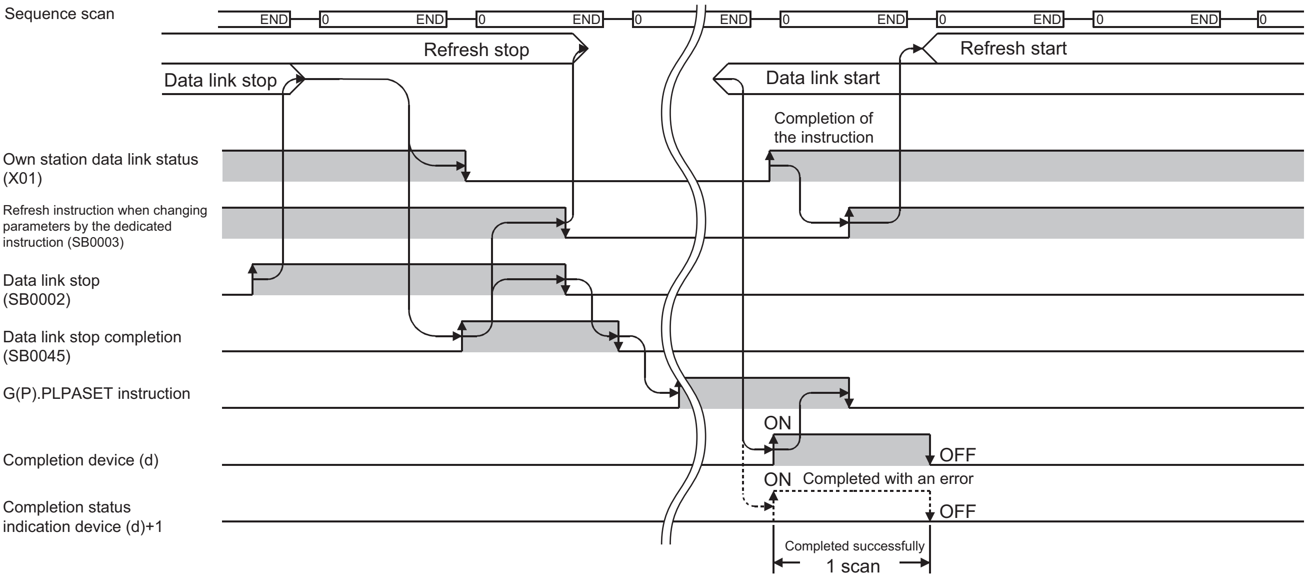
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).RLPASET instruction.

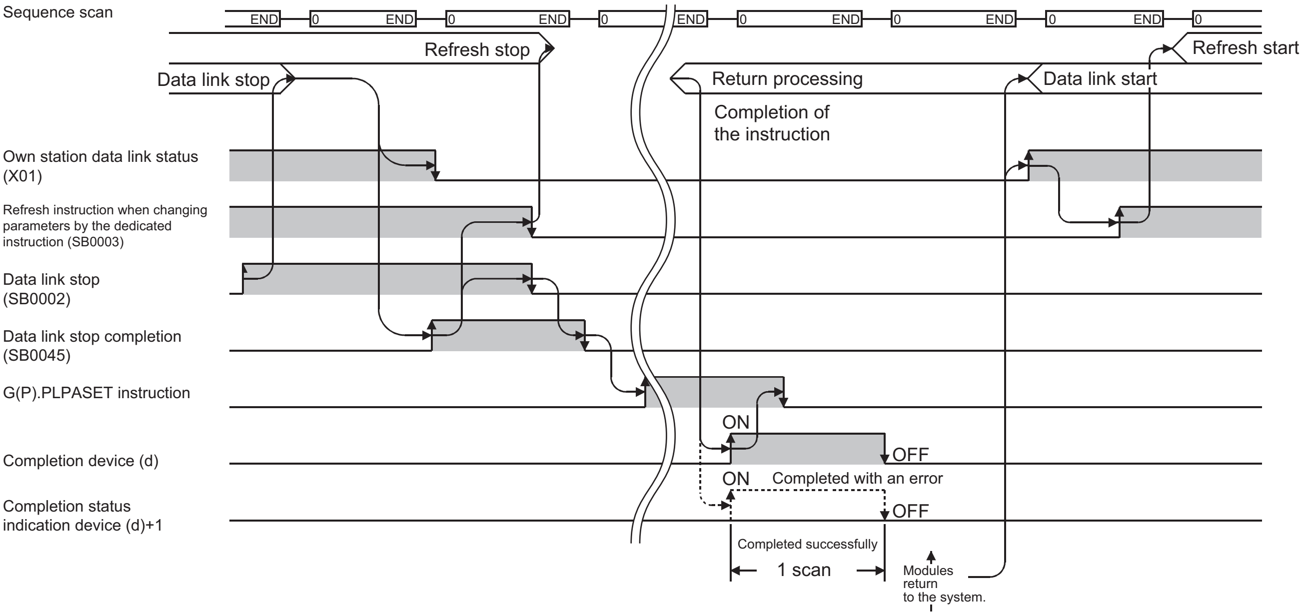
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).RLPASET instruction completes, and turns off during the next END processing.

- After completion of the G(P).RLPASET instruction, turn on SB0003, a refresh instruction that is used for parameter change with the dedicated instruction, to start the cyclic data refresh.
  - If no stations are faulty



  - If all stations are faulty



Precautions

- Two or more G(P).RLPASET instructions cannot be executed concurrently.
- The G(P).RLPASET instruction is not available in a system containing a standby master station.
- Do not set network parameters using the engineering tool for modules for which network parameters are set using the G(P).RLPASET instruction. If the G(P).RLPASET instruction is executed for a module for which network parameters are set using the engineering tool, the instruction will be terminated with an error and the network parameter settings are not reflected.
- Stop the data link using "Data Link Stop" (SB0002) before executing the G(P).RLPASET instruction.

Operation error

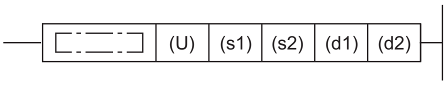
Error code ((s1)+0)	Description
B000H to BFFFH	📖 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

19.8 Performing a Message Transmission to a Remote Device Station

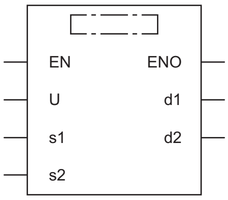
G(P).RDMSG

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

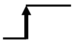
These instructions read/write parameters from/to the remote device station and read the status of the remote device station. The instructions can be executed to a remote device station, for example NZ2AW1C2AL, that supports the message transmission function.

Ladder	ST
	ENO:=G_RDMSG(EN,U,s1,s2,d1,d2); ENO:=GP_RDMSG(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.RDMSG	
GP.RDMSG	

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	Refer to the control data.	Device name	ANY16[2]
(s2)[1]	Start device for storing the message data to be sent	—	Device name	ANY16[2]
(d1)[1]	Start device for storing the message data received	—	Device name	ANY16
(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] For details on the send data and receive data, refer to the following.
📖 Manuals for the remote device stations that support the message transmission function

[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

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	$	
(U)[*1]	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

[*1] Index modification is not available.

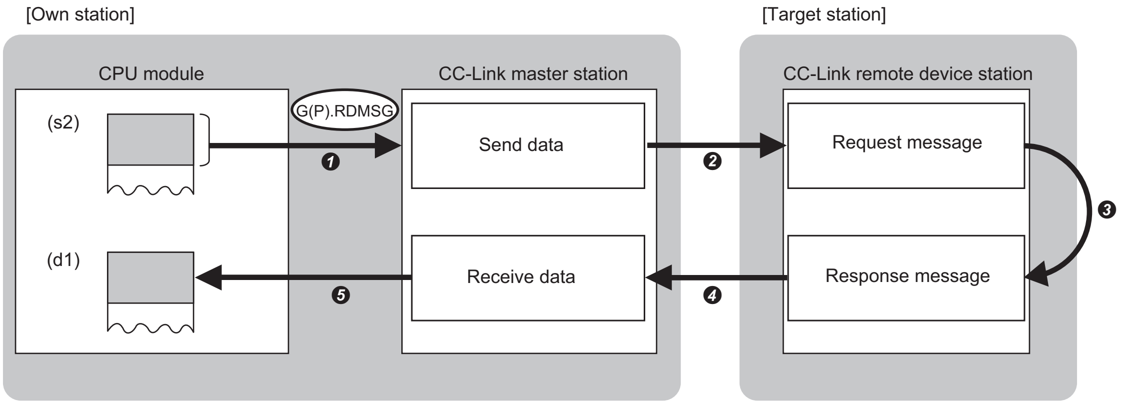
■Control data

Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	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
+1	Station number	Specify the station number of the target station.	1 to 64	User
+2	Send data size	Specify the send message data size in bytes.	1 to 255	User
+3	Receivable data size	Specify the maximum size of the device that stores the receive message data in bytes.	0 to 255	User
+4	Receive data size	The receive message data size is stored in bytes.	—	System

Processing details

- These instructions performs a message transmission to a remote device station.
- The instructions can be executed to a remote device station, for example NZ2AW1C2AL, that supports the message transmission function.
- The following figure shows how the G(P).RDMSG instruction operates during execution.



❶ The send data specified by (s2) is stored to the master station for the size specified by (s1)+2.
❷ The master station sends data to the target station specified by (s1)+1.
❸ Processing is performed for the data in the target station.
❹ The master station receives a processing result from the target station.
❺ The master station stores data in the device specified by (d1) and later, and the device specified by (d2) turns on.

- The completion status of the G(P).RDMSG instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

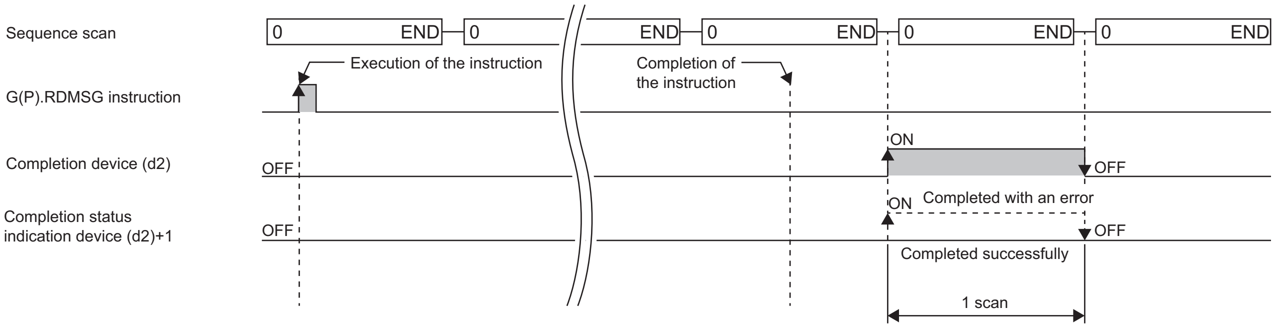
This device turns on during END processing of the scan where the G(P).RDMSG instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

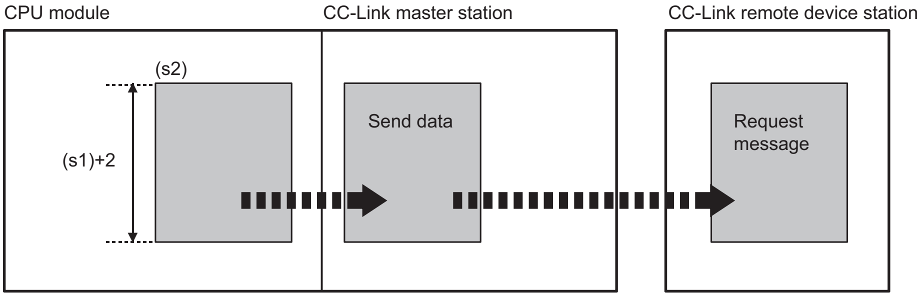
This device turns on or off depending on the completion status of the G(P).RDMSG instruction.

When completed successfully: The device remains off.

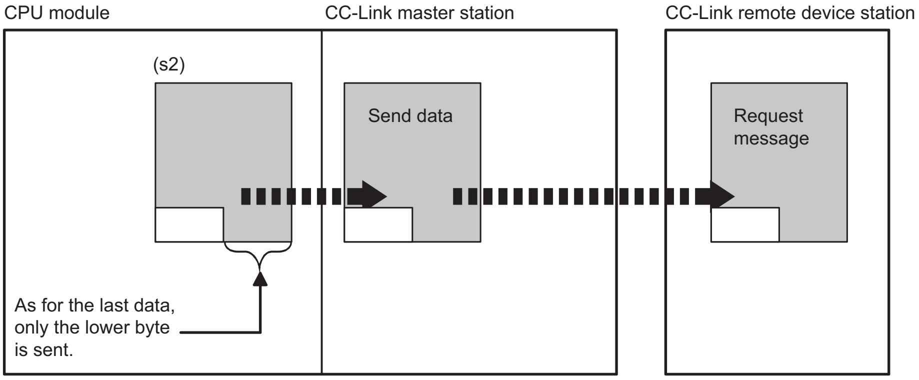
When completed with an error: The device turns on during END processing of the scan where the G(P).RDMSG instruction completes, and turns off during the next END processing.



■Send data size



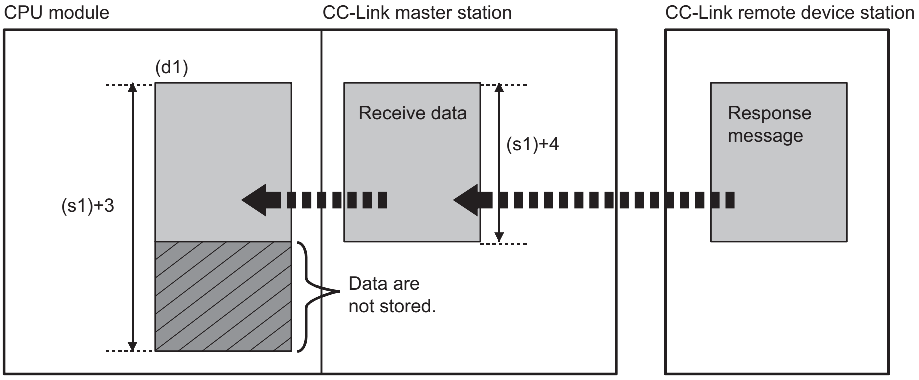
When the number of bytes in the send data ((s1)+2) is odd, the lower byte is send as for the last data.



■Receivable data size and receive data size

Set the receivable data size ((s1)+3) so that it satisfies the following.

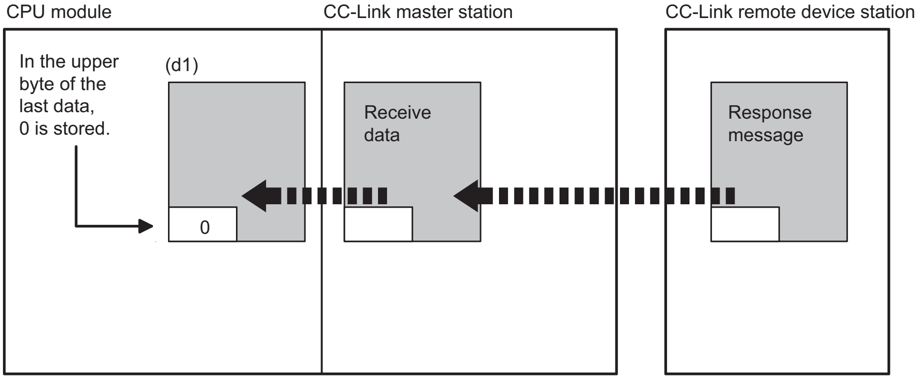
Receivable data size ((s1)+3) ≥ Receive data size ((s1)+4)



If the receivable data size ((s1)+3) is smaller than the receive data size ((s1)+4), the master station cannot receive data from the remote device station. The G(P).RDMSG instruction completes with an error (error code: B418H).

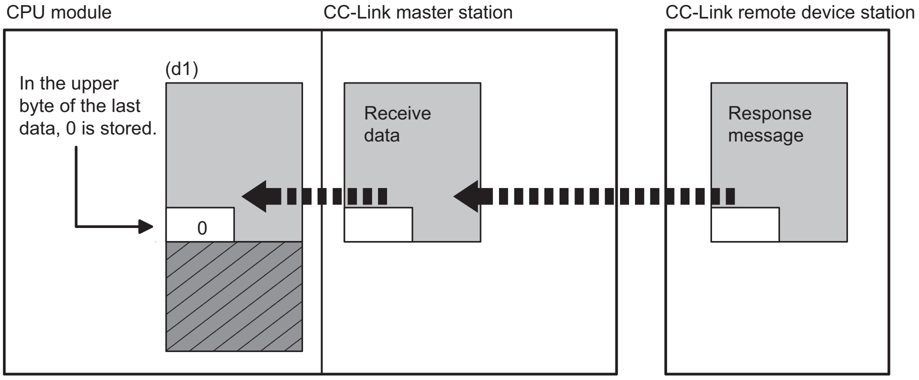
- When the number of bytes in the receivable data ((s1)+3) is odd

If the receive data has the same number of bytes, 0 is stored in the upper byte of the last data.



- When the number of bytes in the receive data ((s1)+4) is odd

The last receive data is stored in the lower byte of the last data storage area in the device memory. In the upper byte of the last data storage area, 0 is stored.



Precautions

- The G(P).RDMSG instruction can be simultaneously executed to two or more remote device stations (up to four stations). Note, however, that only one dedicated instruction can be executed to a single remote device station. If a dedicated instruction is executed before the processing of another instruction has not been completed, the instruction executed later will complete with an error. Create the program so that the next dedicated instruction will start after the completion device turns on, because processing of a dedicated instruction takes several scans until its completion.
- The G(P).RDMSG instruction uses a part or all of remote register that performs cyclic transmission between the master station and the target station in the system. For programming, refer to the manual for the remote device station targeted. Add SW0160 to SW0163 (Remote register use prohibited status) to a program as an interlock.

Operation error

Error code ((s1)+0)	**Description**
B000H to BFFFH	📖 MELSEC iQ-R CC-Link System Master/Local Module User's Manual (Application)

20 SERIAL COMMUNICATION INSTRUCTIONS

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. For the instructions when MELSEC-Q series modules are used, refer to the manuals for each module used and create programs.

For precautions when using modules, refer to the following.

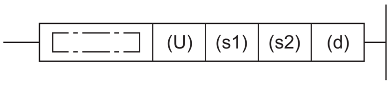
📖 MELSEC iQ-R Module Configuration Manual

20.1 Sending Data Using the On-Demand Function

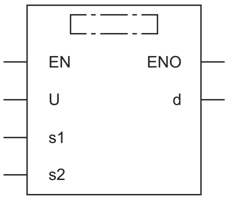
G(P).ONDEMAND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions send data of the specified amount from the specified device by using the MC protocol on-demand function.

Ladder	ST
	ENO:=G_ONDEMAND(EN,U,s1,s2,d); ENO:=GP_ONDEMAND(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.ONDEMAND	
GP.ONDEMAND	

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	Refer to the control data.	Device name	ANY16[1]
(s2)	Start device where the send data is stored	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)*1	—	—	○*3	—	—	—	—	○	○	—	—	○
(s1)	—	—	○*4	—	—	—	—	○	—	—	—	—
(s2)	—	—	○*4	—	—	—	—	○	—	—	—	—
(d)	○*2	—	○*5	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.

*2 FX and FY cannot be used.

*3 RD cannot be used.

*4 FD cannot be used.

*5 T, ST, C, and FD cannot be used.

■Control data

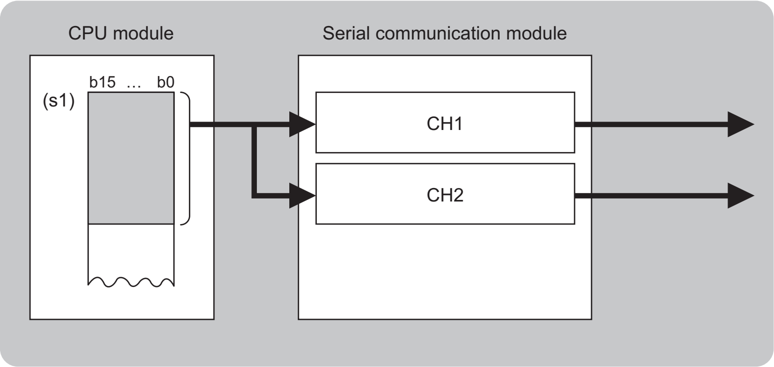
Operand: (s1)

Device	Item	Description	Setting range	Set by
+0	Send channel	Specifies the send channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of sending	Used to store the result of sending with the G(P).ONDEMAND instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Amount of send data	Specifies the amount of the data to be sent.*1	1 or more	User

*1 The amount of send data must be set in units as specified in the engineering tool. For word specification, set the number of words. For byte specification, set the number of bytes.

Processing details

- By using the on-demand function of MC protocol of the module specified by (U), the data stored in the device specified by (s2) and later is sent according to the control data in the device specified by (s1) and later.



CH: Channel

- The completion status of the G(P).ONDEMAND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the G(P).ONDEMAND instruction completes, and turns off during the next END processing.

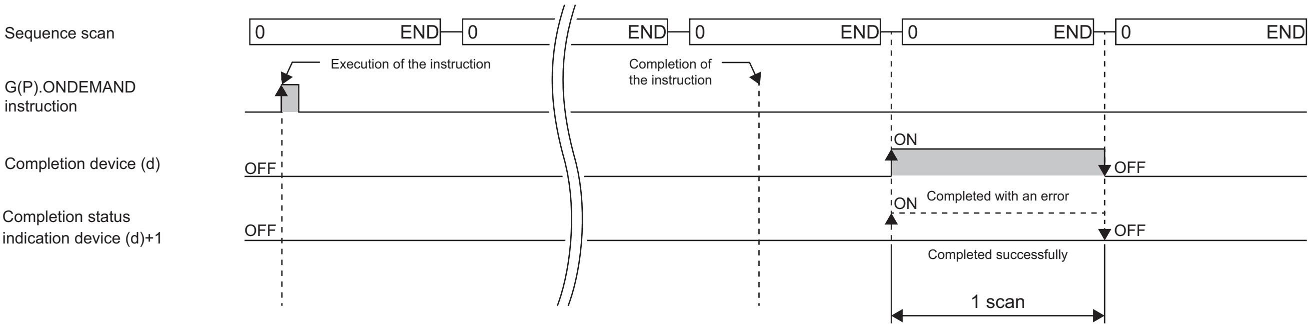
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).ONDEMAND instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).ONDEMAND instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).ONDEMAND instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).ONDEMAND instruction is active or the G(P).ONDEMAND instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
ZP.CSET	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

*2 The following instructions are not available with the same channel as for the G(P).ONDEMAND instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G(P).ONDEMAND instruction, a communication protocol setting error (7FF2H) will occur. (except for the Z.BUFRCVS instruction).

- G(P).CPRTCL, G(P).OUTPUT, G.INPUT, G(P).BIDOUT, Z.BUFRCVS, and G(P).PRR instructions

- The local device and the file register for each program are not available for setting data.

Operation error

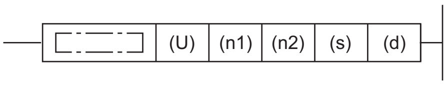
Error code ((s1)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.2 Executing the Protocols Registered for the Predefined Protocol Support Function

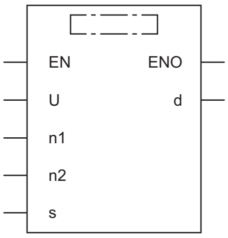
G(P).CPRTCL

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

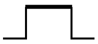
These instructions execute the protocols or special protocols that have been written to the flash ROM by using the communication protocol support function of the engineering tool.

Ladder	ST
	ENO:=G_CPRTCL(EN,U,n1,n2,s,d); ENO:=GP_CPRTCL(EN,U,n1,n2,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.CPRTCL	
GP.CPRTCL	

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
(n1)	Channel to communicate with the external device • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	16-bit unsigned binary	ANY16
(n2)	Number of protocols to be executed consecutively (1 to 8)	1 to 8	16-bit unsigned binary	ANY16
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[1]	—	—	○	—	—	—	—	○	○	—	—	○
(n1)	—	—	○[3]	—	—	—	—	○	○	—	—	—
(n2)	—	—	○[3]	—	—	—	—	○	○	—	—	—
(s)	—	—	○[3]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[3]	—	—	—	—	—	—	—	—	—

[1] Index modification is not available.

[2] FX and FY cannot be used.

[3] FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	Execution result	Used to store the execution result of the G(P).CPRTCL instruction. If two or more protocols are executed, the execution result of the last protocol will be stored.[1] • 0: Normal • Other than 0: Error (error code)	—	System
+1	Resulting number of executed protocols	Used to store the number of actually executed protocols. Also for the protocols that caused an error, the execution result of the last protocol will be stored. If an error is included in the setting data or the control data setting details, "0" will be stored.	1 to 8	System
+2	Execution protocol number specification	Sets the number of the first protocol or special protocol to be executed.[2]	1 to 128, 201 to 207	User
⋮		⋮		
+9		Sets the number of the eighth protocol or special protocol to be executed.[2]		
+10	Matched receive packet number	Used to store the matched receive packet number if the communication type of the first protocol executed is "Send only" or "Send and receive". If the first protocol caused an error during its execution, "0" will be stored. If a special protocol is executed, "0" will be stored.[2]	0, 1 to 16	System
⋮		⋮		
+17		Used to store the matched receive packet number if the communication type of the eighth protocol executed is "Send only" or "Send and receive". If the eighth protocol caused an error during its execution, "0" will be stored. If a special protocol is executed, "0" will be stored.[2]		

[1] If the nth protocol caused an error during its execution while two or more protocols are being executed, none of the subsequent protocols will be executed.

[2] For details on the special protocols, refer to the following.
☞ Page 1680 Special protocol

Processing details

- The protocol setting data written to the flash ROM is executed using the module specified by (U). All the protocols to be executed follow the contents of the control data that has been stored in the device specified by (s) and later. The channel specified by (n1) is used.
- One attempt of the instruction consecutively executes the specified number of protocols (by (n2); maximum 8).
- The completion status of the G(P).CPRTCL instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the G(P).CPRTCL instruction completes, and turns off during the next END processing.

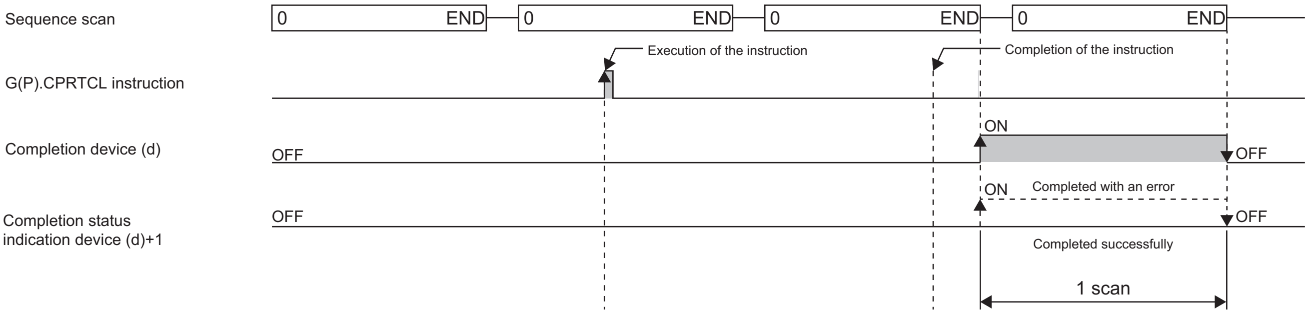
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).CPRTCL instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).CPRTCL instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).CPRTCL instruction operates when completing its execution.



Point

The program execution status can be checked with the buffer memory (address: 4041H/4051H).

■Canceling protocol execution

A protocol can be canceled during its execution.

This function forcibly terminates the protocol being executed when a communication error occurs with the external device.

The function can be executed in the predefined protocol mode only.

- How to cancel the protocol execution

A cancellation can be requested using a sequence program.

Use the following buffer memory areas.

Address (decimal (hexadecimal))		Name	Set value
CH1	CH2		
16448 (4040H)	16464 (4050H)	Protocol cancellation specification	0: No cancellation specified 1: Cancellation requested (by user) 2: Canceled (by the C24)

- Operation after the cancellation is requested

[Operation of the dedicated instruction (G(P).CPRTCL)]

The instruction completes with an error, and the error code is stored in (s)+0 (Execution result). If multiple protocols are being executed sequentially, the cancellation-target protocol ('n'th protocol) is forcibly terminated, and the subsequent protocols are not executed.

[Operation of the C24]

If the function is executed (a cancellation is requested) while no protocol is being executed, the C24 performs no processing.

If the function is executed in the mode other that the predefined protocol mode, the value in the corresponding buffer memory area is ignored.

Point

- After the protocol whose communication type is "Send and receive" is canceled, clear the receive data. When data is received after cancellation, the receive data is held in the OS area (receive data area). This occurs when a response from the external device is slow. For how to clear receive data, refer to Page 1702 Clearing Receive Data.
- While the value set to "Protocol cancellation specification" is 1 (Cancellation requested), another G(P).CPRTCL instruction cannot be executed.

■Special protocol

The following functions are available by executing special protocols with the G(P).CPRTCL instruction.

Functions	Description
Clearing receive data	Enables the receive data to be cleared from the OS area.
Send/receive data monitoring start/stop	Specifies the start or stop of send/receive data monitoring. When special protocols 202 or 203 is executed, the module sets the monitor start instruction "0001H" or the monitor stop instruction "0000H" for the send/receive data monitoring specification (address: 2018H, 2118H).
RS/DTR signal status specification	Specifies the on/off state of the RS(RTS) and ER(DTR) signals. When one of special protocols 204 to 207 is executed, the module turns on/off the bit corresponding to the RS/DTR signal status specification (address: 92H/132H). For details on the RS/DTR signal status specification, refer to the following. 📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

In control data (s)+2 to (s)+9 of the G(P).CPRTCL instruction, specify the number of the special protocol of the function to be executed.

Functions	Special protocol number	Type	Keyword[1]
Clearing receive data	201	Clearing receive data	Receive Data Clear
Send/receive data monitoring start/stop	202	Start send/receive data monitoring	Send/Recv Monitor Start
	203	Stop send/receive data monitoring	Send/Recv Monitor Stop
RS/DTR signal status specification	204	Turn on the ER (DTR) signal	DTR ON
	205	Turn off the ER (DTR) signal	DTR OFF
	206	Turn on the RS (RTS) signal	RS ON
	207	Turn off the RS (RTS) signal	RS OFF

[1] When a special protocol is executed, the keyword is defined as the character string that is stored in the protocol name of the protocol execution history.

Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).CPRTCL instruction is active or the G(P).CPRTCL instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).CPRTCL	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

[2] The following instructions are not available with the same channel as for the G(P).CPRTCL instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G(P).CPRTCL instruction, a communication protocol setting error (7FF2H) will occur (except for the Z.BUFRCVS instruction).

- G(P).ONDEMAND, G(P).OUTPUT, G.INPUT, G(P).BIDOUT, G(P).BIDIN, Z.BUFRCVS, G(P).PRR, and ZP.CSET instructions

- The local device and the file register for each program are not available for setting data.

Operation error

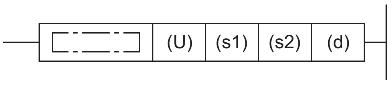
Error code ((s)+0)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

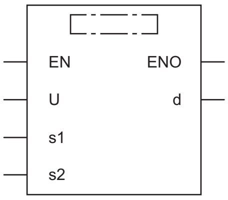
20.3 Sending Data Using the Nonprocedural Protocol

G(P).OUTPUT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions send data in a user-defined message format by using the nonprocedural protocol.

Ladder	ST
	ENO:=G_OUTPUT(EN,U,s1,s2,d); ENO:=GP_OUTPUT(EN,U,s1,s2,d);

FBD/LD


■Execution condition

Instruction	Execution condition
G.OUTPUT	
GP.OUTPUT	

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	Refer to the control data.	Device name	ANY16[1]
(s2)	Start device where the send data is stored	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[1]	—	—	○[3]	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[5]	—	—	—	—	—	—	—	—	—

[1] Index modification is not available.
[2] FX and FY cannot be used.
[3] RD cannot be used.
[4] FD cannot be used.
[5] T, ST, C, and FD cannot be used.

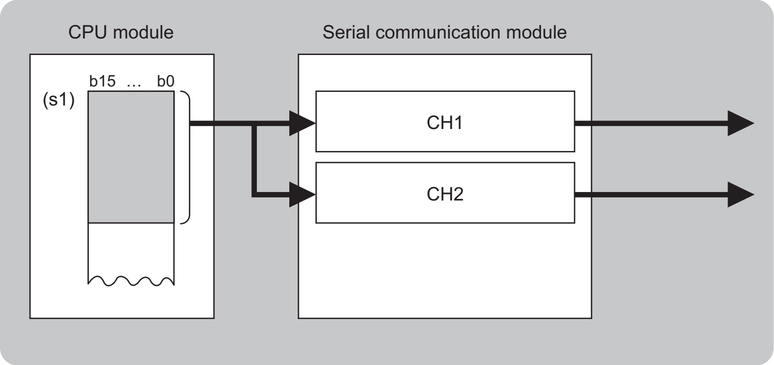
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Send channel	Specifies the send channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of sending	Used to store the result of sending with the G(P).OUTPUT instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Amount of send data	Specifies the amount of the data to be sent.[1]	1 or more	User

[1] The amount of the data to be sent must be set in units as specified in the engineering tool. For byte specification, set the number of bytes. For word specification, set the number of words.

Processing details

- By using the nonprocedural protocol of the module specified by (U), the data stored in the device specified by (s2) and later is sent according to the control data in the device specified by (s1) and later.



CH: Channel

- The completion status of the G(P).OUTPUT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the G(P).OUTPUT instruction completes, and turns off during the next END processing.

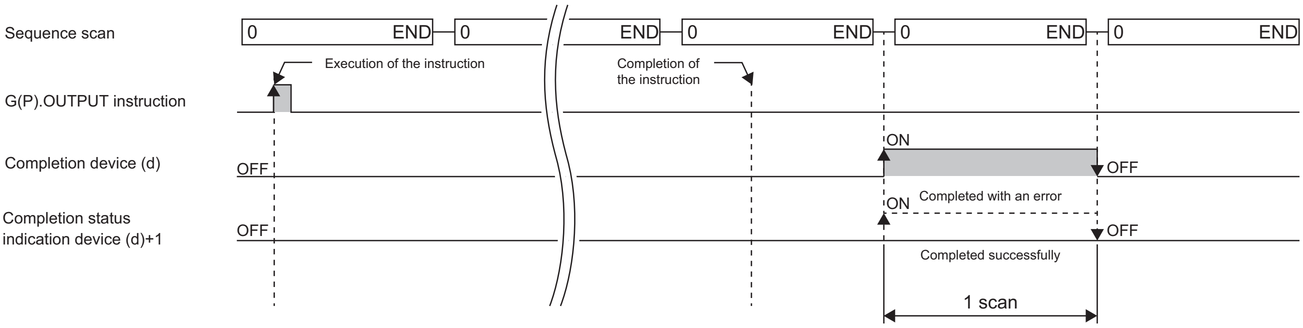
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).OUTPUT instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).OUTPUT instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).OUTPUT instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).OUTPUT instruction is active or the G(P).OUTPUT instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).OUTPUT	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G.INPUT	○	—
G(P).PRR	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
ZP.CSET		
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

*2 The following instructions are not available with the same channel as for the G(P).OUTPUT instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G(P).OUTPUT instruction, a communication protocol setting error (7FF2H) will occur.

- G(P).ONDEMAND, G(P).CPRTCL, G(P).BIDOUT, and G(P).BIDIN instructions

- The local device and the file register for each program are not available for setting data.

Operation error

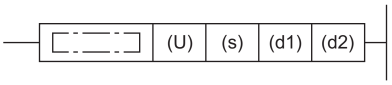
Error code ((s1)+1)	Description
7000H to 7FFFH	🕮 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.4 Receiving Data Using the Nonprocedural Protocol

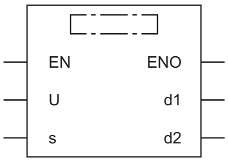
G.INPUT

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

This instruction receives data in a user-defined message format by using the nonprocedural protocol.

Ladder	ST
	ENO:=G_INPUT(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.INPUT	

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
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d1)	Start device for storing the receive data	—	Device name	ANY16[*1]
(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 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

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(U)
(U)[*1]	—	—	○[*3]	—	—	—	—	○	○	—	—	○
(s)	—	—	○[*4]	—	—	—	—	○	—	—	—	—
(d1)	—	—	○[*4]	—	—	—	—	○	—	—	—	—
(d2)	○[*2]	—	○[*5]	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 RD cannot be used.
*4 FD cannot be used.
*5 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive channel	Specifies the receive channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of receiving	Used to store the result of receiving with the G.INPUT instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Amount of receive data	Stores the number of data received.[1]	—	System
+3	Allowable amount of receive data	Sets the allowable number of words of receive data that can be stored in (d1).	1 or more	User

[1] The amount of the data to be received is set in units as specified in the engineering tool. For byte specification, the number of bytes is set. For word specification, the number of words is set.

Processing details

- The data received through the nonprocedural protocol of the module specified by (U) is stored in the device specified by (d1) and later, according to the control data in the device specified by (s) and later.
- If the amount of actually received data is larger than the allowable amount of receive data specified by (s)+3, the data for the allowable amount of receive data is only stored with the remaining receive data discarded. In this case, the allowable amount of receive data is stored in the amount of receive data (s)+2. (the instruction completes successfully).
- The completion status of the G.INPUT instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the G.INPUT instruction completes, and turns off during the next END processing.

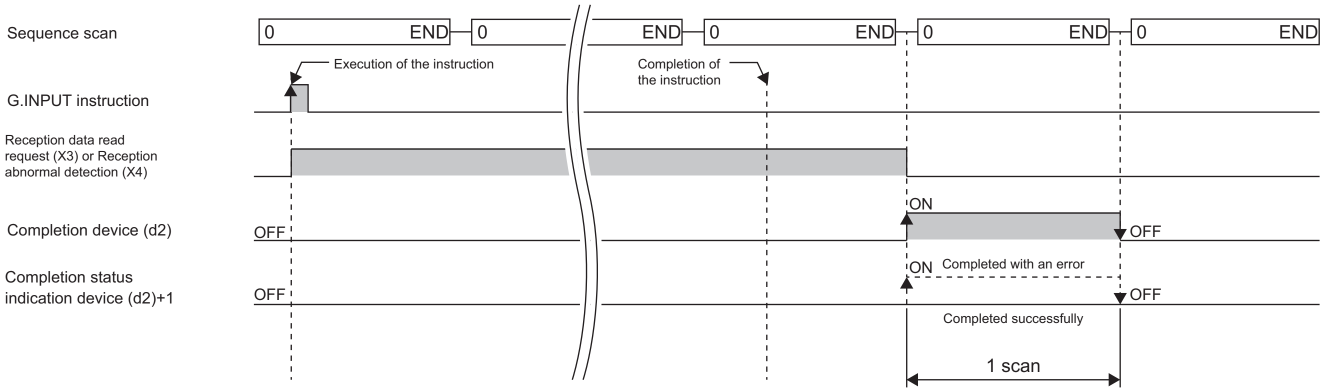
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the G.INPUT instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G.INPUT instruction completes, and turns off during the next END processing.

- The following figures show how the G.INPUT instruction operates when its execution completes.



Precautions

- Any command of G.INPUT cannot be pulse converted.
- G.INPUT must be executed while the I/O signal read request signal is on.
- The following table summarizes the processes that take place if another instruction is issued while the G.INPUT instruction is active or the G.INPUT instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[2]	**Possibility of concurrent execution**	**Handling for concurrent execution**
G(P).OUTPUT	○	—
G(P).PRR		
G.INPUT	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
ZP.CSET	See the right	• If the G.INPUT and ZP.CSET instructions are issued in this order A dedicated instruction concurrent execution error (7FF0H) occurs in the ZP.CSET instruction. Concurrent execution is, however, possible if they use different channels. • If the ZP.CSET and G.INPUT instructions are issued in this order Concurrent execution is possible.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS	×	The G.INPUT and Z.BUFRCVS instructions cannot be used simultaneously. Concurrent execution is, however, possible if they use different channels.
G(P).SPBUSY	○	—
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

[2] The following instructions are not available with the same channel as for the G.INPUT instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G.INPUT instruction, a communication protocol setting error (7FF2H) will occur.

- G(P).ONDEMAND, G(P).CPRTCL, G(P).BIDOUT, and G(P).BIDIN instructions

- The local device and the file register for each program are not available for setting data.

Operation error

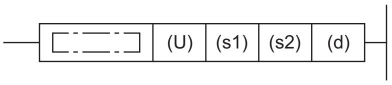
Error code ((s)+1)	**Description**
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.5 Sending Data Using the Bidirectional Protocol

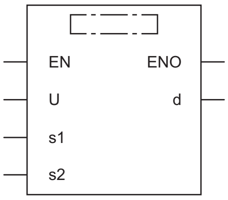
G(P).BIDOUT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

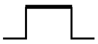
These instructions send data using the bidirectional protocol.

Ladder	ST
	ENO:=G_BIDOUT(EN,U,s1,s2,d); ENO:=GP_BIDOUT(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.BIDOUT	
GP.BIDOUT	

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	Refer to the control data.	Device name	ANY16[1]
(s2)	Start device where the send data is stored	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[1]	—	—	○[3]	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[5]	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 RD cannot be used.
*4 FD cannot be used.
*5 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Send channel	Specifies the send channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of sending	Used to store the result of sending with the G(P).BIDOUT instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Amount of send data	Specifies the amount of the data to be sent.[1]	1 or more	User

*1 The amount of the data to be sent must be set in units as specified in the engineering tool. For byte specification, set the number of bytes. For word specification, set the number of words.

Processing details

- By using the bidirectional protocol of the module specified by (U), the data stored in the device specified by (s2) and later is sent according to the control data in the device specified by (s1) and later.
- The completion status of the G(P).BIDOUT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

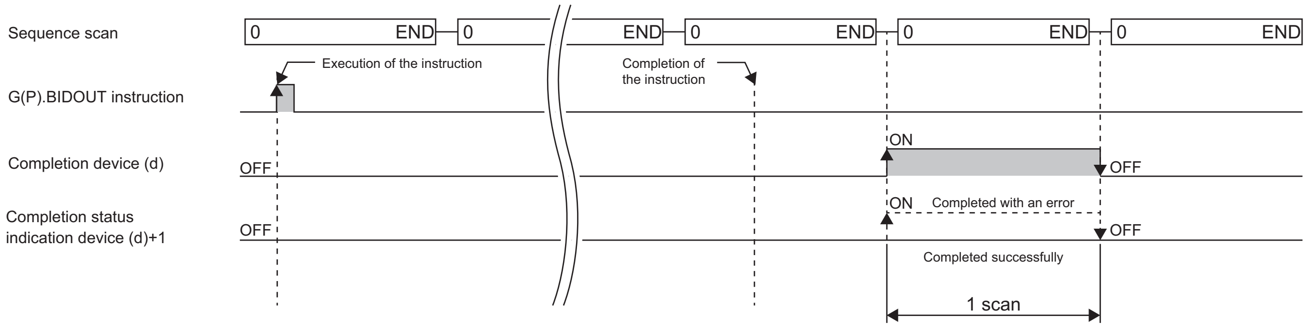
- Completion device (d)

This device turns on during END processing of the scan where the G(P).BIDOUT instruction completes, and turns off during the next END processing.

- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).BIDOUT instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the G(P).BIDOUT instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).BIDOUT instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).BIDOUT instruction is active or the G(P).BIDOUT instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).BIDOUT	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).BIDIN	○	—
ZP.CSET	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

[*2] The following instructions are not available with the same channel as for the G(P).BIDOUT instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G(P).BIDOUT instruction, a communication protocol setting error (7FF2H) will occur.

- G(P).ONDEMAND, G(P).CPRTCL, G(P).BIDOUT, and G(P).BIDIN instructions

Operation error

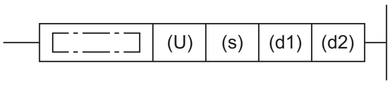
Error code ((s1)+1)	Description
7000H to 7FFFH	🕮 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.6 Receiving Data Using the Bidirectional Protocol

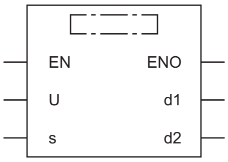
G(P).BIDIN

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

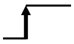
These instructions receive data using the bidirectional protocol.

Ladder	ST
	ENO:=G_BIDIN(EN,U,s,d1,d2); ENO:=GP_BIDIN(EN,U,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
G.BIDIN	
GP.BIDIN	

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
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)	Start device for storing the receive data	—	Device name	ANY16[1]
(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] 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

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	$	
(U)*1	—	—	○*3	—	—	—	—	○	○	—	—	○
(s)	—	—	○*4	—	—	—	—	○	—	—	—	—
(d1)	—	—	○*4	—	—	—	—	○	—	—	—	—
(d2)	○*2	—	○*5	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 RD cannot be used.
*4 FD cannot be used.
*5 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive channel	Specifies the receive channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of receiving	Used to store the result of sending with the G(P).BIDIN instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Amount of receive data	Stores the number of data received.	—	System
+3	Allowable amount of receive data	Sets the allowable number of words of receive data that can be stored in (d1).*1	1 or more	User

*1 The amount of the data to be received must be set in units as specified in the engineering tool. For byte specification, set the number of bytes. For word specification, set the number of words.

Processing details

- The data received using the bidirectional protocol of the module specified by (U) is stored in the device specified by (d1) and later, according to the control data in the device specified by (s) and later.
- The completion status of the G(P).BIDIN instruction can be checked with the completion device (d2) and the completion status indication device (d)+1.

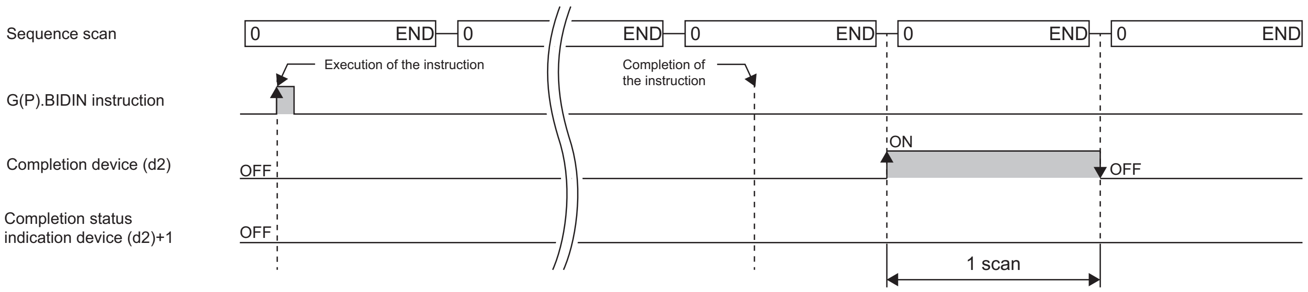
- Completion device (d2)

This device turns on during END processing of the scan where the G(P).BIDIN instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

Unchanged from off.

- The following figure shows how the G(P).BIDIN instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).BIDIN instruction is active or the G(P).BIDIN instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).BIDOUT	×	—
G(P).BIDIN	○	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
ZP.CSET	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS	×	Concurrent execution of the G(P).BIDIN and Z.BUFRCVS instructions is not possible. Concurrent execution is, however, possible if they use different channels.
G(P).SPBUSY	○	—
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

*2 The following instructions are not available with the same channel as for the G(P).BIDIN instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the G(P).BIDIN instruction, a communication protocol setting error (7FF2H) will occur.

- G(P).ONDEMAND, G(P).CPRTCL, G(P).OUTPUT, G.INPUT, and G(P).PRR instructions

Operation error

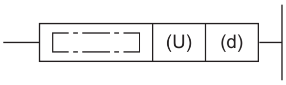
Error code ((s)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

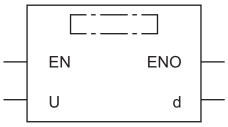
20.7 Reading the Data Send/Receive Status

G(P).SPBUSY

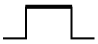
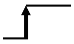
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

These instructions read the send/receive status of data using the instruction.

Ladder	ST
 (U) (d)	ENO:=G_SPBUSY(EN,U,d); ENO:=GP_SPBUSY(EN,U,d);

FBD/LD


■Execution condition

Instruction	Execution condition
G.SPBUSY	
GP.SPBUSY	

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
(d)	Start device for storing the read communication status	—	Device name	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

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	$	
(U)[1]	—	—	○	—	—	—	—	○	○	—	—	○
(d)	—	—	○[2]	—	—	—	—	○	—	—	—	—

[1] Index modification is not available.

[2] FD cannot be used.

Processing details

- The execution status of the instruction to the module specified by the start I/O number is read and stored in the device specified by (d) and later.
- In the associated bit of the device specified by (d), "1" is stored when each instruction starts its processing, and "0" is stored when the instruction completes its processing. Completion of processing of each instruction is at the time when the completion flag of the instruction changes from on to off.

	b15	···	b6	b5	b4	b3	b2	b1	b0
(d)+0			(7)	(6)	(5)	(4)	(3)	(2)	(1)
(d)+1	0								

(1) Execution status of the G(P).ONDEMAND, G(P).OUTPUT, G(P).PRR, or G(P).BIDOUT instruction for channel 1
(2) Execution status of the G.INPUT or G(P).BIDIN instruction for channel 1
(3) Execution status of the G(P).ONDEMAND, G(P).OUTPUT, G(P).PRR, or G(P).BIDOUT instruction for channel 2
(4) Execution status of the G.INPUT or G(P).BIDIN instruction for channel 2
(5) Execution status of the G(P).GETE or G(P).PUTE instruction
(6) Execution status of the G(P).CPRTCL instruction for channel 1
(7) Execution status of the G(P).CPRTCL instruction for channel 2

- If the instruction has been set to be executed while being on, the G(P).SPBUSY instruction operates every scan while the read instruction is on; if the instruction has been set to be executed at the rise, it operates only one scan when the read instruction rises from off to on.
- While the G(P).SPBUSY instruction is active, another instruction can be executed. In addition, while another instruction is active, the G(P).SPBUSY instruction can be executed.

Operation error

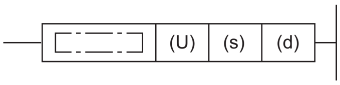
Error code (SD0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)

20.8 Receiving Data Using an Interrupt Program

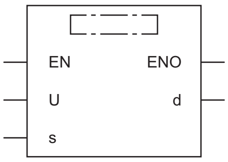
Z.BUFRCVS

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

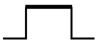
This instruction uses the interrupt program to receive communication data by using the nonprocedural or bidirectional protocol.

Ladder	ST
	ENO:=Z_BUFRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
Z.BUFRCVS	

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	String	ANY16_OR_STRING_SINGLE
(s)	Specifies the receive channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	16-bit unsigned binary	ANY16
(d)	Start device for storing the receive data (The receive data is read from the receive area in the buffer memory.)	—	Device name	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)*1	—	—	○	—	—	—	—	○	—	—	○	○
(s)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(d)	—	—	○*3	—	—	—	—	○	—	—	—	—

*1 Index modification is not available.

*2 FX and FY cannot be used.

*3 FD cannot be used.

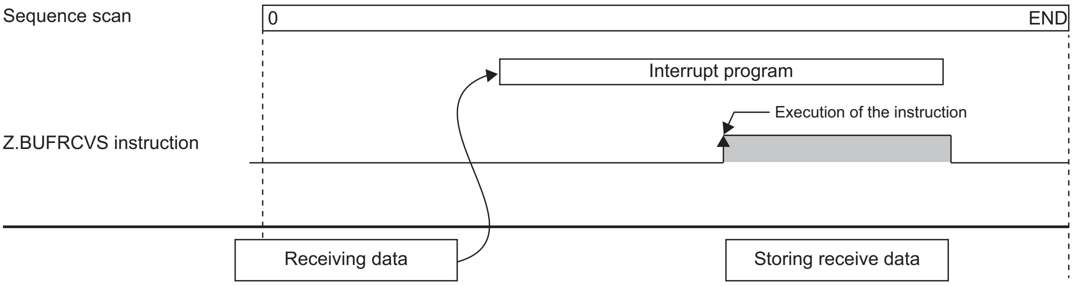
■Receive data

Operand: (d)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	Used to store the amount of data that was read from the data storage area for amount of receive data.[1]	—	System
+1 to n	Receive data	Used to store the data, in ascending order of the address, that was read from the receive data storage area.	—	System

[1] The amount of the data to be received must be set in units as specified in the engineering tool. For byte specification, set the number of bytes. For word specification, set the number of words.

Processing details

- The data received from the external device is stored in the specified device.
- The receive data can be read without considering the address of the receive area of the buffer memory.
- When the Z.BUFRCVS instruction is executed, receive completion takes place; the receive read request signal (Xn3/XnA) or receive error detection signal (Xn4/XnB) turns off automatically. If the receive has been read using the Z.BUFRCVS instruction, the receive read completion signal (Yn1/Yn8) does not need to be turned on.
- The Z.BUFRCVS instruction is used in the interrupt program, and processing completes within one scan.
- The following figures show how the Z.BUFRCVS instruction operates when its execution completes.



Precautions

- When data is to be received with the interrupt program, use the Z.BUFRCVS instruction.
- The following table summarizes the processes that take place if another instruction is issued while the Z.BUFRCVS instruction is active or the Z.BUFRCVS instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*2]	Possibility of concurrent execution	Handling for concurrent execution
G(P).OUTPUT	○	—
G(P).PRR		
G.INPUT	×	The G.INPUT and Z.BUFRCVS instructions cannot be used simultaneously. Concurrent execution is, however, possible if they use different channels.
G(P).BIDOUT	○	—
G(P).BIDIN	×	Concurrent execution of the G(P).BIDIN and Z.BUFRCVS instructions is not possible. Concurrent execution is, however, possible if they use different channels.
ZP.CSET	○	—
G(P).PUTE		
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI		

*2 The following instructions are not available with the same channel as for the Z.BUFRCVS instruction because they use a communication protocol different from the one used by this instruction.

- G(P).ONDEMAND and G(P).CPRTCL instructions

- For the receive data storage device for the Z.BUFRCVS instruction, secure in advance the area larger than the amount of data that will be received from the external device. If it has not been secured, the data located at the rear of the storage device will be overwritten.
- The local device and the file register for each program are not available for setting data.
- The read error codes issued during occurrence of a receive error can be read from the data receive result storage area (address: 258H/268H) of the buffer memory.

Operation error

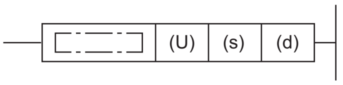
Error code (SD0)	Description
4000H to 4FFFH	📖 MELSEC iQ-R CPU Module User's Manual (Application)

20.9 Sending Data by Using User Frames

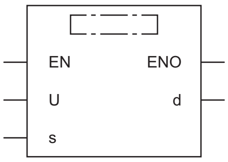
G(P).PRR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

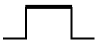
These instructions send data with user frames according to the specification in the user frame specification area for sending, through communication with the nonprocedural protocol.

Ladder	ST
	ENO:=G_PRR(EN,U,s,d); ENO:=GP_PRR(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.PRR	
GP.PRR	

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
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16*1
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)*1	—	—	○*3	—	—	—	—	○	○	—	—	○
(s)	—	—	○*4	—	—	—	—	○	—	—	—	—
(d)	○*2	—	○*4	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 RD cannot be used.
*4 FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Send channel	Specifies the send channel. • 1: Channel 1 (CH1 side) • 2: Channel 2 (CH2 side)	1, 2	User
+1	Result of sending	Used to store the result of sending with the G(P).PRR instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	CR/LF append specification	Specifies whether to append CR or LF to send data.	0, 1	User
+3	Send pointer	Specifies the frame number that indicates the start position of send data within the user frame specification area for sending.	1 to 100	User
+4	Amount of output	Specifies the number of user frames to be sent.	1 to 100	User

Processing details

- By using the nonprocedural protocol of the module specified by (U), the data in the user frames is sent according to the control data stored in the device specified by (s) and later and the user frame specification area for sending of the module.
- The completion status of the G(P).PRR instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the G(P).PRR instruction completes, and turns off during the next END processing.

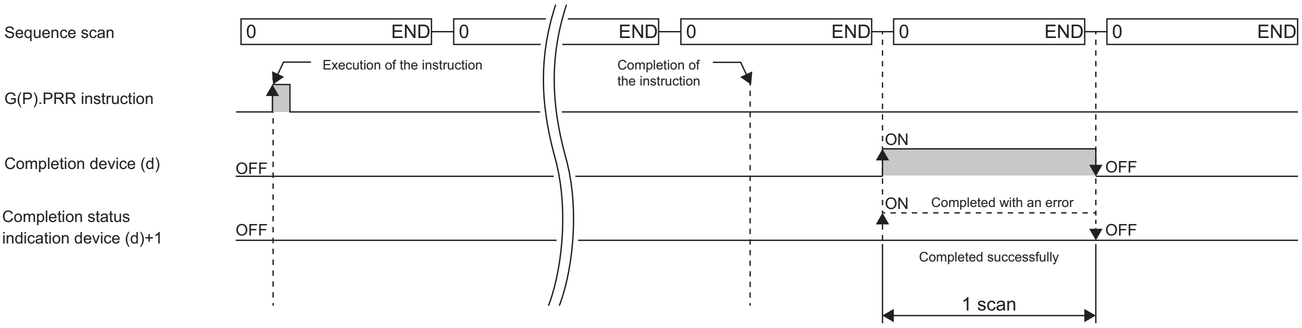
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).PRR instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).PRR instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).PRR instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).PRR instruction is active or the G(P).PRR instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[1]	Possibility of concurrent execution	Handling for concurrent execution
G(P).PRR	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G.INPUT	○	—
G(P).OUTPUT	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
ZP.CSET		
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

[1] The following instructions are not available with the same channel as for the G(P).PRR instruction because they use a communication protocol different from the one used by this instruction.
- G(P).ONDEMAND, G(P).CPRTCL, G(P).BIDOUT, and G(P).BIDIN instructions

- The local device and the file register for each program are not available for setting data.

Operation error

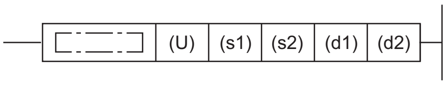
Error code ((s)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

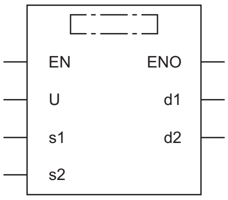
20.10 Clearing Receive Data

ZP.CSET

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

This instruction clears the receive data area without stopping send processing by using the nonprocedural protocol.

Ladder	ST
	ENO:=ZP_CSET(EN,U,s1,s2,d1,d2);

FBD/LD


■Execution condition

Instruction	Execution condition
ZP.CSET	

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	String	ANY16_OR_STRING_SINGLE
(s1)	Channel number of the channel for which the receive data clear request is issued • 1: Channel (CH1 side) • 2: Channel (CH2 side)	1, 2	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)	Dummy	—	Device name	ANY16
(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] 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

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	$	
(U)*1	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(s2)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d1)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d2)	○*2	—	○*3	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution type	Specifies 0.	0	User
+1	Completion status	Used to store the completion status. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Request type	Specifies the contents of request. • 4: Receive data clear request	4	User
+3 to +111	For system	Use prohibited (not available also for other applications such as programming)	—	System

Processing details

- This function only clears receive data from the OS area, but does not clear any data from the user receive area in the buffer memory.
- If the ZP.CSET instruction is issued when the receive read request (Xn3/XnA) or receive error detection (Xn4/XnB) is on, it has to wait until the signal turns off.
- The completion status of the ZP.CSET instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

• Completion device (d2)

This device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

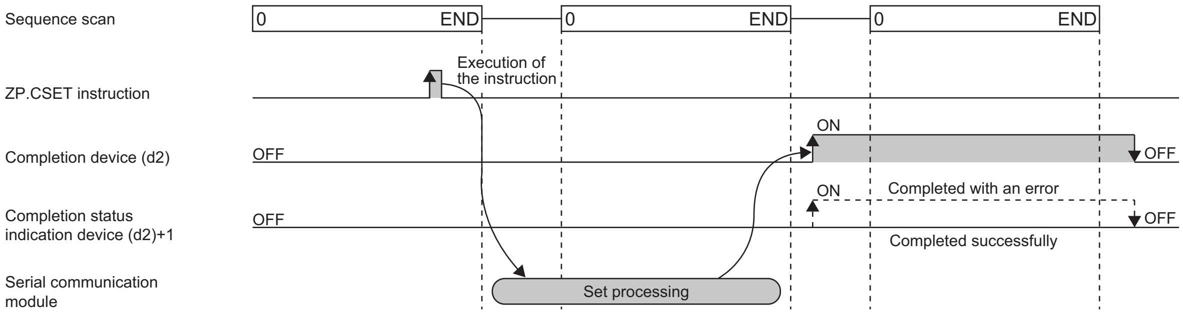
• Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the ZP.CSET instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

- The following figures show how the ZP.CSET instruction operates when its execution completes.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the ZP.CSET instruction is active or the ZP.CSET instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*1]	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT	See the right	• If the G.INPUT and ZP.CSET instructions are issued in this order A dedicated instruction concurrent execution error (7FF0H) occurs in the ZP.CSET instruction. Concurrent execution is, however, possible if they use different channels. • If the ZP.CSET and G.INPUT instructions are issued in this order Concurrent execution is possible.
ZP.CSET	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

*1 The G(P).CPRTCL instruction is not available with the same channel as for the Z(P).CSET instruction because it uses a communication protocol different from the one used by the ZP.CSET instruction. If the above instruction is used with the same channel as for the G(P).CPRTCL instruction, a communication protocol setting error (7FF2H) will occur.

- The local device and the file register for each program are not available for setting data.

Operation error

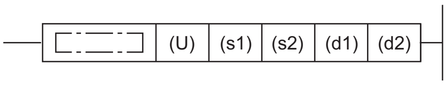
Error code ((s2)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.11 Registering/Canceling the Programmable Controller CPU Monitoring

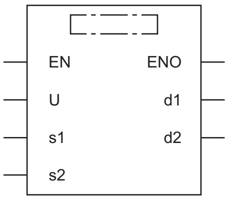
ZP.CSET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

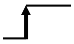
This instruction performs programmable controller CPU monitoring registration to enable use of the programmable controller CPU monitoring function, or cancels it.

Ladder	ST
	ENO:=ZP_CSET(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
ZP.CSET	

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	String	ANY16_OR_STRING_SINGLE
(s1)	Channel number of the channel for which to send the monitoring result • 1: Channel (CH1 side) • 2: Channel (CH2 side)	1, 2	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)	Dummy	—	Device name	ANY16
(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 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

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	$	
(U)^{*1}	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	○^{*2}	—	○^{*3}	—	—	—	—	○	○	—	—	—
(s2)	—	—	○^{*3}	—	—	—	—	○	—	—	—	—
(d1)	—	—	○^{*3}	—	—	—	—	○	—	—	—	—
(d2)	○^{*2}	—	○^{*3}	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.

*2 FX and FY cannot be used.

*3 FD cannot be used.

■Control data

- When programmable controller CPU monitoring is registered

Operand: (s2)

Device	Item		Description	Setting range	Set by
+0	Execution type		Specifies 0.	0	User
+1	Completion status		Used to store the completion status. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Request type		Specifies the contents of request. • 2: Performs programmable controller CPU monitoring registration.	2	User
+3	Unit of cycle time		Specifies the unit of the cycle time. • 0: 100ms • 1: Second • 2: Minute	0 to 2	User
+4	Cycle time		Specifies the cycle time.	1H to FFFFH	User
+5	Programmable controller CPU monitoring function		Specifies the monitoring function. • 1: Sending with a fixed cycle • 2: Sending upon condition matching	1, 2	User
+6	Sending means for programmable controller CPU monitoring		Specifies the sending means. • 0: Sending data (device data, CPU error information)	0	User
+7	Cyclic broadcast send	User frame output start pointer	Specifies the start pointer of the table where the user frame numbers for sending with a fixed cycle have been set. • 0: Not specified (sending upon condition matching) • 1 to 100: Amount of data sent	0, 1 to 100	User
+8		Number of user frames sent	Specifies the number of user frames that are sent, or output, with a fixed cycle. • 0: Not specified (sending upon condition matching) • 1 to 100: Amount of data sent	0, 1 to 100	User
+9	For system		Use prohibited	—	—
+10	Number of reserved word blocks		Specifies the number of blocks of the word device to be monitored.	0 to 10	User
+11	Number of registered bit blocks		Specifies the number of blocks of the bit device to be monitored.	0 to 10	User
+12	Programmable controller CPU error monitoring (programmable controller CPU status monitoring)		Specifies whether to monitor programmable controller CPU errors. • 0: Not monitored • 1: Monitored	0, 1	User
+13	Programmable controller CPU monitoring setting	Device code	Specifies the code of the device to be monitored. • 0: Not monitored • Other than 0: Monitored (code of the device to be monitored)	0 or more	User

Operand: (s2)						
Device	**Item**			**Description**	**Setting range**	**Set by**
+14, +15	1st piece (1st block)	Start device monitored		Specifies the start device monitored of this block.	0 or more	User
+16		Number of registered points		Specifies the number of registered, or read, points of this block. For a bit device, specify the number of points in units of words. • 0: Device not monitored • 1 or more: Number of registered points	0 or more	User
+17		Sending upon condition matching	Monitoring condition	Specifies the monitoring condition for this block. • 0: Not specified (when sending with a fixed cycle) • 1 or more: Monitoring conditions	Refer to the following. 📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)	User
+18			Monitoring condition value	Specifies the monitoring condition value for this block. Specify 0 for sending with a fixed cycle. • 0 or more: Monitoring conditions		User
+19			User frame output start pointer	Specifies the start pointer of the table where the user frame numbers for sending upon condition matching have been set. • 0: Not specified (when sending with a fixed cycle) • 1 to 100: Start pointer	0, 1 to 100	User
+20			Number of user frames sent	Specifies the number of user frames that are sent, or output, upon condition matching. • 0: Not specified (when sending with a fixed cycle) • 1 to 100: Amount of data sent	0, 1 to 100	User
+21	For system			Use prohibited	—	—
+22 to +102	2nd to 10th piece of programmable controller CPU monitoring setting (2nd to 10th block)			Same row as 1st piece of programmable controller CPU monitoring setting	—	User
+103	CPU status monitoring setting (error monitoring 11th piece) (11th block)	Sending upon condition matching	Fixed value	Specifies the fixed value if the CPU status is monitored.	1	User
+104					0	User
+105					0	User
+106					1	User
+107					5	User
+108					1	User
+109			User frame output start pointer	Specifies the start pointer of the table where the user frame numbers for sending upon condition matching have been set. • 0: Not specified (when sending with a fixed cycle) • 1 to 100: Start pointer	0, 1 to 100	User
+110			Number of user frames sent	Specifies the number of user frames that are sent, or output, upon condition matching. • 0: Not specified (when sending with a fixed cycle) • 1 to 100: Amount of data sent	0, 1 to 100	User
+111	For system			Use prohibited	—	—

- When programmable controller CPU monitoring is canceled

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution type	Specifies "0."	0	User
+1	Completion status	Used to store the completion status. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Request type	Specifies the contents of request. • 3: Cancels programmable controller CPU monitoring.	3	User
+3 to +111	For system	Use prohibited (not available also for other applications such as programming)	—	System

Processing details

- To perform programmable controller CPU monitoring registration, register the data that is used for the module to execute the CPU module function. Once the data used to execute the programmable controller CPU monitoring function has completed successfully, the module starts monitoring the programmable controller CPU and sending the send result to the external device.
- To cancel programmable controller CPU monitoring, stop the programmable controller CPU monitoring processing that is being executed by the module. Once programmable controller CPU monitoring has been canceled successfully, the module stops operation of the programmable controller CPU monitoring function.
- If device memory is monitored, a maximum of 10 blocks can be specified for the word and bit devices. To register the device memory to be monitored, specify the word device blocks corresponding to the number of registered word blocks before specifying the bit device blocks corresponding to the registered bit blocks.
- To send the CPU monitoring result to the external device, register the user frames and the user frame numbers using the engineering tool.
- To perform programmable controller CPU monitoring registration again, cancel programmable controller CPU monitoring before registration.
- The completion status of the ZP.CSET instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

This device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the ZP.CSET instruction.

When completed successfully: The device remains off.

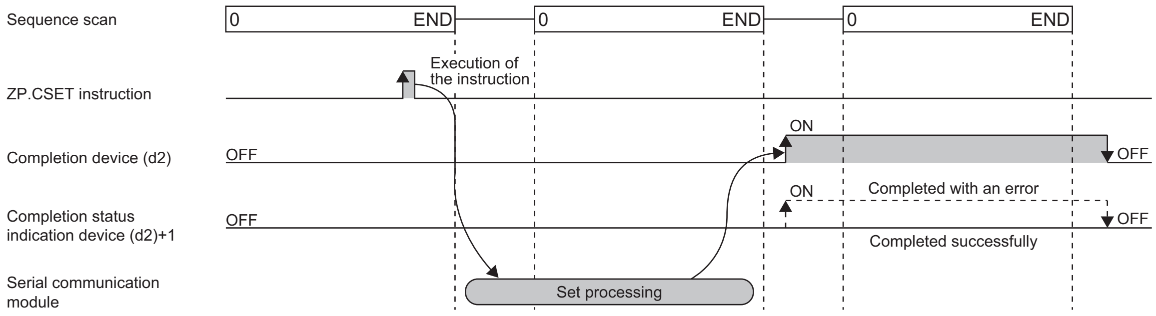
When completed with an error: The device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

- If the programmable controller CPU monitoring registration is to be performed, the following data may be specified in the control data only if the communication protocol setting for the target interface is nonprocedural protocol.

- User frame output start pointer for sending with a fixed cycle/sending upon condition matching
- Number of user frames sent for sending with a fixed cycle/sending upon condition matching

When the communication setting for the target interface is MC protocol, specifying the user frame output start pointer or the number of user frames sent is not required. (They will be ignored if specified).

- The following figures show how the ZP.CSET instruction operates when its execution completes.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the ZP.CSET instruction is active or the ZP.CSET instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[*1]	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT	See the right	• If the G.INPUT and ZP.CSET instructions are issued in this order A dedicated instruction concurrent execution error (7FF0H) occurs in the ZP.CSET instruction. Concurrent execution is, however, possible if they use different channels. • If the ZP.CSET and G.INPUT instructions are issued in this order Concurrent execution is possible.
ZP.CSET	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

[*1] The following instructions are not available with the same channel as for the ZP.CSET instruction because they use a communication protocol different from the one used by this instruction. In addition, if these are used with the same channel as for the ZP.CSET instruction, a communication protocol setting error (7FF2H) will occur.

- G(P).CPRTCL, G(P).BIDOUT, and G(P).BIDIN instructions

- The local device and the file register for each program are not available for setting data.

Operation error

Error code ((s2)+1)	Description
7000H to 7FFFH	🕮 MELSEC iQ-R Serial Communication Module User's Manual (Application)

The programmable controller CPU monitoring settings (control data (s2)+13 to (s2)+102) are checked, not when the CSET instruction is executed, when the specified cycle time has elapsed.

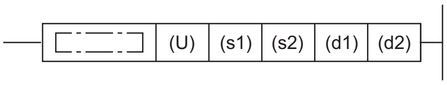
If the CSET instruction is completed successfully and the registered monitoring data is not sent from the serial communication module within the specified cycle time, check the programmable controller CPU monitoring function execution results (buffer memory address: 2205H/2305H) to check for errors and troubleshoot.

20.12 Initial Setting

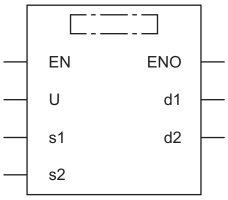
ZP.CSET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

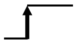
This instruction sets the unit (word or byte) of the amount of send/receive data and the data communication area.

Ladder	ST
	ENO:=ZP_CSET(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
ZP.CSET	

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	String	ANY16_OR_STRING_SINGLE
(s1)	Channel number of the channel for which the set value is changed. • 1: Channel (CH1 side) • 2: Channel (CH2 side)	1, 2	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[1]
(d1)	Dummy	—	Device name	ANY16
(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] 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

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	$	
(U)*1	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	○*2	—	○*3	—	—	—	—	○	○	—	—	—
(s2)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d1)	—	—	○*3	—	—	—	—	○	—	—	—	—
(d2)	○*2	—	○*3	—	—	—	—	—	—	—	—	—

*1 Index modification is not available.
*2 FX and FY cannot be used.
*3 FD cannot be used.

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	Execution type	Specifies 0.	0	User
+1	Completion status	Used to store the completion status. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Request type	Specifies the contents of request. • 1: Changes the unit of data (word or byte) and the buffer memory assignment.	1	User
+3	Word/byte specification	Specifies the unit of the amount of send/receive data. • 0: Current set value • 1: In units of words • 2: In units of bytes	0 to 2	User
+4	Buffer memory start address for on-demand function	Specifies the start address of the buffer memory area to be used for the on-demand function. • 0: The current set value is used. • 400H to 1AFFH, 2600H to 3FFFH: Start address	0H, 400H to 1AFFH, 2600H to 3FFFH	User
+5	Size of buffer memory for on-demand function	Specifies the size (words) of the buffer memory area to be used for the on-demand function. • 0: The current set value is used. • 1H to 1A00H: Size	0H, 1H to 1A00H	User
+6	Sending area start address	Specifies the start address of the sending area to be used for the nonprocedural or bidirectional protocol. • 0: The current set value is used. • 400H to 1AFFH, 2600H to 3FFFH: Start address	0H, 400H to 1AFFH, 2600H to 3FFFH	User
+7	Size of sending area	Specifies the size (words) of the sending area to be used for the nonprocedural or bidirectional protocol. The one word start area of the sending area is used for specifying the amount of send data. • 0: The current set value is used. • 1H to 1A00H: Size	0H, 1H to 1A00H	User
+8	Receiving area start address	Specifies the start address of the receiving area to be used for the nonprocedural or bidirectional protocol. • 0: The current set value is used. • 400H to 1AFFH, 2600H to 3FFFH: Start address	0H, 400H to 1AFFH, 2600H to 3FFFH	User
+9	Size of receiving area	Specifies the size (words) of the receiving area to be used for the nonprocedural or bidirectional protocol. The one word start area of the receiving area is used for specifying the amount of send data. • 0H: The current set value is used. • 1H to 1A00H: Size	0H, 1H to 1A00H	User
+10 to +111	For system	Use prohibited (not available also for other applications such as programming)	—	System

Processing details

- The following current set values are changed, which are used for data communication with each communication protocol.
  - Unit of the amount of the data to be sent or received (word or byte)
  - Sending area of the buffer memory to be used for the MC protocol on-demand function
  - Sending or receiving area of the buffer memory to be used for the nonprocedural protocol
  - Sending or receiving area of the buffer memory to be used for the bidirectional protocol
- To change any of the above set values from the CPU module, execute the ZP.CSET instruction. In addition, execute it one scan before the start of data communication or earlier. After the start of data communication, the ZP.CSET instruction cannot be executed, therefore, the set value cannot be changed. The system does not allow concurrent execution of two or more ZP.CSET instructions set to perform the initial setting.
- The completion status of the ZP.CSET instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.
  - Completion device (d2)

This device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

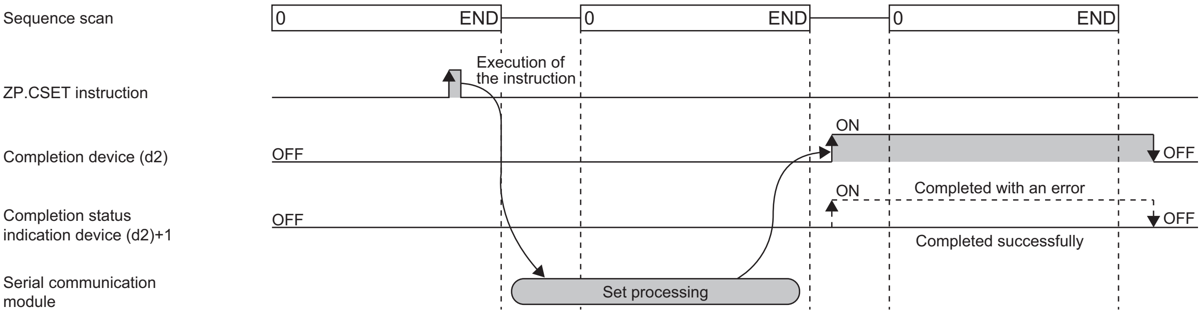
- Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the ZP.CSET instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the ZP.CSET instruction completes, and turns off during the next END processing.

- The following figures show how the ZP.CSET instruction operates when its execution completes.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the ZP.CSET instruction is active or the ZP.CSET instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently[1]	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction. Concurrent execution is, however, possible if they use different channels.
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT	See the right	• If the G.INPUT and ZP.CSET instructions are issued in this order A dedicated instruction concurrent execution error (7FF0H) occurs in the ZP.CSET instruction. Concurrent execution is, however, possible if they use different channels. • If the ZP.CSET and G.INPUT instructions are issued in this order Concurrent execution is possible.
ZP.CSET	×	The instruction is ignored and does not start processing until the active instruction completes. Concurrent execution is, however, possible if they use different channels.
G(P).PUTE	○	—
G(P).GETE		
Z.BUFRCVS		
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.

*1 The G(P).CPRTCL instruction is not available with the same channel as for the Z(P).CSET instruction because it uses a communication protocol different from the one used by the ZP.CSET instruction. If the above instruction is used with the same channel as for the G(P).CPRTCL instruction, a communication protocol setting error (7FF2H) will occur.

- The local device and the file register for each program are not available for setting data.

Operation error

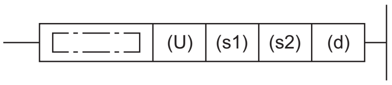
Error code ((s2)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.13 Registering User Frames

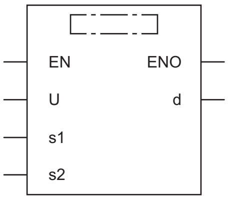
G(P).PUTE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These instructions register user frames.

Ladder	ST
	ENO:=G_PUTE(EN,U,s1,s2,d); ENO:=GP_PUTE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.PUTE	
GP.PUTE	

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	Refer to the control data.	Device name	ANY16[1]
(s2)	Start device for storing the registered data	—	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[1]	—	—	○[3]	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[4]	—	—	—	—	○	—	—	—	—
(d)	○[2]	—	○[5]	—	—	—	—	—	—	—	—	—

[1] Index modification is not available.

[2] FX and FY cannot be used.

[3] RD cannot be used.

[4] FD cannot be used.

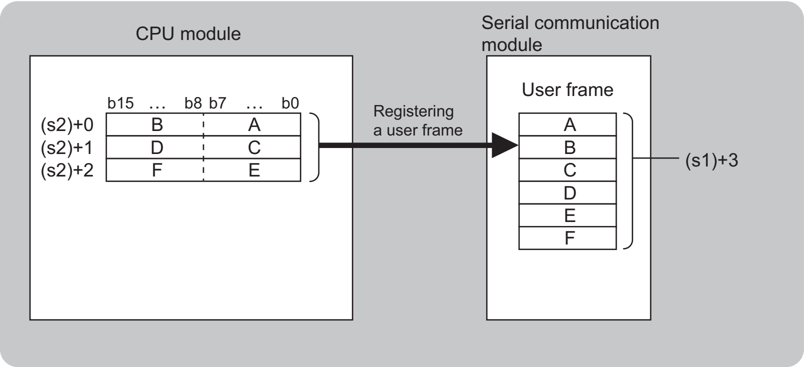
[5] T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Registration/delete specification	Specifies whether to register or delete the user frame specified by (s1)+2. • 1: Registered • 3: Deleted	1, 3	User
+1	Registration/delete result	Used to store the result of registration or deletion with the G(P).PUTE instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Frame number	Specifies the user frame number.	1000 to 1199	User
+3	Number of registered bytes	• 1 to 80: Number of bytes of the user frame to be registered To delete the user frame, specify a value from 1 to 80 as the dummy.	1 to 80	User

Processing details

- User frames are registered or deleted for the module specified by (U).
- If a user frame is registered
  - To register a user frame, specify 1 for the device of (s1)+0. The data in the device specified by (s2) and later will be registered according to the control data.
  - The registered data must be stored as follows in the device specified by (s2) and later. Each of the devices to store the registered data requires the data for the following number of points from the device specified by (s2): (amount of registered data)/2.



- If a user frame is deleted
  - To delete a user frame, specify "3" for the device of (s1)+0. The user frame with the frame number specified by (s1)+2 will be deleted.
  - Both the number of registered bytes specified by (s1)+3 and the registered data storage device are required for the G(P).PUTE instruction format although they are not used by the G(P).PUTE instruction. Specify a value from 1 to 80 for (s1)+3. Specify a dummy device for (s2).
- The completion status of the G(P).PUTE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the G(P).PUTE instruction completes, and turns off during the next END processing.

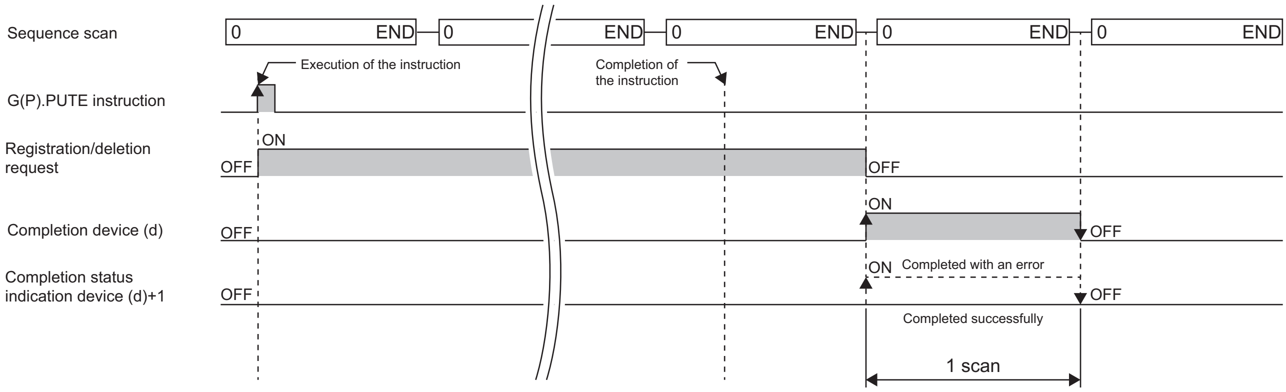
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).PUTE instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).PUTE instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).PUTE instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).PUTE instruction is active or the G(P).PUTE instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	○	—
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT		
ZP.CSET		
G(P).PUTE	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.
G(P).GETE	×	The instruction is ignored and does not start processing until the active instruction completes.
Z.BUFRCVS	○	—
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.
G(P).CPRTCL	○	—

Operation error

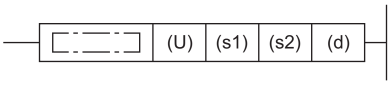
Error code ((s1)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

20.14 Reading User Frames

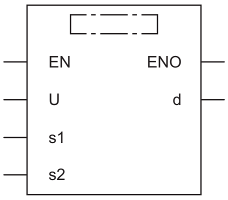
G(P).GETE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

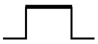
These instructions read user frames.

Ladder	ST
	ENO:=G_GETE(EN,U,s1,s2,d); ENO:=GP_GETE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.GETE	
GP.GETE	

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	Refer to the control data.	Device name	ANY16[*1]
(s2)	Start device for storing the registered data that has been read.	—	Device name	ANY16[*1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[*1]	—	—	○[*3]	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[*4]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[*4]	—	—	—	—	○	—	—	—	—
(d)	○[*2]	—	○[*5]	—	—	—	—	—	—	—	—	—

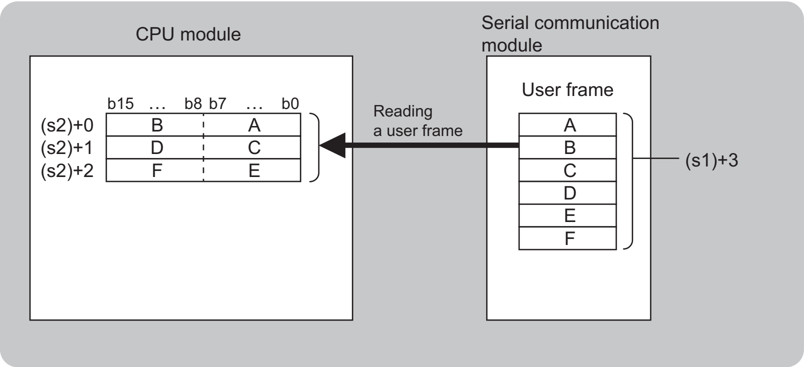
[*1] Index modification is not available.
[*2] FX and FY cannot be used.
[*3] RD cannot be used.
[*4] FD cannot be used.
[*5] T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Dummy	—	0	—
+1	Result of reading	Used to store the result of sending with the G(P).GETE instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Specified frame number	Specifies the user-registered number.	1000 to 1199	User
+3	Allowable number of bytes	Specifies the number of bytes storable in (s2), of the registered data of the read user frame.	1 to 80	User
	Number of registered bytes	Used to store the number of bytes of the registered data of the read user frame. • 1 to 80: Number of bytes of registered data	—	System

Processing details

- The data of the user frame is read from the module specified by (U).



- The completion status of the G(P).GETE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.
  - Completion device (d)

This device turns on during END processing of the scan where the G(P).GETE instruction completes, and turns off during the next END processing.

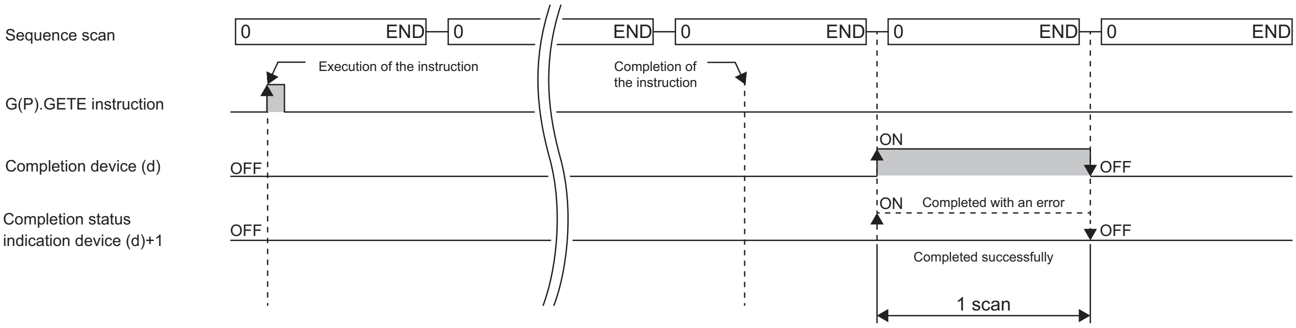
  - Completion status indication device (d)+1

This device turns on or off depending on the completion status of the G(P).GETE instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the G(P).GETE instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).GETE instruction operates when completing its execution.



Precautions

- The following table summarizes the processes that take place if another instruction is issued while the G(P).GETE instruction is active or the G(P).GETE instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	○	—
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT		
ZP.CSET		
G(P).PUTE	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.
G(P).GETE	×	The instruction is ignored and does not start processing until the active instruction completes.
Z.BUFRCVS	○	—
G(P).SPBUSY		
ZP.UINI	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.
G(P).CPRTCL	○	—

Operation error

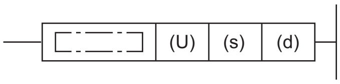
Error code ((s1)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

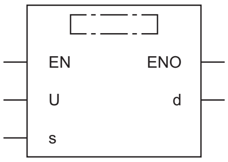
20.15 Switching the Mode

ZP.UINI

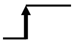
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

This instruction changes the communication protocol, transmission setting, or station number.

Ladder	ST
	ENO:=ZP_UINI(EN,U,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
ZP.UINI	

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	String	ANY16_OR_STRING_SINGLE
(s)	Start device where control data is stored	Refer to the control data.	Device name	ANY16[*1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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

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	$	
(U)[*1]	—	—	○	—	—	—	—	○	—	—	○	○
(s)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(d)	○[*2]	—	○[*4]	—	—	—	—	—	—	—	—	—

[*1] Index modification is not available.
[*2] FX and FY cannot be used.
[*3] FD cannot be used.
[*4] T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	For system	Always specifies 0.	0	User
+1	Execution result	Used to store the result of sending with the ZP.UINI instruction. • 0: Normal • Other than 0: Error (error code)	—	System
+2	Execution type	Specify the execution type. • 0: Change the settings according to the setting details indicated by (s)+3 and later. • 1: Restores the setting details of the user parameters that have been set in the engineering tool.	0, 1	User
+3	CH1 transmission specifications setting	Sets the CH1 side transmission setting. Check the current value of the buffer memory area (595 (253H)) to make the settings identical.	0 to 4095 (0000H to 0FFFH)	User
+4	CH1 communication protocol setting	Sets the CH1 side communication protocol. Check the current value of the buffer memory area (594(252H)) to make the settings identical.	0 to 9	User
+5	CH2 transmission specifications setting	Sets the CH2 side transmission setting. Check the current value of the buffer memory area (611(263H)) to make the settings identical.	0 to 4095 (0000H to 0FFFH)	User
+6	CH2 communication protocol setting	Sets the CH2 side communication protocol. Check the current value of the buffer memory area (610(262H)) to make the settings identical.	0 to 9	User
+7	Station number setting	Sets the station number.	0 to 31	User
+8 to +12	For system	Always specifies 0.	0	User

Processing details

- This instruction changes the communication protocol, transmission setting, or station number of each channel of the module specified by (U).
- The completion status of the ZP.UINI instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the ZP.UINI instruction completes, and turns off during the next END processing.

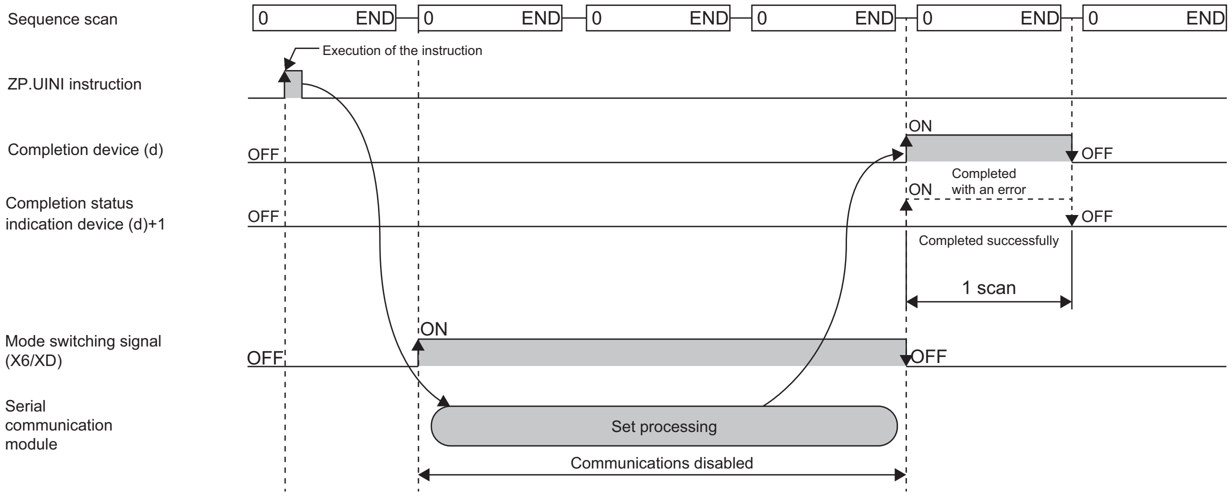
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the ZP.UINI instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the ZP.UINI instruction completes, and turns off during the next END processing.

- The following figures show how the ZP.UINI instruction operates when its execution completes.



Precautions

- Stop the whole data communication with the external device before issuing the ZP.UINI instruction. If the ZP.UINI instruction is issued while data communication is in progress with the external device, the following will occur.
  - If the ZP.UINI instruction is issued during receiving

Receiving is interrupted, and the whole receive data stored so far is discarded.

  - If the ZP.UINI instruction is issued during sending

When the ZP.UINI instruction is accepted, sending is interrupted.

- If connection is provided through a modem, cut off the line to the external device before issuing the ZP.UINI instruction. If the ZP.UINI instruction is issued, the line will be cut off.
- If the communication protocol or transmission setting has been changed using the ZP.UINI instruction, the communication protocol and transmission settings of the external device side must be matched with the new settings. In addition, if the station number setting, or own station number, has been changed, the station number in the request message of the external device must be modified to the new station number.
- The following table summarizes the processes that take place if another instruction is issued while the ZP.UINI instruction is active or the ZP.UINI instruction is issued while another instruction is active with the same channel.

Instruction to execute concurrently	Possibility of concurrent execution	Handling for concurrent execution
G(P).ONDEMAND	×	A dedicated instruction concurrent execution error (7FF0H) occurs in the second instruction.
G(P).OUTPUT		
G(P).PRR		
G(P).BIDOUT		
G(P).BIDIN		
G.INPUT		
ZP.CSET		
G(P).PUTE		
G(P).GETE		
G(P).CPRTCL		
Z.BUFRCVS	○	—
G(P).SPBUSY		
ZP.UINI	×	The instruction is ignored and does not start processing until the active instruction completes.

- Before the ZP.UINI instruction is executed, the setting change in the transmission settings must be kept set to "Permitted" by using the parameter setting of the module. In addition, if the parameter is not yet set, the setting change in the transmission settings will be assumed to be "Permitted" during operation.
- Simultaneous use of mode change with the ZP.UINI instruction and that with the mode change request signal (Yn2/Yn9) must not be attempted. If it is attempted, an error will occur at an error code of 7FF5H.
- To use the current value of the communication protocol, transmission specifications, or station number as it is, acquire the value from each of the status storage areas in buffer memory before setting it in the control data.

Address (decimal (hexadecimal))		Application
CH1	CH2	
591(24FH)		Station number (instruction setting)
594(252H)	610(262H)	Communication protocol status
595(253H)	611(263H)	Transmission status

Operation error

Error code ((s)+1)	Description
7000H to 7FFFH	📖 MELSEC iQ-R Serial Communication Module User's Manual (Application)

21 A/D CONVERSION INSTRUCTIONS

Point

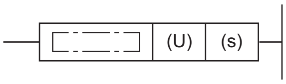
This chapter describes the instructions used commonly by MELSEC iQ-R series modules. When using MELSEC-Q series modules, refer to the manual for each module used and create programs.
For precautions when using modules, refer to the following.
📖 MELSEC iQ-R Module Configuration Manual

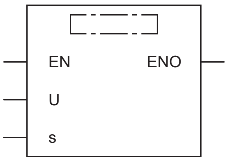
21.1 Switching the Mode

G(P).OFFGAN

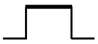
RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These instructions switch the analog module mode.

Ladder	ST
	ENO:=G_OFFGAN(EN,U,s); ENO:=GP_OFFGAN(EN,U,s);

FBD/LD


■Execution condition

Instruction	Execution condition
G.OFFGAN	
GP.OFFGAN	

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 • 16-bit signed binary	ANY16
(s)	Switching the mode • 0: Shift to normal mode (normal output mode)[1] • 1: Shift to offset/gain setting mode Setting a value outside the above range results in "Shift to offset/gain setting mode".	0, 1	• 16-bit unsigned binary • 16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] The mode name differs depending on the module used: normal output mode for the R60DA4, R60DAV8, R60DAI8, and R60DAH4 and normal mode for other analog modules.

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s)	—	—	○[1]	—	—	—	—	○	—	—	—	—

[1] FD cannot be used.

Processing details

These instructions switch the mode of an analog module.

- Normal mode (Normal output mode) → Offset/gain setting mode (The offset/gain setting mode status flag (XA) turns on.)
- Offset/gain setting mode → Normal mode (Normal output mode) (The offset/gain setting mode status flag (XA) turns off.)

Point

- When the mode shifts from normal or normal output to offset/gain setting, Module Ready (X0) changes on to off.
- When the mode shifts from offset/gain setting to normal or normal output, Module Ready (X0) changes off to on. If a program exists that performs initial setting with Module Ready (X0) on, note that initial setting processing will be executed.
- When the mode shifts from offset/gain setting to normal on the A/D converter module and temperature input module, it automatically resumes the operation with the previous operating conditions.
- When the D/A converter module shifts from the offset/gain setting mode to the normal mode (normal output mode), all channels enter in the D/A conversion disabled state. To resume D/A conversion, set "0" (D/A conversion enabled) in "CH□D/A conversion enabled/disabled setting" of the appropriate channel before turning on and off Operating condition setting request (Y9).

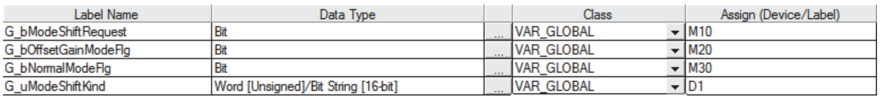
■Program example

The following figure shows an example of a program which uses the G(P).OFFGAN instruction with the R60AD4.

- Program content

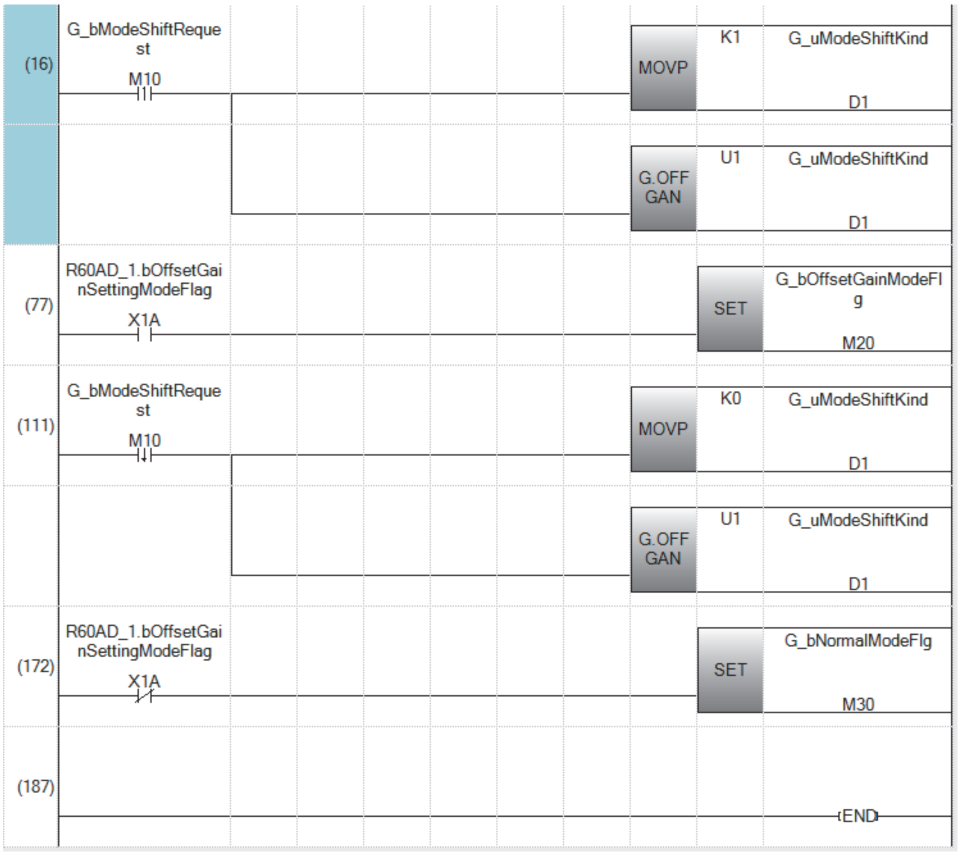
Description
When "Mode Shift Request" (M10) is turned on, the A/D converter module mounted at the position of I/O numbers X/Y10 to X/Y1F shifts to the offset/gain setting mode.
When "Mode Shift Request" (M10) is turned off, the module returns to the normal mode.

- Label setting

Classification	Label name	Description	Device
Module label	R60AD_1.bOffsetGainSettingModeFlag	Offset/gain setting mode status flag	X1A
Label to be defined	Define the global label as shown below. 		

Label Name	Data Type		Class	Assign (Device/Label)
G_bModeShiftRequest	Bit	...	VAR_GLOBAL	M10
G_bOffsetGainModeFlg	Bit	...	VAR_GLOBAL	M20
G_bNormalModeFlg	Bit	...	VAR_GLOBAL	M30
G_uModeShiftKind	Word [Unsigned]/Bit String [16-bit]	...	VAR_GLOBAL	D1

- Program example



(16) Shifts to offset/gain setting mode.
(77) Describes the offset/gain setting processing.
(111) Shifts to normal mode.
(172) Describes the normal mode processing.

Precautions

In the following cases, the G(P).OFFGAN instruction will be disabled.

- The module has been set as a target of synchronization.
- The simultaneous conversion mode has been selected on the high speed analog-digital converter module.
- The wave output mode has been selected on the D/A converter module.
- The high-speed output mode has been selected on the high speed digital-analog converter module.

Operation error

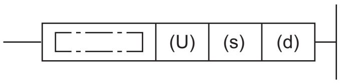
There is no operation error.

21.2 Reading the User Range Setting Values

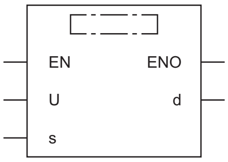
G(P).OGLOAD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

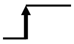
These instructions read the offset/gain setting values of the user range settings of an analog module into the CPU module.

Ladder	ST
	ENO:=G_OGLOAD(EN,U,s,d); ENO:=GP_OGLOAD(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.OGLOAD	
GP.OGLOAD	

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 • 16-bit signed binary	ANY16
(s)	Start device where the control data is stored	Predefined devices	Device name	ANY16[*1]
(d)	Device to be turned on one scan when instruction processing completes. If the instruction is completed with an error, (d)+1 is also turned on.	Predefined devices	Bit	ANYBIT_ARRAY[*2] (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

*2 When specifying setting data by using a label, use an array with two or more elements.

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s)	—	—	○[*1]	—	—	—	—	○	—	—	—	—
(d)	○[*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

- For R60AD4, R60ADH4

Set only the save data type setting (s)+2. A data write to an area to be set by the system does not result in a normal read of the offset/gain setting values.

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Specifies which one of the voltage and current is read as the offset/gain setting value. • 0: Voltage specification • 1: Current specification b15 … b3 b2 b1 b0 0 CH4 CH3 CH2 CH1	0000H to 000FH	User
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH1 User range setting offset value (L)	—	—	System
+21	CH1 User range setting offset value (H)	—	—	System
+22	CH1 User range setting gain value (L)	—	—	System
+23	CH1 User range setting gain value (H)	—	—	System
+24	CH2 User range setting offset value (L)	—	—	System
+25	CH2 User range setting offset value (H)	—	—	System
+26	CH2 User range setting gain value (L)	—	—	System
+27	CH2 User range setting gain value (H)	—	—	System
+28	CH3 User range setting offset value (L)	—	—	System
+29	CH3 User range setting offset value (H)	—	—	System
+30	CH3 User range setting gain value (L)	—	—	System
+31	CH3 User range setting gain value (H)	—	—	System
+32	CH4 User range setting offset value (L)	—	—	System
+33	CH4 User range setting offset value (H)	—	—	System
+34	CH4 User range setting gain value (L)	—	—	System
+35	CH4 User range setting gain value (H)	—	—	System

- For R60DA4, R60DAH4

Set only the save data type setting (s)+2. A data write to an area to be set by the system does not result in a normal read of the offset/gain setting values.

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Specify the user range setting for reading offset/gain setting values. • 0: User range setting (voltage specification) • 1: User range setting (current specification) b15 … b3 b2 b1 b0 0 | CH4 | CH3 | CH2 | CH1	0000H to 000FH	User
+3	System area	—	—	—
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH1 User range setting offset value	—	—	System
+13	CH1 User range setting gain value	—	—	System
+14	CH2 User range setting offset value	—	—	System
+15	CH2 User range setting gain value	—	—	System
+16	CH3 User range setting offset value	—	—	System
+17	CH3 User range setting gain value	—	—	System
+18	CH4 User range setting offset value	—	—	System
+19	CH4 User range setting gain value	—	—	System

- For R60ADV8, R60ADI8, R60AD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting[*1]	Specifies which one of the voltage and current is read as the offset/gain setting value. • 0: Voltage specification • 1: Current specification b15 … b7 b6 b5 b4 b3 b2 b1 b0 0 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to 00FFH	User
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH5 Factory default setting offset value (L)	—	—	System
+21	CH5 Factory default setting offset value (H)	—	—	System
+22	CH5 Factory default setting gain value (L)	—	—	System
+23	CH5 Factory default setting gain value (H)	—	—	System
+24	CH6 Factory default setting offset value (L)	—	—	System
+25	CH6 Factory default setting offset value (H)	—	—	System
+26	CH6 Factory default setting gain value (L)	—	—	System
+27	CH6 Factory default setting gain value (H)	—	—	System
+28	CH7 Factory default setting offset value (L)	—	—	System
+29	CH7 Factory default setting offset value (H)	—	—	System
+30	CH7 Factory default setting gain value (L)	—	—	System
+31	CH7 Factory default setting gain value (H)	—	—	System
+32	CH8 Factory default setting offset value (L)	—	—	System
+33	CH8 Factory default setting offset value (H)	—	—	System
+34	CH8 Factory default setting gain value (L)	—	—	System
+35	CH8 Factory default setting gain value (H)	—	—	System
+36	CH1 User range setting offset value (L)	—	—	System
+37	CH1 User range setting offset value (H)	—	—	System
+38	CH1 User range setting gain value (L)	—	—	System
+39	CH1 User range setting gain value (H)	—	—	System
+40	CH2 User range setting offset value (L)	—	—	System
+41	CH2 User range setting offset value (H)	—	—	System
+42	CH2 User range setting gain value (L)	—	—	System
+43	CH2 User range setting gain value (H)	—	—	System
+44	CH3 User range setting offset value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+45	CH3 User range setting offset value (H)	—	—	System
+46	CH3 User range setting gain value (L)	—	—	System
+47	CH3 User range setting gain value (H)	—	—	System
+48	CH4 User range setting offset value (L)	—	—	System
+49	CH4 User range setting offset value (H)	—	—	System
+50	CH4 User range setting gain value (L)	—	—	System
+51	CH4 User range setting gain value (H)	—	—	System
+52	CH5 User range setting offset value (L)	—	—	System
+53	CH5 User range setting offset value (H)	—	—	System
+54	CH5 User range setting gain value (L)	—	—	System
+55	CH5 User range setting gain value (H)	—	—	System
+56	CH6 User range setting offset value (L)	—	—	System
+57	CH6 User range setting offset value (H)	—	—	System
+58	CH6 User range setting gain value (L)	—	—	System
+59	CH6 User range setting gain value (H)	—	—	System
+60	CH7 User range setting offset value (L)	—	—	System
+61	CH7 User range setting offset value (H)	—	—	System
+62	CH7 User range setting gain value (L)	—	—	System
+63	CH7 User range setting gain value (H)	—	—	System
+64	CH8 User range setting offset value (L)	—	—	System
+65	CH8 User range setting offset value (H)	—	—	System
+66	CH8 User range setting gain value (L)	—	—	System
+67	CH8 User range setting gain value (H)	—	—	System

*1 For the R60ADV8 and the R60ADI8, this area corresponds to System area and so is not available.

- For R60DAV8, R60DAI8, R60DA8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting[*2]	Specify the user range setting for reading offset/gain setting values. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to AAAAH	User
+3	System area	—	—	—
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH5 Factory default setting offset value	—	—	System
+13	CH5 Factory default setting gain value	—	—	System
+14	CH6 Factory default setting offset value	—	—	System
+15	CH6 Factory default setting gain value	—	—	System
+16	CH7 Factory default setting offset value	—	—	System
+17	CH7 Factory default setting gain value	—	—	System
+18	CH8 Factory default setting offset value	—	—	System
+19	CH8 Factory default setting gain value	—	—	System
+20	CH1 User range setting offset value	—	—	System
+21	CH1 User range setting gain value	—	—	System
+22	CH2 User range setting offset value	—	—	System
+23	CH2 User range setting gain value	—	—	System
+24	CH3 User range setting offset value	—	—	System
+25	CH3 User range setting gain value	—	—	System
+26	CH4 User range setting offset value	—	—	System
+27	CH4 User range setting gain value	—	—	System
+28	CH5 User range setting offset value	—	—	System
+29	CH5 User range setting gain value	—	—	System
+30	CH6 User range setting offset value	—	—	System
+31	CH6 User range setting gain value	—	—	System
+32	CH7 User range setting offset value	—	—	System
+33	CH7 User range setting gain value	—	—	System
+34	CH8 User range setting offset value	—	—	System
+35	CH8 User range setting gain value	—	—	System
+36	System area	—	—	—

[*2] For the R60DAV8 and the R60DAI8, this area corresponds to System area and so is not available.

- For R60TD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	System area	—	—	—
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH1 User range setting offset value (L)	—	—	System
+9	CH1 User range setting offset value (H)	—	—	System
+10	CH1 User range setting gain value (L)	—	—	System
+11	CH1 User range setting gain value (H)	—	—	System
+12	CH1 User range setting thermoelectromotive force offset value (L)	—	—	System
+13	CH1 User range setting thermoelectromotive force offset value (H)	—	—	System
+14	CH1 User range setting thermoelectromotive force gain value (L)	—	—	System
+15	CH1 User range setting thermoelectromotive force gain value (H)	—	—	System
+16	CH2 Factory default setting offset value (L)	—	—	System
+17	CH2 Factory default setting offset value (H)	—	—	System
+18	CH2 Factory default setting gain value (L)	—	—	System
+19	CH2 Factory default setting gain value (H)	—	—	System
+20	CH2 User range setting offset value (L)	—	—	System
+21	CH2 User range setting offset value (H)	—	—	System
+22	CH2 User range setting gain value (L)	—	—	System
+23	CH2 User range setting gain value (H)	—	—	System
+24	CH2 User range setting thermoelectromotive force offset value (L)	—	—	System
+25	CH2 User range setting thermoelectromotive force offset value (H)	—	—	System
+26	CH2 User range setting thermoelectromotive force gain value (L)	—	—	System
+27	CH2 User range setting thermoelectromotive force gain value (H)	—	—	System
+28	CH3 Factory default setting offset value (L)	—	—	System
+29	CH3 Factory default setting offset value (H)	—	—	System
+30	CH3 Factory default setting gain value (L)	—	—	System
+31	CH3 Factory default setting gain value (H)	—	—	System
+32	CH3 User range setting offset value (L)	—	—	System
+33	CH3 User range setting offset value (H)	—	—	System
+34	CH3 User range setting gain value (L)	—	—	System
+35	CH3 User range setting gain value (H)	—	—	System
+36	CH3 User range setting thermoelectromotive force offset value (L)	—	—	System
+37	CH3 User range setting thermoelectromotive force offset value (H)	—	—	System
+38	CH3 User range setting thermoelectromotive force gain value (L)	—	—	System
+39	CH3 User range setting thermoelectromotive force gain value (H)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+40	CH4 Factory default setting offset value (L)	—	—	System
+41	CH4 Factory default setting offset value (H)	—	—	System
+42	CH4 Factory default setting gain value (L)	—	—	System
+43	CH4 Factory default setting gain value (H)	—	—	System
+44	CH4 User range setting offset value (L)	—	—	System
+45	CH4 User range setting offset value (H)	—	—	System
+46	CH4 User range setting gain value (L)	—	—	System
+47	CH4 User range setting gain value (H)	—	—	System
+48	CH4 User range setting thermoelectromotive force offset value (L)	—	—	System
+49	CH4 User range setting thermoelectromotive force offset value (H)	—	—	System
+50	CH4 User range setting thermoelectromotive force gain value (L)	—	—	System
+51	CH4 User range setting thermoelectromotive force gain value (H)	—	—	System
+52	CH5 Factory default setting offset value (L)	—	—	System
+53	CH5 Factory default setting offset value (H)	—	—	System
+54	CH5 Factory default setting gain value (L)	—	—	System
+55	CH5 Factory default setting gain value (H)	—	—	System
+56	CH5 User range setting offset value (L)	—	—	System
+57	CH5 User range setting offset value (H)	—	—	System
+58	CH5 User range setting gain value (L)	—	—	System
+59	CH5 User range setting gain value (H)	—	—	System
+60	CH5 User range setting thermoelectromotive force offset value (L)	—	—	System
+61	CH5 User range setting thermoelectromotive force offset value (H)	—	—	System
+62	CH5 User range setting thermoelectromotive force gain value (L)	—	—	System
+63	CH5 User range setting thermoelectromotive force gain value (H)	—	—	System
+64	CH6 Factory default setting offset value (L)	—	—	System
+65	CH6 Factory default setting offset value (H)	—	—	System
+66	CH6 Factory default setting gain value (L)	—	—	System
+67	CH6 Factory default setting gain value (H)	—	—	System
+68	CH6 User range setting offset value (L)	—	—	System
+69	CH6 User range setting offset value (H)	—	—	System
+70	CH6 User range setting gain value (L)	—	—	System
+71	CH6 User range setting gain value (H)	—	—	System
+72	CH6 User range setting thermoelectromotive force offset value (L)	—	—	System
+73	CH6 User range setting thermoelectromotive force offset value (H)	—	—	System
+74	CH6 User range setting thermoelectromotive force gain value (L)	—	—	System
+75	CH6 User range setting thermoelectromotive force gain value (H)	—	—	System
+76	CH7 Factory default setting offset value (L)	—	—	System
+77	CH7 Factory default setting offset value (H)	—	—	System
+78	CH7 Factory default setting gain value (L)	—	—	System
+79	CH7 Factory default setting gain value (H)	—	—	System
+80	CH7 User range setting offset value (L)	—	—	System
+81	CH7 User range setting offset value (H)	—	—	System
+82	CH7 User range setting gain value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+83	CH7 User range setting gain value (H)	—	—	System
+84	CH7 User range setting thermoelectromotive force offset value (L)	—	—	System
+85	CH7 User range setting thermoelectromotive force offset value (H)	—	—	System
+86	CH7 User range setting thermoelectromotive force gain value (L)	—	—	System
+87	CH7 User range setting thermoelectromotive force gain value (H)	—	—	System
+88	CH8 Factory default setting offset value (L)	—	—	System
+89	CH8 Factory default setting offset value (H)	—	—	System
+90	CH8 Factory default setting gain value (L)	—	—	System
+91	CH8 Factory default setting gain value (H)	—	—	System
+92	CH8 User range setting offset value (L)	—	—	System
+93	CH8 User range setting offset value (H)	—	—	System
+94	CH8 User range setting gain value (L)	—	—	System
+95	CH8 User range setting gain value (H)	—	—	System
+96	CH8 User range setting thermoelectromotive force offset value (L)	—	—	System
+97	CH8 User range setting thermoelectromotive force offset value (H)	—	—	System
+98	CH8 User range setting thermoelectromotive force gain value (L)	—	—	System
+99	CH8 User range setting thermoelectromotive force gain value (H)	—	—	System

- For R60RD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	System area	—	—	—
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH1 User range setting offset value (L)	—	—	System
+9	CH1 User range setting offset value (H)	—	—	System
+10	CH1 User range setting gain value (L)	—	—	System
+11	CH1 User range setting gain value (H)	—	—	System
+12	CH1 User range setting offset resistance value (L)	—	—	System
+13	CH1 User range setting offset resistance value (H)	—	—	System
+14	CH1 User range setting gain resistance value (L)	—	—	System
+15	CH1 User range setting gain resistance value (H)	—	—	System
+16	CH2 Factory default setting offset value (L)	—	—	System
+17	CH2 Factory default setting offset value (H)	—	—	System
+18	CH2 Factory default setting gain value (L)	—	—	System
+19	CH2 Factory default setting gain value (H)	—	—	System
+20	CH2 User range setting offset value (L)	—	—	System
+21	CH2 User range setting offset value (H)	—	—	System
+22	CH2 User range setting gain value (L)	—	—	System
+23	CH2 User range setting gain value (H)	—	—	System
+24	CH2 User range setting offset resistance value (L)	—	—	System
+25	CH2 User range setting offset resistance value (H)	—	—	System
+26	CH2 User range setting gain resistance value (L)	—	—	System
+27	CH2 User range setting gain resistance value (H)	—	—	System
+28	CH3 Factory default setting offset value (L)	—	—	System
+29	CH3 Factory default setting offset value (H)	—	—	System
+30	CH3 Factory default setting gain value (L)	—	—	System
+31	CH3 Factory default setting gain value (H)	—	—	System
+32	CH3 User range setting offset value (L)	—	—	System
+33	CH3 User range setting offset value (H)	—	—	System
+34	CH3 User range setting gain value (L)	—	—	System
+35	CH3 User range setting gain value (H)	—	—	System
+36	CH3 User range setting offset resistance value (L)	—	—	System
+37	CH3 User range setting offset resistance value (H)	—	—	System
+38	CH3 User range setting gain resistance value (L)	—	—	System
+39	CH3 User range setting gain resistance value (H)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+40	CH4 Factory default setting offset value (L)	—	—	System
+41	CH4 Factory default setting offset value (H)	—	—	System
+42	CH4 Factory default setting gain value (L)	—	—	System
+43	CH4 Factory default setting gain value (H)	—	—	System
+44	CH4 User range setting offset value (L)	—	—	System
+45	CH4 User range setting offset value (H)	—	—	System
+46	CH4 User range setting gain value (L)	—	—	System
+47	CH4 User range setting gain value (H)	—	—	System
+48	CH4 User range setting offset resistance value (L)	—	—	System
+49	CH4 User range setting offset resistance value (H)	—	—	System
+50	CH4 User range setting gain resistance value (L)	—	—	System
+51	CH4 User range setting gain resistance value (H)	—	—	System
+52	CH5 Factory default setting offset value (L)	—	—	System
+53	CH5 Factory default setting offset value (H)	—	—	System
+54	CH5 Factory default setting gain value (L)	—	—	System
+55	CH5 Factory default setting gain value (H)	—	—	System
+56	CH5 User range setting offset value (L)	—	—	System
+57	CH5 User range setting offset value (H)	—	—	System
+58	CH5 User range setting gain value (L)	—	—	System
+59	CH5 User range setting gain value (H)	—	—	System
+60	CH5 User range setting offset resistance value (L)	—	—	System
+61	CH5 User range setting offset resistance value (H)	—	—	System
+62	CH5 User range setting gain resistance value (L)	—	—	System
+63	CH5 User range setting gain resistance value (H)	—	—	System
+64	CH6 Factory default setting offset value (L)	—	—	System
+65	CH6 Factory default setting offset value (H)	—	—	System
+66	CH6 Factory default setting gain value (L)	—	—	System
+67	CH6 Factory default setting gain value (H)	—	—	System
+68	CH6 User range setting offset value (L)	—	—	System
+69	CH6 User range setting offset value (H)	—	—	System
+70	CH6 User range setting gain value (L)	—	—	System
+71	CH6 User range setting gain value (H)	—	—	System
+72	CH6 User range setting offset resistance value (L)	—	—	System
+73	CH6 User range setting offset resistance value (H)	—	—	System
+74	CH6 User range setting gain resistance value (L)	—	—	System
+75	CH6 User range setting gain resistance value (H)	—	—	System
+76	CH7 Factory default setting offset value (L)	—	—	System
+77	CH7 Factory default setting offset value (H)	—	—	System
+78	CH7 Factory default setting gain value (L)	—	—	System
+79	CH7 Factory default setting gain value (H)	—	—	System
+80	CH7 User range setting offset value (L)	—	—	System
+81	CH7 User range setting offset value (H)	—	—	System
+82	CH7 User range setting gain value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+83	CH7 User range setting gain value (H)	—	—	System
+84	CH7 User range setting offset resistance value (L)	—	—	System
+85	CH7 User range setting offset resistance value (H)	—	—	System
+86	CH7 User range setting gain resistance value (L)	—	—	System
+87	CH7 User range setting gain resistance value (H)	—	—	System
+88	CH8 Factory default setting offset value (L)	—	—	System
+89	CH8 Factory default setting offset value (H)	—	—	System
+90	CH8 Factory default setting gain value (L)	—	—	System
+91	CH8 Factory default setting gain value (H)	—	—	System
+92	CH8 User range setting offset value (L)	—	—	System
+93	CH8 User range setting offset value (H)	—	—	System
+94	CH8 User range setting gain value (L)	—	—	System
+95	CH8 User range setting gain value (H)	—	—	System
+96	CH8 User range setting offset resistance value (L)	—	—	System
+97	CH8 User range setting offset resistance value (H)	—	—	System
+98	CH8 User range setting gain resistance value (L)	—	—	System
+99	CH8 User range setting gain resistance value (H)	—	—	System

- For R60AD16-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Specifies which one of the voltage and current is read as the offset/gain setting value. • 0: Voltage specification • 1: Current specification b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH16 CH15 CH14 CH13 CH12 CH11 CH10 CH9 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to FFFFH	User
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH5 Factory default setting offset value (L)	—	—	System
+21	CH5 Factory default setting offset value (H)	—	—	System
+22	CH5 Factory default setting gain value (L)	—	—	System
+23	CH5 Factory default setting gain value (H)	—	—	System
+24	CH6 Factory default setting offset value (L)	—	—	System
+25	CH6 Factory default setting offset value (H)	—	—	System
+26	CH6 Factory default setting gain value (L)	—	—	System
+27	CH6 Factory default setting gain value (H)	—	—	System
+28	CH7 Factory default setting offset value (L)	—	—	System
+29	CH7 Factory default setting offset value (H)	—	—	System
+30	CH7 Factory default setting gain value (L)	—	—	System
+31	CH7 Factory default setting gain value (H)	—	—	System
+32	CH8 Factory default setting offset value (L)	—	—	System
+33	CH8 Factory default setting offset value (H)	—	—	System
+34	CH8 Factory default setting gain value (L)	—	—	System
+35	CH8 Factory default setting gain value (H)	—	—	System
+36	CH9 Factory default setting offset value (L)	—	—	System
+37	CH9 Factory default setting offset value (H)	—	—	System
+38	CH9 Factory default setting gain value (L)	—	—	System
+39	CH9 Factory default setting gain value (H)	—	—	System
+40	CH10 Factory default setting offset value (L)	—	—	System
+41	CH10 Factory default setting offset value (H)	—	—	System
+42	CH10 Factory default setting gain value (L)	—	—	System
+43	CH10 Factory default setting gain value (H)	—	—	System
+44	CH11 Factory default setting offset value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+45	CH11 Factory default setting offset value (H)	—	—	System
+46	CH11 Factory default setting gain value (L)	—	—	System
+47	CH11 Factory default setting gain value (H)	—	—	System
+48	CH12 Factory default setting offset value (L)	—	—	System
+49	CH12 Factory default setting offset value (H)	—	—	System
+50	CH12 Factory default setting gain value (L)	—	—	System
+51	CH12 Factory default setting gain value (H)	—	—	System
+52	CH13 Factory default setting offset value (L)	—	—	System
+53	CH13 Factory default setting offset value (H)	—	—	System
+54	CH13 Factory default setting gain value (L)	—	—	System
+55	CH13 Factory default setting gain value (H)	—	—	System
+56	CH14 Factory default setting offset value (L)	—	—	System
+57	CH14 Factory default setting offset value (H)	—	—	System
+58	CH14 Factory default setting gain value (L)	—	—	System
+59	CH14 Factory default setting gain value (H)	—	—	System
+60	CH15 Factory default setting offset value (L)	—	—	System
+61	CH15 Factory default setting offset value (H)	—	—	System
+62	CH15 Factory default setting gain value (L)	—	—	System
+63	CH15 Factory default setting gain value (H)	—	—	System
+64	CH16 Factory default setting offset value (L)	—	—	System
+65	CH16 Factory default setting offset value (H)	—	—	System
+66	CH16 Factory default setting gain value (L)	—	—	System
+67	CH16 Factory default setting gain value (H)	—	—	System
+68	CH1 User range setting offset value (L)	—	—	System
+69	CH1 User range setting offset value (H)	—	—	System
+70	CH1 User range setting gain value (L)	—	—	System
+71	CH1 User range setting gain value (H)	—	—	System
+72	CH2 User range setting offset value (L)	—	—	System
+73	CH2 User range setting offset value (H)	—	—	System
+74	CH2 User range setting gain value (L)	—	—	System
+75	CH2 User range setting gain value (H)	—	—	System
+76	CH3 User range setting offset value (L)	—	—	System
+77	CH3 User range setting offset value (H)	—	—	System
+78	CH3 User range setting gain value (L)	—	—	System
+79	CH3 User range setting gain value (H)	—	—	System
+80	CH4 User range setting offset value (L)	—	—	System
+81	CH4 User range setting offset value (H)	—	—	System
+82	CH4 User range setting gain value (L)	—	—	System
+83	CH4 User range setting gain value (H)	—	—	System
+84	CH5 User range setting offset value (L)	—	—	System
+85	CH5 User range setting offset value (H)	—	—	System
+86	CH5 User range setting gain value (L)	—	—	System
+87	CH5 User range setting gain value (H)	—	—	System
+88	CH6 User range setting offset value (L)	—	—	System
+89	CH6 User range setting offset value (H)	—	—	System
+90	CH6 User range setting gain value (L)	—	—	System
+91	CH6 User range setting gain value (H)	—	—	System
+92	CH7 User range setting offset value (L)	—	—	System
+93	CH7 User range setting offset value (H)	—	—	System
+94	CH7 User range setting gain value (L)	—	—	System
+95	CH7 User range setting gain value (H)	—	—	System
+96	CH8 User range setting offset value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+97	CH8 User range setting offset value (H)	—	—	System
+98	CH8 User range setting gain value (L)	—	—	System
+99	CH8 User range setting gain value (H)	—	—	System
+100	CH9 User range setting offset value (L)	—	—	System
+101	CH9 User range setting offset value (H)	—	—	System
+102	CH9 User range setting gain value (L)	—	—	System
+103	CH9 User range setting gain value (H)	—	—	System
+104	CH10 User range setting offset value (L)	—	—	System
+105	CH10 User range setting offset value (H)	—	—	System
+106	CH10 User range setting gain value (L)	—	—	System
+107	CH10 User range setting gain value (H)	—	—	System
+108	CH11 User range setting offset value (L)	—	—	System
+109	CH11 User range setting offset value (H)	—	—	System
+110	CH11 User range setting gain value (L)	—	—	System
+111	CH11 User range setting gain value (H)	—	—	System
+112	CH12 User range setting offset value (L)	—	—	System
+113	CH12 User range setting offset value (H)	—	—	System
+114	CH12 User range setting gain value (L)	—	—	System
+115	CH12 User range setting gain value (H)	—	—	System
+116	CH13 User range setting offset value (L)	—	—	System
+117	CH13 User range setting offset value (H)	—	—	System
+118	CH13 User range setting gain value (L)	—	—	System
+119	CH13 User range setting gain value (H)	—	—	System
+120	CH14 User range setting offset value (L)	—	—	System
+121	CH14 User range setting offset value (H)	—	—	System
+122	CH14 User range setting gain value (L)	—	—	System
+123	CH14 User range setting gain value (H)	—	—	System
+124	CH15 User range setting offset value (L)	—	—	System
+125	CH15 User range setting offset value (H)	—	—	System
+126	CH15 User range setting gain value (L)	—	—	System
+127	CH15 User range setting gain value (H)	—	—	System
+128	CH16 User range setting offset value (L)	—	—	System
+129	CH16 User range setting offset value (H)	—	—	System
+130	CH16 User range setting gain value (L)	—	—	System
+131	CH16 User range setting gain value (H)	—	—	System

- For R60DA16-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting 1	Specify the user range setting for reading offset/gain setting values. This device allows a specification from CH1 to CH8. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to AAAAH	User
+3	Save data type setting 2	Specify the user range setting for reading offset/gain setting values. This device allows a specification from CH9 to CH16. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH16 CH15 CH14 CH13 CH12 CH11 CH10 CH9	0000H to AAAAH	User
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH5 Factory default setting offset value	—	—	System
+13	CH5 Factory default setting gain value	—	—	System
+14	CH6 Factory default setting offset value	—	—	System
+15	CH6 Factory default setting gain value	—	—	System
+16	CH7 Factory default setting offset value	—	—	System
+17	CH7 Factory default setting gain value	—	—	System
+18	CH8 Factory default setting offset value	—	—	System
+19	CH8 Factory default setting gain value	—	—	System
+20	CH9 Factory default setting offset value	—	—	System
+21	CH9 Factory default setting gain value	—	—	System
+22	CH10 Factory default setting offset value	—	—	System
+23	CH10 Factory default setting gain value	—	—	System
+24	CH11 Factory default setting offset value	—	—	System
+25	CH11 Factory default setting gain value	—	—	System
+26	CH12 Factory default setting offset value	—	—	System
+27	CH12 Factory default setting gain value	—	—	System
+28	CH13 Factory default setting offset value	—	—	System
+29	CH13 Factory default setting gain value	—	—	System
+30	CH14 Factory default setting offset value	—	—	System
+31	CH14 Factory default setting gain value	—	—	System
+32	CH15 Factory default setting offset value	—	—	System
+33	CH15 Factory default setting gain value	—	—	System
+34	CH16 Factory default setting offset value	—	—	System
+35	CH16 Factory default setting gain value	—	—	System
+36	CH1 User range setting offset value	—	—	System
+37	CH1 User range setting gain value	—	—	System
+38	CH2 User range setting offset value	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+39	CH2 User range setting gain value	—	—	System
+40	CH3 User range setting offset value	—	—	System
+41	CH3 User range setting gain value	—	—	System
+42	CH4 User range setting offset value	—	—	System
+43	CH4 User range setting gain value	—	—	System
+44	CH5 User range setting offset value	—	—	System
+45	CH5 User range setting gain value	—	—	System
+46	CH6 User range setting offset value	—	—	System
+47	CH6 User range setting gain value	—	—	System
+48	CH7 User range setting offset value	—	—	System
+49	CH7 User range setting gain value	—	—	System
+50	CH8 User range setting offset value	—	—	System
+51	CH8 User range setting gain value	—	—	System
+52	CH9 User range setting offset value	—	—	System
+53	CH9 User range setting gain value	—	—	System
+54	CH10 User range setting offset value	—	—	System
+55	CH10 User range setting gain value	—	—	System
+56	CH11 User range setting offset value	—	—	System
+57	CH11 User range setting gain value	—	—	System
+58	CH12 User range setting offset value	—	—	System
+59	CH12 User range setting gain value	—	—	System
+60	CH13 User range setting offset value	—	—	System
+61	CH13 User range setting gain value	—	—	System
+62	CH14 User range setting offset value	—	—	System
+63	CH14 User range setting gain value	—	—	System
+64	CH15 User range setting offset value	—	—	System
+65	CH15 User range setting gain value	—	—	System
+66	CH16 User range setting offset value	—	—	System
+67	CH16 User range setting gain value	—	—	System
+68	System area	—	—	—

Processing details

- These instructions read the offset/gain setting values of the user range settings of an analog module into the CPU module.
- Execution and normal/error completion of the G(P).OGLOAD instruction can be checked with the completion device specified by the setting data (d) and the completion status indication device (d)+1.

  - Completion device (d)

Turns on during END processing of the scan that arises upon completion of the G(P).OGLOAD instruction, and turns off during the next END processing.

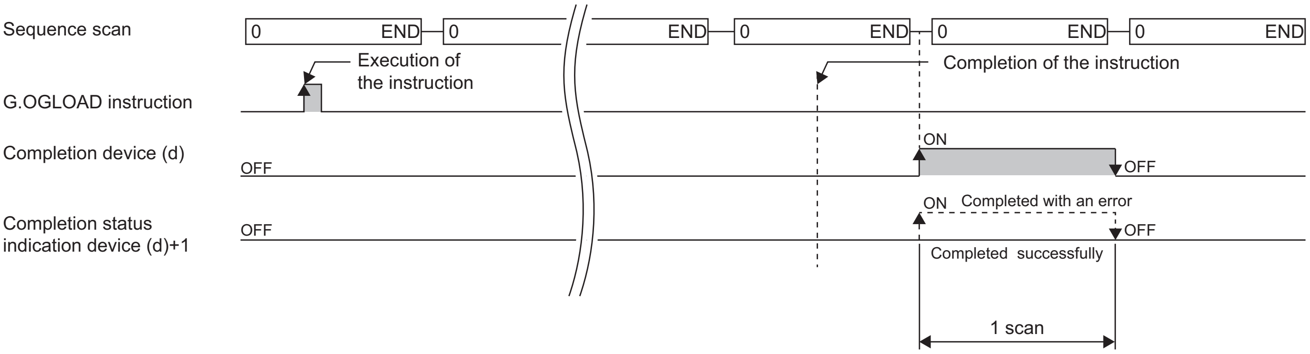
  - Completion status indication device (d)+1

Turns on or off depending on the status resulting from completion of the G(P).OGLOAD instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during END processing of the scan that arises upon completion of the G(P).OGLOAD instruction, and turns off during the next END processing.

- The following figure shows how the G(P).OGLOAD instruction operates when completing its execution.



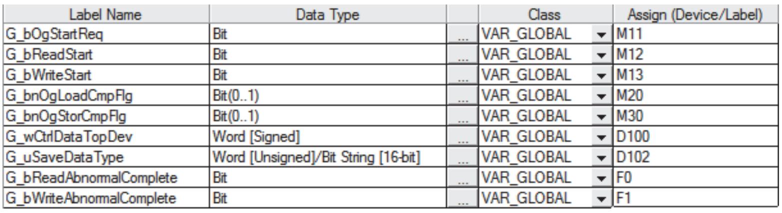
■Program example

The following figure shows an example of a program which uses the G(P).OGLOAD instruction with the R60AD4.

- Program content

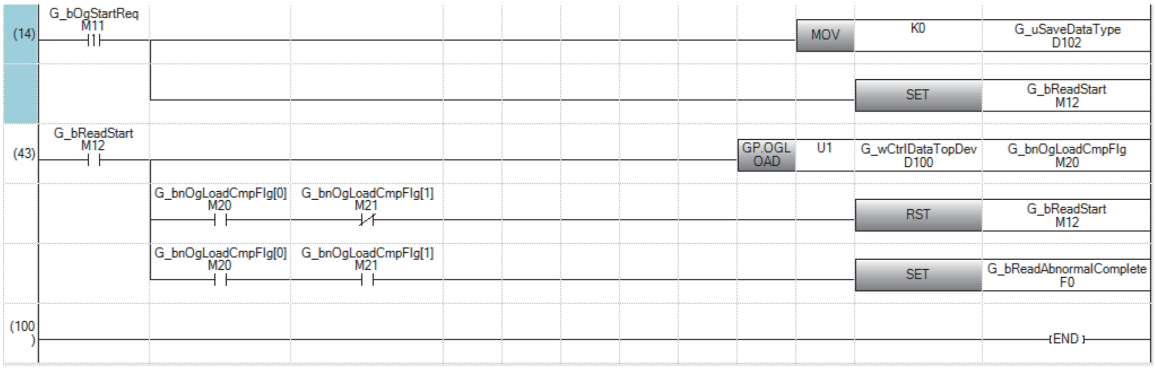
Description
When "Dedicated Instruction Execution Request" (M11) is turned on, the offset/gain settings of the analog module mounted at the position of I/O numbers X/Y10 to X/Y1F are read out to the "head device (D100) for storing the offset gain setting" and later.

- Label setting

Classification	Label name	Description	Device
Module label	—	—	—
Label to be defined	Define the global label as shown below. 		

Label Name	Data Type	Class	Assign (Device/Label)
G_bOgStartReq	Bit	VAR_GLOBAL	M11
G_bReadStart	Bit	VAR_GLOBAL	M12
G_bWriteStart	Bit	VAR_GLOBAL	M13
G_bnOgLoadCmpFlg	Bit(0..1)	VAR_GLOBAL	M20
G_bnOgStorCmpFlg	Bit(0..1)	VAR_GLOBAL	M30
G_wCtrlDataTopDev	Word [Signed]	VAR_GLOBAL	D100
G_uSaveDataType	Word [Unsigned]/Bit String [16-bit]	VAR_GLOBAL	D102
G_bReadAbnormalComplete	Bit	VAR_GLOBAL	F0
G_bWriteAbnormalComplete	Bit	VAR_GLOBAL	F1

- Program example



(14) Set control data.
Save data types need not be set for the following modules.
- R60ADV8
- R60ADI8
- R60DAV8
- R60DAI8
- R60TD8-G
- R60RD8-G

(43) Read the offset/gain settings.

Precautions

In the following cases, the G(P).OGLOAD instruction will be disabled.

- The module has been set as a target of synchronization.
- The simultaneous conversion mode has been selected on the high speed analog-digital converter module.
- The wave output mode has been selected on the D/A converter module.
- The high-speed output mode has been selected on the high speed digital-analog converter module.

Operation error

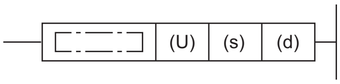
Error code ((s)+1)	Description
1863H	The value set in the save data type setting 1 (S)+2 or the save data type setting 2 (s)+3 is out of the valid range.

21.3 Restoring the User Range Setting Values

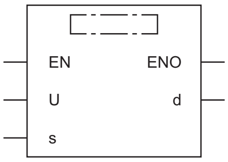
G(P).OGSTOR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

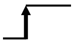
These instructions restore the offset/gain settings in the user range setting stored in a CPU module into an analog module.

Ladder	ST
	ENO:=G_OGSTOR(EN,U,s,d); ENO:=GP_OGSTOR(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.OGSTOR	
GP.OGSTOR	

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	0000H to 00FEH	• 16-bit unsigned binary • 16-bit signed binary	ANY16
(s)[1]	Start device where the control data is stored	Predefined devices	Device name	ANY16[2]
(d)	Device to be turned on one scan when instruction processing completes. If the instruction is completed with an error, (d)+1 is also turned on.	Predefined devices	Bit	ANYBIT_ARRAY[3] (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

[1] Specify the device that is specified in (s) when the G(P).OGLOAD instruction is to be executed. The data read with the G(P).OGLOAD instruction must be not be changed. Otherwise, normal operation cannot be guaranteed.

[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.

[3] When specifying setting data by using a label, use an array with two or more elements.

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s)	—	—	○[1]	—	—	—	—	○	—	—	—	—
(d)	○[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

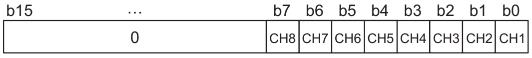
- For R60AD4, R60ADH4

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Used to store the set value that was set for Save data type setting (s)+2 in the G(P).OGLOAD instruction. • 0: Voltage specification • 1: Current specification b15 … b3 b2 b1 b0 0 | CH4 | CH3 | CH2 | CH1	0000H to 000FH	System
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH1 User range setting offset value (L)	—	—	System
+21	CH1 User range setting offset value (H)	—	—	System
+22	CH1 User range setting gain value (L)	—	—	System
+23	CH1 User range setting gain value (H)	—	—	System
+24	CH2 User range setting offset value (L)	—	—	System
+25	CH2 User range setting offset value (H)	—	—	System
+26	CH2 User range setting gain value (L)	—	—	System
+27	CH2 User range setting gain value (H)	—	—	System
+28	CH3 User range setting offset value (L)	—	—	System
+29	CH3 User range setting offset value (H)	—	—	System
+30	CH3 User range setting gain value (L)	—	—	System
+31	CH3 User range setting gain value (H)	—	—	System
+32	CH4 User range setting offset value (L)	—	—	System
+33	CH4 User range setting offset value (H)	—	—	System
+34	CH4 User range setting gain value (L)	—	—	System
+35	CH4 User range setting gain value (H)	—	—	System

- For R60DA4, R60DAH4

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Used to store the set value that was set for Save data type setting (s)+2 in the G(P).OGLOAD instruction. • 0: User range setting (voltage specification) • 1: User range setting (current specification) b15 … b3 b2 b1 b0 0 CH4 CH3 CH2 CH1	0000H to 000FH	System
+3	System area	—	—	—
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH1 User range setting offset value	—	—	System
+13	CH1 User range setting gain value	—	—	System
+14	CH2 User range setting offset value	—	—	System
+15	CH2 User range setting gain value	—	—	System
+16	CH3 User range setting offset value	—	—	System
+17	CH3 User range setting gain value	—	—	System
+18	CH4 User range setting offset value	—	—	System
+19	CH4 User range setting gain value	—	—	System

- For R60ADV8, R60ADI8, R60AD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting[*1]	Used to store the set value that was set for Save data type setting (s)+2 in the G(P).OGLOAD instruction. • 0: Voltage specification • 1: Current specification 	0000H to 00FFH	System
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH5 Factory default setting offset value (L)	—	—	System
+21	CH5 Factory default setting offset value (H)	—	—	System
+22	CH5 Factory default setting gain value (L)	—	—	System
+23	CH5 Factory default setting gain value (H)	—	—	System
+24	CH6 Factory default setting offset value (L)	—	—	System
+25	CH6 Factory default setting offset value (H)	—	—	System
+26	CH6 Factory default setting gain value (L)	—	—	System
+27	CH6 Factory default setting gain value (H)	—	—	System
+28	CH7 Factory default setting offset value (L)	—	—	System
+29	CH7 Factory default setting offset value (H)	—	—	System
+30	CH7 Factory default setting gain value (L)	—	—	System
+31	CH7 Factory default setting gain value (H)	—	—	System
+32	CH8 Factory default setting offset value (L)	—	—	System
+33	CH8 Factory default setting offset value (H)	—	—	System
+34	CH8 Factory default setting gain value (L)	—	—	System
+35	CH8 Factory default setting gain value (H)	—	—	System
+36	CH1 User range setting offset value (L)	—	—	System
+37	CH1 User range setting offset value (H)	—	—	System
+38	CH1 User range setting gain value (L)	—	—	System
+39	CH1 User range setting gain value (H)	—	—	System
+40	CH2 User range setting offset value (L)	—	—	System
+41	CH2 User range setting offset value (H)	—	—	System
+42	CH2 User range setting gain value (L)	—	—	System
+43	CH2 User range setting gain value (H)	—	—	System
+44	CH3 User range setting offset value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+45	CH3 User range setting offset value (H)	—	—	System
+46	CH3 User range setting gain value (L)	—	—	System
+47	CH3 User range setting gain value (H)	—	—	System
+48	CH4 User range setting offset value (L)	—	—	System
+49	CH4 User range setting offset value (H)	—	—	System
+50	CH4 User range setting gain value (L)	—	—	System
+51	CH4 User range setting gain value (H)	—	—	System
+52	CH5 User range setting offset value (L)	—	—	System
+53	CH5 User range setting offset value (H)	—	—	System
+54	CH5 User range setting gain value (L)	—	—	System
+55	CH5 User range setting gain value (H)	—	—	System
+56	CH6 User range setting offset value (L)	—	—	System
+57	CH6 User range setting offset value (H)	—	—	System
+58	CH6 User range setting gain value (L)	—	—	System
+59	CH6 User range setting gain value (H)	—	—	System
+60	CH7 User range setting offset value (L)	—	—	System
+61	CH7 User range setting offset value (H)	—	—	System
+62	CH7 User range setting gain value (L)	—	—	System
+63	CH7 User range setting gain value (H)	—	—	System
+64	CH8 User range setting offset value (L)	—	—	System
+65	CH8 User range setting offset value (H)	—	—	System
+66	CH8 User range setting gain value (L)	—	—	System
+67	CH8 User range setting gain value (H)	—	—	System

[1] For the R60ADV8 and the R60ADI8, this area corresponds to System area and so is not available.

- For R60DAV8, R60DAI8, R60DA8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting[*2]	Used to store the set value that was set for Save data type setting (s)+2 in the G(P).OGLOAD instruction. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to AAAAH	System
+3	System area	—	—	—
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH5 Factory default setting offset value	—	—	System
+13	CH5 Factory default setting gain value	—	—	System
+14	CH6 Factory default setting offset value	—	—	System
+15	CH6 Factory default setting gain value	—	—	System
+16	CH7 Factory default setting offset value	—	—	System
+17	CH7 Factory default setting gain value	—	—	System
+18	CH8 Factory default setting offset value	—	—	System
+19	CH8 Factory default setting gain value	—	—	System
+20	CH1 User range setting offset value	—	—	System
+21	CH1 User range setting gain value	—	—	System
+22	CH2 User range setting offset value	—	—	System
+23	CH2 User range setting gain value	—	—	System
+24	CH3 User range setting offset value	—	—	System
+25	CH3 User range setting gain value	—	—	System
+26	CH4 User range setting offset value	—	—	System
+27	CH4 User range setting gain value	—	—	System
+28	CH5 User range setting offset value	—	—	System
+29	CH5 User range setting gain value	—	—	System
+30	CH6 User range setting offset value	—	—	System
+31	CH6 User range setting gain value	—	—	System
+32	CH7 User range setting offset value	—	—	System
+33	CH7 User range setting gain value	—	—	System
+34	CH8 User range setting offset value	—	—	System
+35	CH8 User range setting gain value	—	—	System
+36	System area	—	—	—

[*2] For the R60DAV8 and the R60DAI8, this area corresponds to System area and so is not available.

- For R60TD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	System area	—	—	—
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH1 User range setting offset value (L)	—	—	System
+9	CH1 User range setting offset value (H)	—	—	System
+10	CH1 User range setting gain value (L)	—	—	System
+11	CH1 User range setting gain value (H)	—	—	System
+12	CH1 User range setting thermoelectromotive force offset value (L)	—	—	System
+13	CH1 User range setting thermoelectromotive force offset value (H)	—	—	System
+14	CH1 User range setting thermoelectromotive force gain value (L)	—	—	System
+15	CH1 User range setting thermoelectromotive force gain value (H)	—	—	System
+16	CH2 Factory default setting offset value (L)	—	—	System
+17	CH2 Factory default setting offset value (H)	—	—	System
+18	CH2 Factory default setting gain value (L)	—	—	System
+19	CH2 Factory default setting gain value (H)	—	—	System
+20	CH2 User range setting offset value (L)	—	—	System
+21	CH2 User range setting offset value (H)	—	—	System
+22	CH2 User range setting gain value (L)	—	—	System
+23	CH2 User range setting gain value (H)	—	—	System
+24	CH2 User range setting thermoelectromotive force offset value (L)	—	—	System
+25	CH2 User range setting thermoelectromotive force offset value (H)	—	—	System
+26	CH2 User range setting thermoelectromotive force gain value (L)	—	—	System
+27	CH2 User range setting thermoelectromotive force gain value (H)	—	—	System
+28	CH3 Factory default setting offset value (L)	—	—	System
+29	CH3 Factory default setting offset value (H)	—	—	System
+30	CH3 Factory default setting gain value (L)	—	—	System
+31	CH3 Factory default setting gain value (H)	—	—	System
+32	CH3 User range setting offset value (L)	—	—	System
+33	CH3 User range setting offset value (H)	—	—	System
+34	CH3 User range setting gain value (L)	—	—	System
+35	CH3 User range setting gain value (H)	—	—	System
+36	CH3 User range setting thermoelectromotive force offset value (L)	—	—	System
+37	CH3 User range setting thermoelectromotive force offset value (H)	—	—	System
+38	CH3 User range setting thermoelectromotive force gain value (L)	—	—	System
+39	CH3 User range setting thermoelectromotive force gain value (H)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+40	CH4 Factory default setting offset value (L)	—	—	System
+41	CH4 Factory default setting offset value (H)	—	—	System
+42	CH4 Factory default setting gain value (L)	—	—	System
+43	CH4 Factory default setting gain value (H)	—	—	System
+44	CH4 User range setting offset value (L)	—	—	System
+45	CH4 User range setting offset value (H)	—	—	System
+46	CH4 User range setting gain value (L)	—	—	System
+47	CH4 User range setting gain value (H)	—	—	System
+48	CH4 User range setting thermoelectromotive force offset value (L)	—	—	System
+49	CH4 User range setting thermoelectromotive force offset value (H)	—	—	System
+50	CH4 User range setting thermoelectromotive force gain value (L)	—	—	System
+51	CH4 User range setting thermoelectromotive force gain value (H)	—	—	System
+52	CH5 Factory default setting offset value (L)	—	—	System
+53	CH5 Factory default setting offset value (H)	—	—	System
+54	CH5 Factory default setting gain value (L)	—	—	System
+55	CH5 Factory default setting gain value (H)	—	—	System
+56	CH5 User range setting offset value (L)	—	—	System
+57	CH5 User range setting offset value (H)	—	—	System
+58	CH5 User range setting gain value (L)	—	—	System
+59	CH5 User range setting gain value (H)	—	—	System
+60	CH5 User range setting thermoelectromotive force offset value (L)	—	—	System
+61	CH5 User range setting thermoelectromotive force offset value (H)	—	—	System
+62	CH5 User range setting thermoelectromotive force gain value (L)	—	—	System
+63	CH5 User range setting thermoelectromotive force gain value (H)	—	—	System
+64	CH6 Factory default setting offset value (L)	—	—	System
+65	CH6 Factory default setting offset value (H)	—	—	System
+66	CH6 Factory default setting gain value (L)	—	—	System
+67	CH6 Factory default setting gain value (H)	—	—	System
+68	CH6 User range setting offset value (L)	—	—	System
+69	CH6 User range setting offset value (H)	—	—	System
+70	CH6 User range setting gain value (L)	—	—	System
+71	CH6 User range setting gain value (H)	—	—	System
+72	CH6 User range setting thermoelectromotive force offset value (L)	—	—	System
+73	CH6 User range setting thermoelectromotive force offset value (H)	—	—	System
+74	CH6 User range setting thermoelectromotive force gain value (L)	—	—	System
+75	CH6 User range setting thermoelectromotive force gain value (H)	—	—	System
+76	CH7 Factory default setting offset value (L)	—	—	System
+77	CH7 Factory default setting offset value (H)	—	—	System
+78	CH7 Factory default setting gain value (L)	—	—	System
+79	CH7 Factory default setting gain value (H)	—	—	System
+80	CH7 User range setting offset value (L)	—	—	System
+81	CH7 User range setting offset value (H)	—	—	System
+82	CH7 User range setting gain value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+83	CH7 User range setting gain value (H)	—	—	System
+84	CH7 User range setting thermoelectromotive force offset value (L)	—	—	System
+85	CH7 User range setting thermoelectromotive force offset value (H)	—	—	System
+86	CH7 User range setting thermoelectromotive force gain value (L)	—	—	System
+87	CH7 User range setting thermoelectromotive force gain value (H)	—	—	System
+88	CH8 Factory default setting offset value (L)	—	—	System
+89	CH8 Factory default setting offset value (H)	—	—	System
+90	CH8 Factory default setting gain value (L)	—	—	System
+91	CH8 Factory default setting gain value (H)	—	—	System
+92	CH8 User range setting offset value (L)	—	—	System
+93	CH8 User range setting offset value (H)	—	—	System
+94	CH8 User range setting gain value (L)	—	—	System
+95	CH8 User range setting gain value (H)	—	—	System
+96	CH8 User range setting thermoelectromotive force offset value (L)	—	—	System
+97	CH8 User range setting thermoelectromotive force offset value (H)	—	—	System
+98	CH8 User range setting thermoelectromotive force gain value (L)	—	—	System
+99	CH8 User range setting thermoelectromotive force gain value (H)	—	—	System

- For R60RD8-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	System area	—	—	—
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH1 User range setting offset value (L)	—	—	System
+9	CH1 User range setting offset value (H)	—	—	System
+10	CH1 User range setting gain value (L)	—	—	System
+11	CH1 User range setting gain value (H)	—	—	System
+12	CH1 User range setting offset resistance value (L)	—	—	System
+13	CH1 User range setting offset resistance value (H)	—	—	System
+14	CH1 User range setting gain resistance value (L)	—	—	System
+15	CH1 User range setting gain resistance value (H)	—	—	System
+16	CH2 Factory default setting offset value (L)	—	—	System
+17	CH2 Factory default setting offset value (H)	—	—	System
+18	CH2 Factory default setting gain value (L)	—	—	System
+19	CH2 Factory default setting gain value (H)	—	—	System
+20	CH2 User range setting offset value (L)	—	—	System
+21	CH2 User range setting offset value (H)	—	—	System
+22	CH2 User range setting gain value (L)	—	—	System
+23	CH2 User range setting gain value (H)	—	—	System
+24	CH2 User range setting offset resistance value (L)	—	—	System
+25	CH2 User range setting offset resistance value (H)	—	—	System
+26	CH2 User range setting gain resistance value (L)	—	—	System
+27	CH2 User range setting gain resistance value (H)	—	—	System
+28	CH3 Factory default setting offset value (L)	—	—	System
+29	CH3 Factory default setting offset value (H)	—	—	System
+30	CH3 Factory default setting gain value (L)	—	—	System
+31	CH3 Factory default setting gain value (H)	—	—	System
+32	CH3 User range setting offset value (L)	—	—	System
+33	CH3 User range setting offset value (H)	—	—	System
+34	CH3 User range setting gain value (L)	—	—	System
+35	CH3 User range setting gain value (H)	—	—	System
+36	CH3 User range setting offset resistance value (L)	—	—	System
+37	CH3 User range setting offset resistance value (H)	—	—	System
+38	CH3 User range setting gain resistance value (L)	—	—	System
+39	CH3 User range setting gain resistance value (H)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+40	CH4 Factory default setting offset value (L)	—	—	System
+41	CH4 Factory default setting offset value (H)	—	—	System
+42	CH4 Factory default setting gain value (L)	—	—	System
+43	CH4 Factory default setting gain value (H)	—	—	System
+44	CH4 User range setting offset value (L)	—	—	System
+45	CH4 User range setting offset value (H)	—	—	System
+46	CH4 User range setting gain value (L)	—	—	System
+47	CH4 User range setting gain value (H)	—	—	System
+48	CH4 User range setting offset resistance value (L)	—	—	System
+49	CH4 User range setting offset resistance value (H)	—	—	System
+50	CH4 User range setting gain resistance value (L)	—	—	System
+51	CH4 User range setting gain resistance value (H)	—	—	System
+52	CH5 Factory default setting offset value (L)	—	—	System
+53	CH5 Factory default setting offset value (H)	—	—	System
+54	CH5 Factory default setting gain value (L)	—	—	System
+55	CH5 Factory default setting gain value (H)	—	—	System
+56	CH5 User range setting offset value (L)	—	—	System
+57	CH5 User range setting offset value (H)	—	—	System
+58	CH5 User range setting gain value (L)	—	—	System
+59	CH5 User range setting gain value (H)	—	—	System
+60	CH5 User range setting offset resistance value (L)	—	—	System
+61	CH5 User range setting offset resistance value (H)	—	—	System
+62	CH5 User range setting gain resistance value (L)	—	—	System
+63	CH5 User range setting gain resistance value (H)	—	—	System
+64	CH6 Factory default setting offset value (L)	—	—	System
+65	CH6 Factory default setting offset value (H)	—	—	System
+66	CH6 Factory default setting gain value (L)	—	—	System
+67	CH6 Factory default setting gain value (H)	—	—	System
+68	CH6 User range setting offset value (L)	—	—	System
+69	CH6 User range setting offset value (H)	—	—	System
+70	CH6 User range setting gain value (L)	—	—	System
+71	CH6 User range setting gain value (H)	—	—	System
+72	CH6 User range setting offset resistance value (L)	—	—	System
+73	CH6 User range setting offset resistance value (H)	—	—	System
+74	CH6 User range setting gain resistance value (L)	—	—	System
+75	CH6 User range setting gain resistance value (H)	—	—	System
+76	CH7 Factory default setting offset value (L)	—	—	System
+77	CH7 Factory default setting offset value (H)	—	—	System
+78	CH7 Factory default setting gain value (L)	—	—	System
+79	CH7 Factory default setting gain value (H)	—	—	System
+80	CH7 User range setting offset value (L)	—	—	System
+81	CH7 User range setting offset value (H)	—	—	System
+82	CH7 User range setting gain value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+83	CH7 User range setting gain value (H)	—	—	System
+84	CH7 User range setting offset resistance value (L)	—	—	System
+85	CH7 User range setting offset resistance value (H)	—	—	System
+86	CH7 User range setting gain resistance value (L)	—	—	System
+87	CH7 User range setting gain resistance value (H)	—	—	System
+88	CH8 Factory default setting offset value (L)	—	—	System
+89	CH8 Factory default setting offset value (H)	—	—	System
+90	CH8 Factory default setting gain value (L)	—	—	System
+91	CH8 Factory default setting gain value (H)	—	—	System
+92	CH8 User range setting offset value (L)	—	—	System
+93	CH8 User range setting offset value (H)	—	—	System
+94	CH8 User range setting gain value (L)	—	—	System
+95	CH8 User range setting gain value (H)	—	—	System
+96	CH8 User range setting offset resistance value (L)	—	—	System
+97	CH8 User range setting offset resistance value (H)	—	—	System
+98	CH8 User range setting gain resistance value (L)	—	—	System
+99	CH8 User range setting gain resistance value (H)	—	—	System

- For R60AD16-G

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting	Used to store the set value that was set for Save data type setting (s)+2 in the G(P).OGLOAD instruction. • 0: Voltage specification • 1: Current specification b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH16 CH15 CH14 CH13 CH12 CH11 CH10 CH9 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to FFFFH	System
+3	System area	—	—	—
+4	CH1 Factory default setting offset value (L)	—	—	System
+5	CH1 Factory default setting offset value (H)	—	—	System
+6	CH1 Factory default setting gain value (L)	—	—	System
+7	CH1 Factory default setting gain value (H)	—	—	System
+8	CH2 Factory default setting offset value (L)	—	—	System
+9	CH2 Factory default setting offset value (H)	—	—	System
+10	CH2 Factory default setting gain value (L)	—	—	System
+11	CH2 Factory default setting gain value (H)	—	—	System
+12	CH3 Factory default setting offset value (L)	—	—	System
+13	CH3 Factory default setting offset value (H)	—	—	System
+14	CH3 Factory default setting gain value (L)	—	—	System
+15	CH3 Factory default setting gain value (H)	—	—	System
+16	CH4 Factory default setting offset value (L)	—	—	System
+17	CH4 Factory default setting offset value (H)	—	—	System
+18	CH4 Factory default setting gain value (L)	—	—	System
+19	CH4 Factory default setting gain value (H)	—	—	System
+20	CH5 Factory default setting offset value (L)	—	—	System
+21	CH5 Factory default setting offset value (H)	—	—	System
+22	CH5 Factory default setting gain value (L)	—	—	System
+23	CH5 Factory default setting gain value (H)	—	—	System
+24	CH6 Factory default setting offset value (L)	—	—	System
+25	CH6 Factory default setting offset value (H)	—	—	System
+26	CH6 Factory default setting gain value (L)	—	—	System
+27	CH6 Factory default setting gain value (H)	—	—	System
+28	CH7 Factory default setting offset value (L)	—	—	System
+29	CH7 Factory default setting offset value (H)	—	—	System
+30	CH7 Factory default setting gain value (L)	—	—	System
+31	CH7 Factory default setting gain value (H)	—	—	System
+32	CH8 Factory default setting offset value (L)	—	—	System
+33	CH8 Factory default setting offset value (H)	—	—	System
+34	CH8 Factory default setting gain value (L)	—	—	System
+35	CH8 Factory default setting gain value (H)	—	—	System
+36	CH9 Factory default setting offset value (L)	—	—	System
+37	CH9 Factory default setting offset value (H)	—	—	System
+38	CH9 Factory default setting gain value (L)	—	—	System
+39	CH9 Factory default setting gain value (H)	—	—	System
+40	CH10 Factory default setting offset value (L)	—	—	System
+41	CH10 Factory default setting offset value (H)	—	—	System
+42	CH10 Factory default setting gain value (L)	—	—	System
+43	CH10 Factory default setting gain value (H)	—	—	System
+44	CH11 Factory default setting offset value (L)	—	—	System

Operand: (s)

Device	Item	Description	Setting range	Set by
+45	CH11 Factory default setting offset value (H)	—	—	System
+46	CH11 Factory default setting gain value (L)	—	—	System
+47	CH11 Factory default setting gain value (H)	—	—	System
+48	CH12 Factory default setting offset value (L)	—	—	System
+49	CH12 Factory default setting offset value (H)	—	—	System
+50	CH12 Factory default setting gain value (L)	—	—	System
+51	CH12 Factory default setting gain value (H)	—	—	System
+52	CH13 Factory default setting offset value (L)	—	—	System
+53	CH13 Factory default setting offset value (H)	—	—	System
+54	CH13 Factory default setting gain value (L)	—	—	System
+55	CH13 Factory default setting gain value (H)	—	—	System
+56	CH14 Factory default setting offset value (L)	—	—	System
+57	CH14 Factory default setting offset value (H)	—	—	System
+58	CH14 Factory default setting gain value (L)	—	—	System
+59	CH14 Factory default setting gain value (H)	—	—	System
+60	CH15 Factory default setting offset value (L)	—	—	System
+61	CH15 Factory default setting offset value (H)	—	—	System
+62	CH15 Factory default setting gain value (L)	—	—	System
+63	CH15 Factory default setting gain value (H)	—	—	System
+64	CH16 Factory default setting offset value (L)	—	—	System
+65	CH16 Factory default setting offset value (H)	—	—	System
+66	CH16 Factory default setting gain value (L)	—	—	System
+67	CH16 Factory default setting gain value (H)	—	—	System
+68	CH1 User range setting offset value (L)	—	—	System
+69	CH1 User range setting offset value (H)	—	—	System
+70	CH1 User range setting gain value (L)	—	—	System
+71	CH1 User range setting gain value (H)	—	—	System
+72	CH2 User range setting offset value (L)	—	—	System
+73	CH2 User range setting offset value (H)	—	—	System
+74	CH2 User range setting gain value (L)	—	—	System
+75	CH2 User range setting gain value (H)	—	—	System
+76	CH3 User range setting offset value (L)	—	—	System
+77	CH3 User range setting offset value (H)	—	—	System
+78	CH3 User range setting gain value (L)	—	—	System
+79	CH3 User range setting gain value (H)	—	—	System
+80	CH4 User range setting offset value (L)	—	—	System
+81	CH4 User range setting offset value (H)	—	—	System
+82	CH4 User range setting gain value (L)	—	—	System
+83	CH4 User range setting gain value (H)	—	—	System
+84	CH5 User range setting offset value (L)	—	—	System
+85	CH5 User range setting offset value (H)	—	—	System
+86	CH5 User range setting gain value (L)	—	—	System
+87	CH5 User range setting gain value (H)	—	—	System
+88	CH6 User range setting offset value (L)	—	—	System
+89	CH6 User range setting offset value (H)	—	—	System
+90	CH6 User range setting gain value (L)	—	—	System
+91	CH6 User range setting gain value (H)	—	—	System
+92	CH7 User range setting offset value (L)	—	—	System
+93	CH7 User range setting offset value (H)	—	—	System
+94	CH7 User range setting gain value (L)	—	—	System
+95	CH7 User range setting gain value (H)	—	—	System
+96	CH8 User range setting offset value (L)	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+97	CH8 User range setting offset value (H)	—	—	System
+98	CH8 User range setting gain value (L)	—	—	System
+99	CH8 User range setting gain value (H)	—	—	System
+100	CH9 User range setting offset value (L)	—	—	System
+101	CH9 User range setting offset value (H)	—	—	System
+102	CH9 User range setting gain value (L)	—	—	System
+103	CH9 User range setting gain value (H)	—	—	System
+104	CH10 User range setting offset value (L)	—	—	System
+105	CH10 User range setting offset value (H)	—	—	System
+106	CH10 User range setting gain value (L)	—	—	System
+107	CH10 User range setting gain value (H)	—	—	System
+108	CH11 User range setting offset value (L)	—	—	System
+109	CH11 User range setting offset value (H)	—	—	System
+110	CH11 User range setting gain value (L)	—	—	System
+111	CH11 User range setting gain value (H)	—	—	System
+112	CH12 User range setting offset value (L)	—	—	System
+113	CH12 User range setting offset value (H)	—	—	System
+114	CH12 User range setting gain value (L)	—	—	System
+115	CH12 User range setting gain value (H)	—	—	System
+116	CH13 User range setting offset value (L)	—	—	System
+117	CH13 User range setting offset value (H)	—	—	System
+118	CH13 User range setting gain value (L)	—	—	System
+119	CH13 User range setting gain value (H)	—	—	System
+120	CH14 User range setting offset value (L)	—	—	System
+121	CH14 User range setting offset value (H)	—	—	System
+122	CH14 User range setting gain value (L)	—	—	System
+123	CH14 User range setting gain value (H)	—	—	System
+124	CH15 User range setting offset value (L)	—	—	System
+125	CH15 User range setting offset value (H)	—	—	System
+126	CH15 User range setting gain value (L)	—	—	System
+127	CH15 User range setting gain value (H)	—	—	System
+128	CH16 User range setting offset value (L)	—	—	System
+129	CH16 User range setting offset value (H)	—	—	System
+130	CH16 User range setting gain value (L)	—	—	System
+131	CH16 User range setting gain value (H)	—	—	System

- For R60DA16-G

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Save data type setting 1	Used to store the set value that was set for Save data type setting 1 (s)+2 in the G(P).OGLOAD instruction. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH8 CH7 CH6 CH5 CH4 CH3 CH2 CH1	0000H to AAAAH	System
+3	Save data type setting 2	Used to store the set value that was set for Save data type setting 2 (s)+3 in the G(P).OGLOAD instruction. • 0H: User range setting 1 (current specification) • 1H: User range setting 2 (voltage specification) • 2H: User range setting 3 (voltage specification) b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 CH16 CH15 CH14 CH13 CH12 CH11 CH10 CH9	0000H to AAAAH	System
+4	CH1 Factory default setting offset value	—	—	System
+5	CH1 Factory default setting gain value	—	—	System
+6	CH2 Factory default setting offset value	—	—	System
+7	CH2 Factory default setting gain value	—	—	System
+8	CH3 Factory default setting offset value	—	—	System
+9	CH3 Factory default setting gain value	—	—	System
+10	CH4 Factory default setting offset value	—	—	System
+11	CH4 Factory default setting gain value	—	—	System
+12	CH5 Factory default setting offset value	—	—	System
+13	CH5 Factory default setting gain value	—	—	System
+14	CH6 Factory default setting offset value	—	—	System
+15	CH6 Factory default setting gain value	—	—	System
+16	CH7 Factory default setting offset value	—	—	System
+17	CH7 Factory default setting gain value	—	—	System
+18	CH8 Factory default setting offset value	—	—	System
+19	CH8 Factory default setting gain value	—	—	System
+20	CH9 Factory default setting offset value	—	—	System
+21	CH9 Factory default setting gain value	—	—	System
+22	CH10 Factory default setting offset value	—	—	System
+23	CH10 Factory default setting gain value	—	—	System
+24	CH11 Factory default setting offset value	—	—	System
+25	CH11 Factory default setting gain value	—	—	System
+26	CH12 Factory default setting offset value	—	—	System
+27	CH12 Factory default setting gain value	—	—	System
+28	CH13 Factory default setting offset value	—	—	System
+29	CH13 Factory default setting gain value	—	—	System
+30	CH14 Factory default setting offset value	—	—	System
+31	CH14 Factory default setting gain value	—	—	System
+32	CH15 Factory default setting offset value	—	—	System
+33	CH15 Factory default setting gain value	—	—	System
+34	CH16 Factory default setting offset value	—	—	System
+35	CH16 Factory default setting gain value	—	—	System
+36	CH1 User range setting offset value	—	—	System
+37	CH1 User range setting gain value	—	—	System
+38	CH2 User range setting offset value	—	—	System

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+39	CH2 User range setting gain value	—	—	System
+40	CH3 User range setting offset value	—	—	System
+41	CH3 User range setting gain value	—	—	System
+42	CH4 User range setting offset value	—	—	System
+43	CH4 User range setting gain value	—	—	System
+44	CH5 User range setting offset value	—	—	System
+45	CH5 User range setting gain value	—	—	System
+46	CH6 User range setting offset value	—	—	System
+47	CH6 User range setting gain value	—	—	System
+48	CH7 User range setting offset value	—	—	System
+49	CH7 User range setting gain value	—	—	System
+50	CH8 User range setting offset value	—	—	System
+51	CH8 User range setting gain value	—	—	System
+52	CH9 User range setting offset value	—	—	System
+53	CH9 User range setting gain value	—	—	System
+54	CH10 User range setting offset value	—	—	System
+55	CH10 User range setting gain value	—	—	System
+56	CH11 User range setting offset value	—	—	System
+57	CH11 User range setting gain value	—	—	System
+58	CH12 User range setting offset value	—	—	System
+59	CH12 User range setting gain value	—	—	System
+60	CH13 User range setting offset value	—	—	System
+61	CH13 User range setting gain value	—	—	System
+62	CH14 User range setting offset value	—	—	System
+63	CH14 User range setting gain value	—	—	System
+64	CH15 User range setting offset value	—	—	System
+65	CH15 User range setting gain value	—	—	System
+66	CH16 User range setting offset value	—	—	System
+67	CH16 User range setting gain value	—	—	System
+68	System area	—	—	—

Processing details

- These instructions restore the offset/gain settings in the user range setting stored in a CPU module into an analog module.
- Execution and normal/error completion of the G(P).OGSTOR instruction can be checked with the completion device specified by the setting data (d) and the completion status indication device (d)+1.

  - Completion device (d)

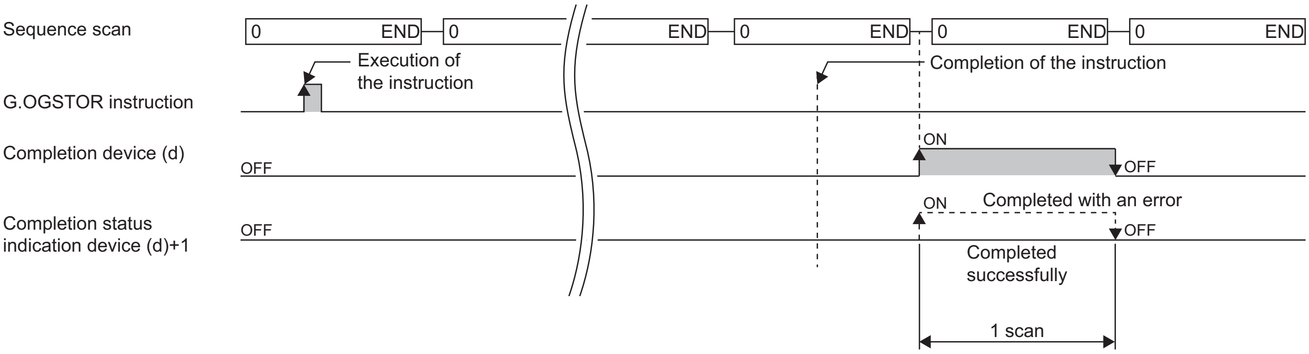
  Turns on during END processing of the scan that arises upon completion of the G(P).OGSTOR instruction, and turns off during the next END processing.

  - Completion status indication device (d)+1

  Turns on or off depending on the status resulting from completion of the G(P).OGSTOR instruction.

  When completed normally: Unchanged from off.

  When completed with an error: Turns on during END processing of the scan that arises upon completion of the G(P).OGSTOR instruction, and turns off during the next END processing.
- The following figure shows how the G(P).OGSTOR instruction operates when completing its execution.



- After the offset/gain setting value has been restored, the base rate decreases to about three times or more of the unrestored accuracy.

Point

When the G(P).OGSTOR instruction is executed, the D/A converter module stops the D/A conversion. To resume the D/A conversion, turn on and off the operating condition setting request (Y9).

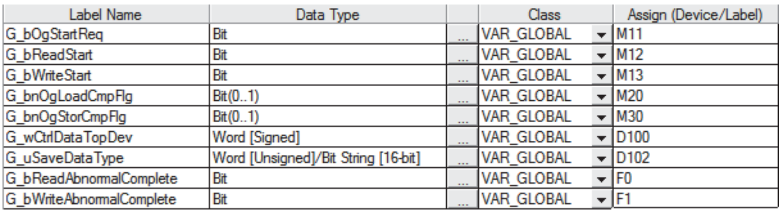
■Program example

The following figure shows an example of a program which uses the G(P).OGSTOR instruction with the R60AD4.

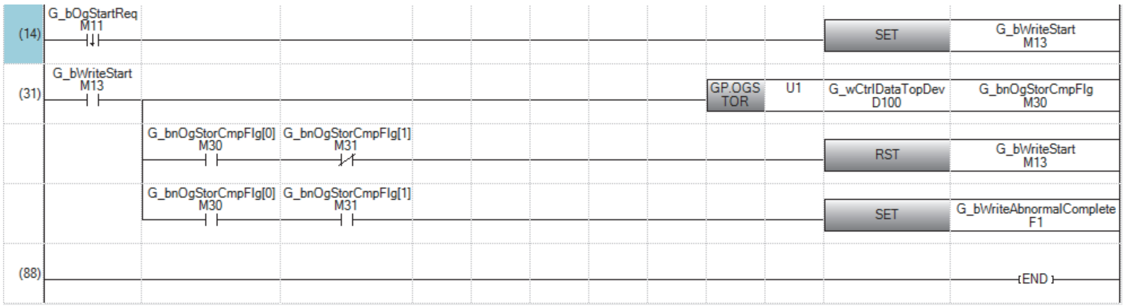
- Program content

Description
When "Dedicated Instruction Execution Request" (M11) is turned off, the offset/gain settings are restored in the analog module mounted at the position of I/O numbers X/Y10 to X/Y1F.

- Label setting

<table>
<tr><th>Classification</th><th>Label name</th><th>Description</th><th>Device</th></tr>
<tr><td>Module label</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>Label to be defined</td><td colspan="3">Define the global label as shown below.

<table>
<tr><th>Label Name</th><th>Data Type</th><th></th><th>Class</th><th>Assign (Device/Label)</th></tr>
<tr><td>G_bOgStartReq</td><td>Bit</td><td>...</td><td>VAR_GLOBAL</td><td>M11</td></tr>
<tr><td>G_bReadStart</td><td>Bit</td><td>...</td><td>VAR_GLOBAL</td><td>M12</td></tr>
<tr><td>G_bWriteStart</td><td>Bit</td><td>...</td><td>VAR_GLOBAL</td><td>M13</td></tr>
<tr><td>G_bnOgLoadCmpFlg</td><td>Bit(0..1)</td><td>...</td><td>VAR_GLOBAL</td><td>M20</td></tr>
<tr><td>G_bnOgStorCmpFlg</td><td>Bit(0..1)</td><td>...</td><td>VAR_GLOBAL</td><td>M30</td></tr>
<tr><td>G_wCtrlDataTopDev</td><td>Word [Signed]</td><td>...</td><td>VAR_GLOBAL</td><td>D100</td></tr>
<tr><td>G_uSaveDataType</td><td>Word [Unsigned]/Bit String [16-bit]</td><td>...</td><td>VAR_GLOBAL</td><td>D102</td></tr>
<tr><td>G_bReadAbnormalComplete</td><td>Bit</td><td>...</td><td>VAR_GLOBAL</td><td>F0</td></tr>
<tr><td>G_bWriteAbnormalComplete</td><td>Bit</td><td>...</td><td>VAR_GLOBAL</td><td>F1</td></tr>
</table></td></tr>
</table>

- Program example



(14) Set control data.

(31) Restore the offset/gain settings.

Precautions

In the following cases, execution of the G(P).OGSTOR instruction results in an error.

- The module has been set as a target of synchronization.
- The simultaneous conversion mode has been selected on the high speed analog-digital converter module.
- The wave output mode has been selected on the D/A converter module.

In the following case, execution of the G(P).OGSTOR instruction will be disabled.

- The high-speed output mode has been selected on the high speed digital-analog converter module.

Operation error

Error code ((s)+1)	Description
1860H	In the following cases, the G(P).OGSTOR instruction will be being executed. • During offset/gain setting module • Has been set as a target of module-to-module synchronization. • During simultaneous conversion mode (R60ADH4) • During wave output mode (D/A converter module)
1861H	The G(P).OGSTOR instruction is being executed consecutively.
1862H	The G(P).OGSTOR instruction is being executed for a model different from that for which the G(P).OGLOAD instruction has been executed.
	The G(P).OGSTOR instruction is being executed before the G(P).OGLOAD instruction is executed.
1863H	The value set in the save data type setting 1 (S)+2 or the save data type setting 2 (s)+3 is out of the valid range.

22 POSITIONING INSTRUCTIONS

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. When using MELSEC-Q series modules, refer to the manual for each module used and create programs.

For precautions when using modules, refer to the following.

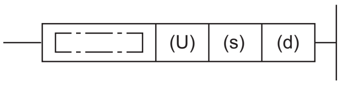
📖 MELSEC iQ-R Module Configuration Manual

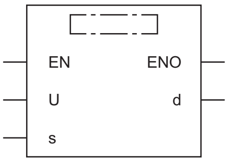
22.1 Restoring the Absolute Position

G.ABRST□, Z.ABRST□

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These instructions restore the absolute position of specified axis.

Ladder	ST
	ENO:=G_ABRST1(EN,U,s,d); ENO:=G_ABRST2(EN,U,s,d); ENO:=G_ABRST3(EN,U,s,d); ENO:=G_ABRST4(EN,U,s,d); ENO:=Z_ABRST1(EN,U,s,d); ENO:=Z_ABRST2(EN,U,s,d); ENO:=Z_ABRST3(EN,U,s,d); ENO:=Z_ABRST4(EN,U,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
G.ABRST1 G.ABRST2 G.ABRST3 G.ABRST4 Z.ABRST1 Z.ABRST2 Z.ABRST3 Z.ABRST4	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G.ABRST□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
	Z.ABRST□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device where the control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device to be turned on one scan upon completion of instruction If the instruction is completed with an error, (d)+1 is also turned on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	G.ABRST□	—	—	○	—	—	—	—	○	○	—	—	○
	Z.ABRST□	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○[3]	—	—	—	—	○	—	—	—	—
(d)		○[1]	—	○[2]	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, C, and FD cannot be used.

*3 FD cannot be used.

■Control data

Operand: (s)

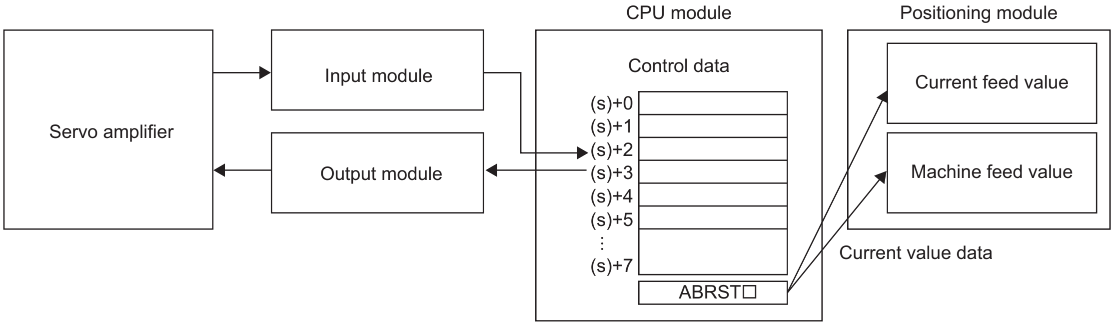
Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Signal received from the servo amplifier	Write the signal status, below, imported from the servo amplifier to the input module. • b0: ABS data bit 0 • b1: ABS data bit 1 • b2: Send data ready flag	0, 1	User
+3	Signal to be sent to the servo amplifier	Performs the operation with the dedicated instruction by using the "Signal received from the servo amplifier" in (s)+2. The ON/OFF state of the following data, output to the servo amplifier, is stored. • b0: Servo on • b1: ABS transfer mode • b2: ABS request flag	—	System
+4	Status	Status of communication with the servo amplifier • 0: Communication complete (set by the user at start of communication) • Other than 0: Now communicating (stored by the system)	0	User/system
+5 to +7	System area	—	—	—

Processing details

- The positioning data is read from the servo amplifier corresponding to the absolute position along the following target axis, and the value with the unit converted is stored in the "Current read value" and "Machine feed value" areas in the positioning module. For the absolute position detection system, restore the absolute position once when it is powered on or the CPU module is reset.

Instruction symbol	Target axis
G.ABRST1, Z.ABRST1	Axis 1
G.ABRST2, Z.ABRST2	Axis 2
G.ABRST3, Z.ABRST3	Axis 3
G.ABRST4, Z.ABRST4	Axis 4

- The following figure shows how the ABRST□ instruction operates.



- For communication with the servo amplifier corresponding to the absolute position (data read/write), the I/O module is used. To use the ABRST□ instruction, prepare I/O modules having the following number of points to communicate with servo amplifier per axis.
  - Input: 3 points
  - Output: 3 points
- The execution of the ABRST□ instruction and whether it has been completed normally or with an error can be checked with the completion device (d) or completion status indication device (d)+1.
  - Completion device (d)

This device turns on during the END processing of the scan where the ABRST□ instruction completed, and turns off during the next END processing.

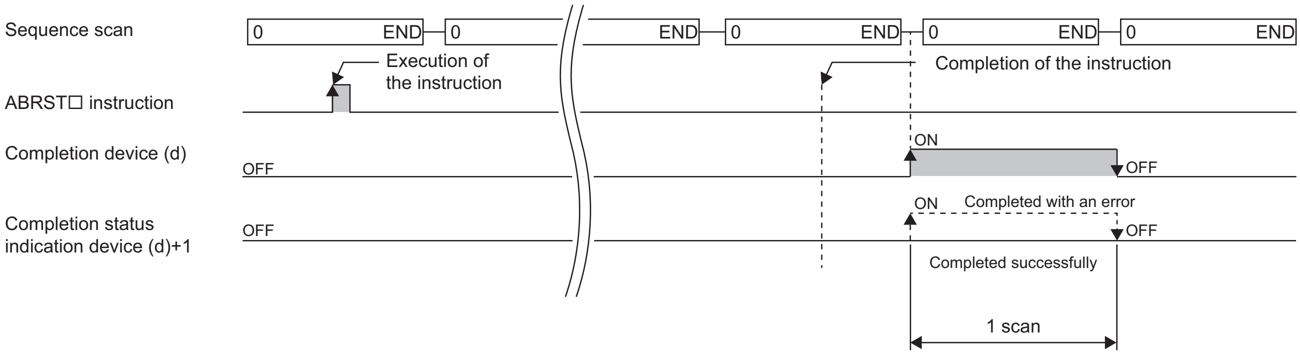
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the ABRST□ instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during the END processing of the scan where the ABRST□ instruction completed, and turns off during the next END processing (the same on/off operation as the completion device is performed).

- The following figure shows the operation at completion of the ABRST□ instruction.



- Completion of absolute position restoration can be checked with the (s)+4 status.
- The ABRST□ instruction performs absolute position restoration by following procedure.

1. Output data from ((s)+3).

Servo on, ABS transfer mode, and ABS request flag are output to the output module by using the program.

2. Set data in ((s)+2).

ABS data bit 0/bit 1 and the send data ready flag are set using the program.

3. Execute the ABRST□ instruction.

4. Check whether the value in ((s)+4) is 0 or not.

When the value is other than 0, the processing returns to step 1.

When the value is 0, the processing ends.

Precautions

- If the absolute position detection system has been constructed, absolute position restoration must be performed once after the power is turned on or reset. The servo does not turn on until the absolute position restoration completes with the positioning module.
- Absolute position restoration must be performed while the programmable controller ready signal [Y0] is off.
- The absolute position can be restored (the ABRST□ instruction can be executed) while a servo amplifier is operable. Note that when the absolute position is restored, the Servo on signal may turn off (servo off) during the period of the scan time plus approximately 60ms and the module may operate. To restore the absolute position during the servo off state, install an electromagnetic brake separately so that signals are output to the electromagnetic brake while the ABRST□ instruction is being executed.
- The following instructions cannot be executed simultaneously to a single axis. For different axes, any of the following can be executed concurrently with a G.ABRST instruction.
  - Positioning start instruction (PSTRT□)
  - Absolute position restoration instruction (ABRST□)
  - Teaching instruction (TEACH□)

Operation error

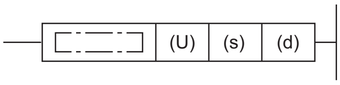
Error code ((s)+1)	Description
1860H	A value other than 0 was set in "Status" of (s)+4 (at start of communication with the servo amplifier).
1861H	"Status" of (s)+4 was changed during absolute position restoration (i.e. during communication with the servo amplifier).
1865H	An instruction was specified for an undefined axis (e.g. the G.ABRST3 instruction was specified when RD75P2 is used).

22.2 Starting the Positioning

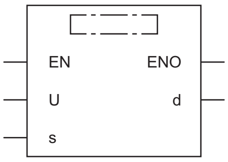
GP.PSTRT□, ZP.PSTRT□

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

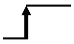
These instructions start to position the specified axis.

Ladder	ST
	ENO:=GP_PSTRT1(EN,U,s,d); ENO:=GP_PSTRT2(EN,U,s,d); ENO:=GP_PSTRT3(EN,U,s,d); ENO:=GP_PSTRT4(EN,U,s,d); ENO:=ZP_PSTRT1(EN,U,s,d); ENO:=ZP_PSTRT2(EN,U,s,d); ENO:=ZP_PSTRT3(EN,U,s,d); ENO:=ZP_PSTRT4(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.PSTRT1 GP.PSTRT2 GP.PSTRT3 GP.PSTRT4 ZP.PSTRT1 ZP.PSTRT2 ZP.PSTRT3 ZP.PSTRT4	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.PSTRT□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
	ZP.PSTRT□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device where the control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device to be turned on one scan upon completion of instruction If the instruction is completed with an error, (d)+1 is also turned on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.PSTRT□	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.PSTRT□	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*3	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 T, ST, C, and FD cannot be used.

*3 FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Start number	Specifies the number of the following data that is started with the PSTRT□ instruction. • Positioning data number: 1 to 600 • Block start: 7000 to 7004 • Machine OPR: 9001 • Fast OPR: 9002 • Present value change: 9003 • Multiple axes concurrent start: 9004	1 to 600 7000 to 7004 9001 to 9004	User

Processing details

- The positioning is started for the specified axes below.

Instruction symbol	Target axis
GP.PSTRT1, ZP.PSTRT1	Axis 1
GP.PSTRT2, ZP.PSTRT2	Axis 2
GP.PSTRT3, ZP.PSTRT3	Axis 3
GP.PSTRT4, ZP.PSTRT4	Axis 4

- Block start, origin return start, present value change start, and multiple axes concurrent start are available by specifying one of 7000 to 7004 or 9001 to 9004 with "Start number" in (s)+2.
- The execution of the RSTRT□ instruction and whether it has been completed normally or with an error can be checked with the completion device (d) or completion status indication device (d)+1.

- Completion device (d)

This device turns on during the END processing of the scan where the PSTRT□ instruction completed, and turns off during the next END processing.

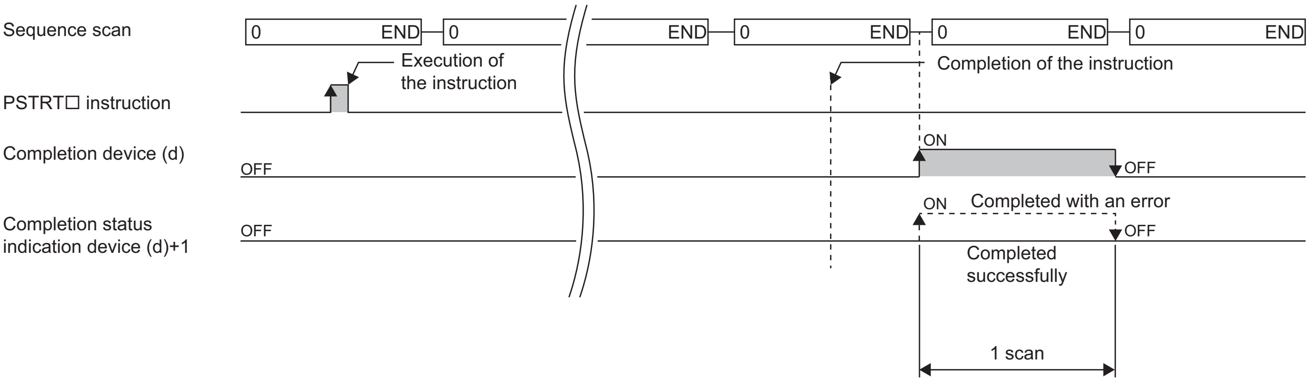
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the PSTRT□ instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during the END processing of the scan where the PSTRT□ instruction completed, and turns off during the next END processing (the same on/off operation as the completion device is performed).

- The following figure shows the operation at completion of the PSTRT□ instruction.



Precautions

- If the positioning is started by using the PSTRT□ instruction, the Position start signal [Y10, Y11, Y12, Y13] will not turn on. In this case, although the start completion signals [X10, X11, X12, X13] turn on, the ON time is short; the program may fail to detect the ON state. For this reason, start completion cannot be checked using the start completion signals [X10, X11, X12, X13]. Check the positioning control status with the start command of the PSTRT□ instruction or the BUSY signal [XC, XD, XE, XF].
- If the positioning is started by using the PSTRT□ instruction, and then the stop command is input before the positioning completes, the completion device (d) turns on for one scan and the execution of the PSTRT□ instruction completes.
- The following instructions cannot be executed simultaneously to a single axis. For different axes, any of the following can be executed concurrently with a G.ABRST instruction.
  - Positioning start instruction (PSTRT□)
  - Absolute position restoration instruction (ABRST□)
  - Teaching instruction (TEACH□)
- The PSTRT□ instruction is executed when the RD75 READY signal [X0] is on. While the RD75 READY signal [X0] is off, the instruction cannot be executed even though it is requested. (No processing is performed.) Before executing the instruction, turn on both the PLC READY signal [Y0] and the RD75 READY signal [X0].
- If multiple axes concurrent start is executed using the PSTRT□ instruction, the completion device (d) turns on when the positioning completes for the axis where the PSTRT□ instruction has been executed (e.g. Axis 1 for GP.PSTRT1).
- If the PSTRT□ instruction is used, the starting time will delay 0 to 0.88ms relative to the Positioning start signal [Y10, Y11, Y12, Y13].

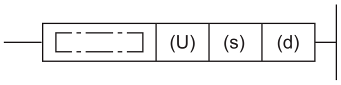
Operation error

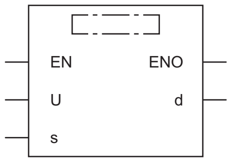
Error code ((s)+1)	Description
1862H	A value other than 1 to 600, 7000 to 7004, or 9001 to 9004 was set in "Starting number" of (s)+2.
1865H	An instruction was specified for an undefined axis (e.g. the GP.PSTRT3 instruction was specified when RD75P2 is used).

GP.TEACH□, ZP.TEACH□

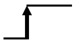
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform teaching for the specified axis.

Ladder	ST
	ENO:=GP_TEACH1(EN,U,s,d); ENO:=GP_TEACH2(EN,U,s,d); ENO:=GP_TEACH3(EN,U,s,d); ENO:=GP_TEACH4(EN,U,s,d); ENO:=ZP_TEACH1(EN,U,s,d); ENO:=ZP_TEACH2(EN,U,s,d); ENO:=ZP_TEACH3(EN,U,s,d); ENO:=ZP_TEACH4(EN,U,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
GP.TEACH1 GP.TEACH2 GP.TEACH3 GP.TEACH4 ZP.TEACH1 ZP.TEACH2 ZP.TEACH3 ZP.TEACH4	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.TEACH□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
	ZP.TEACH□	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device where the control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device to be turned on one scan upon completion of instruction If the instruction is completed with an error, (d)+1 is also turned on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

Operand		Bit		Word			Double word		Indirect specification	Constant			Others
		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	$	
(U)	GP.TEACH□	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.TEACH□	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*3	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, C, and FD cannot be used.
*3 FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Teaching data selection	Selects the address to which to write the current feed value (positioning or arc address). • 0: Writes the current feed value to the positioning address. • 1: Writes the current feed value to the arc address.	0, 1	User
+3	Positioning data number	Sets the positioning data number at which to perform the teaching.	1 to 600	User

Processing details

- The data of "[Md.20] Current feed value" of one of the following target axes is set in the positioning or arc address. The positioning data other than the positioning or arc address must be set using the engineering tool or program.

Instruction symbol	Target axis
GP.TEACH1, ZP.TEACH1	Axis 1
GP.TEACH2, ZP.TEACH2	Axis 2
GP.TEACH3, ZP.TEACH3	Axis 3
GP.TEACH4, ZP.TEACH4	Axis 4

- Positioning data No.1 to 600 can be taught.
- To move to an address (position) to be set as the positioning or arc address of the positioning data, JOG, inching, or manual pulse operation is used.
- The execution of the TEACH□ instruction and whether it has been completed normally or with an error can be checked with the completion device (d) or completion status indication device (d)+1.

- Completion device (d)

This device turns on during the END processing of the scan where the TEACH□ instruction completed, and turns off during the next END processing.

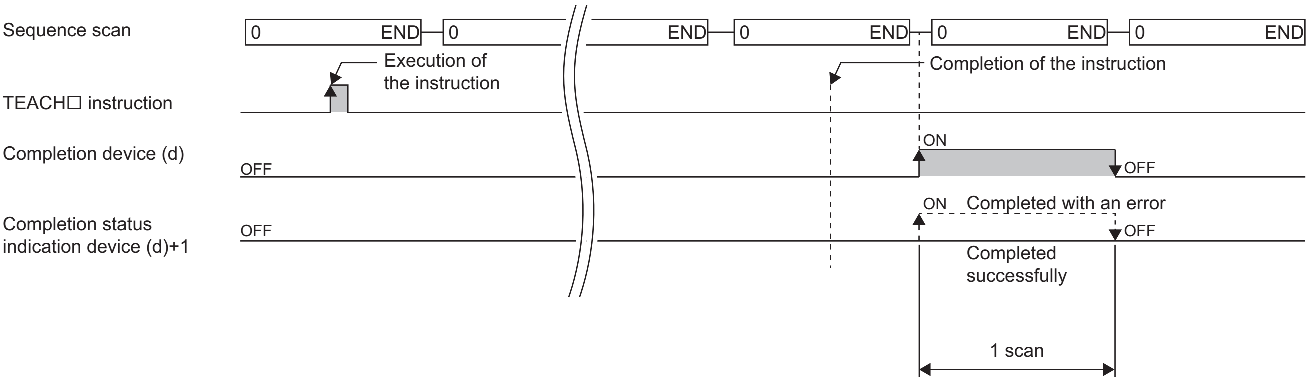
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the TEACH□ instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during the END processing of the scan where the TEACH□ instruction completed, and turns off during the next END processing (the same on/off operation as the completion device is performed).

- The following figure shows the operation at completion of the TEACH☐ instruction.



Precautions

- For a single axis, any of the following instructions cannot be executed concurrently with a G.ABRST instruction. although different axes, any of the following can be executed concurrently with a GP.TEACH instruction.
  - Positioning start instruction (PSTRT☐)
  - Absolute position restoration instruction (ABRST☐)
  - Teaching instruction (TEACH☐)
- The TEACH☐ instruction is executed when the BUSY signal [XC, XD, XE, or XF] is off. While the BUSY signal [XC, XD, XE, XF] is on, the instruction cannot be executed. (No processing is performed.) Before executing the instruction, check that the BUSY signal [XC, XD, XE, XF] of the corresponding axis is off.

Operation error

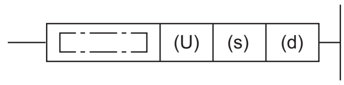
Error code ((s)+1)	Description
1863H	A value other than 0 or 1 was set in "Teaching data selection" of (s)+2.
1864H	A value other than 1 to 600 was set in "Positioning data number" of (s)+3
1865H	An instruction was specified for an undefined axis (e.g. the GP.TEACH3 instruction was specified when RD75P2 is used).

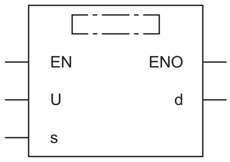
22.4 Backing up Module Data (Writing Data to the Flash ROM)

GP.PFWRT, ZP.PFWRT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These instructions write the module extension parameters (positioning data and block start data) in the buffer memory to the module extension parameter file.

Ladder	ST
	ENO:=GP_PFWRT(EN,U,s,d); ENO:=ZP_PFWRT(EN,U,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
GP.PFWRT ZP.PFWRT	↑

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.PFWRT	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
	ZP.PFWRT	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device where the control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device to be turned on one scan upon completion of instruction If the instruction is completed with an error, (d)+1 is also turned on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.PFWRT	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.PFWRT	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○[3]	—	—	—	—	○	—	—	—	—
(d)		○[1]	—	○[2]	—	—	—	—	—	—	—	—	—

[1] FX and FY cannot be used.

[2] T, ST, C, and FD cannot be used.

[3] FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System

Processing details

- These instructions write the module extension parameters in the buffer memory to the module extension parameter file. For details, refer to the following.

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

- The execution of the PFWRT instruction and whether it has been completed normally or with an error can be checked with the completion device (d) or completion status indication device (d)+1.

• Completion device (d)

This device turns on during the END processing of the scan where the PFWRT instruction completed, and turns off during the next END processing.

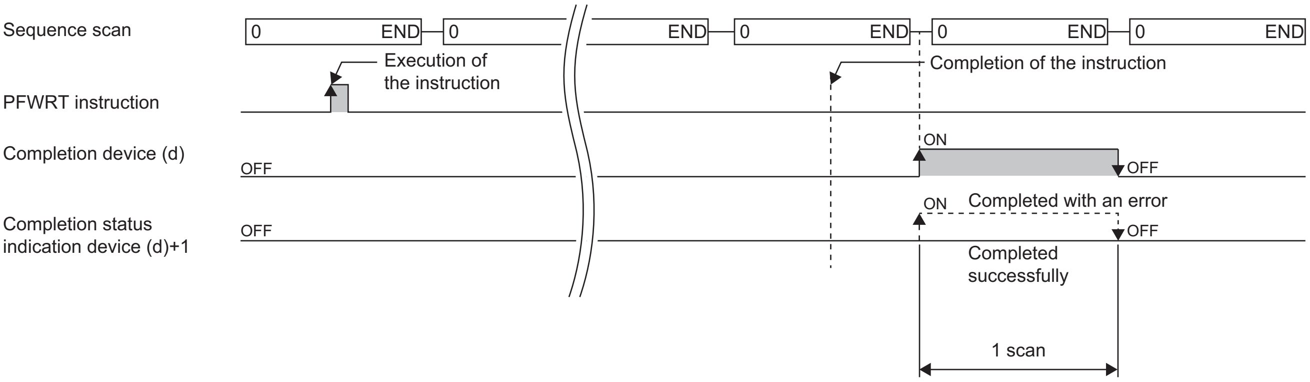
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the PFWRT instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during the END processing of the scan where the PFWRT instruction completed, and turns off during the next END processing (the same on/off operation as the completion device is performed).

- The following figure shows the operation at completion of the PFWRT instruction.



Precautions

- While the module extension parameters are being written by using the PFWRT instruction, do not power off the system or reset the CPU module. If the power is turned off or the CPU module is reset while the module extension parameters are being written, data is not normally written and normal positioning start is disabled. If normal positioning start is disabled, restart the system as follows.

Method	Description
Restart using the engineering tool	Write the positioning and block start data to the positioning module from the "Write to programmable controller" of the engineering tool.
Restart using the program	After initializing parameters by using the PINIT instruction, set the module extension parameters in the buffer memory of the positioning module. Thereafter, execute the PFWRT instruction.

- Data write to the flash ROM of the positioning module can repeat a maximum of one hundred thousand times. Any attempt to write data to the flash memory beyond this count results in failure.
- After the power is turned on or the CPU module is reset once, data write can repeat a maximum of 25 times if the program is used. Any attempt to write data to the flash ROM memory beyond 25 times results in a flash ROM write count error at error code 1080H, resulting in failure in data write. If one try of write results in a flash ROM write count error, check and modify the write program. If a flash ROM write count error occurred, reset the error of the positioning module using "[Cd.5] Axis error reset," or turn on the power or reset the CPU module again.
- The PFWRT instruction is executed when the RD75 READY signal [X0] is off. While the RD75 READY signal [X0] is on, the instruction cannot be executed. Before executing the instruction, turn off both the PLC READY signal [Y0] and the RD75 READY signal [X0].

Operation error

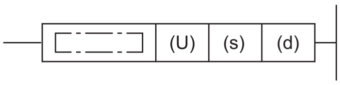
Error code ((s)+1)	Description
1080H	Flash ROM write count error

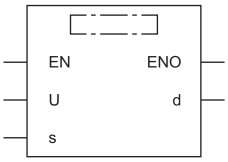
22.5 Initializing the Module

GP.PINIT, ZP.PINIT

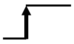
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions reset the module parameters and module extension parameters (positioning data and block start data) in the buffer memory, and the settings in the module extension parameter file to the factory default settings (initial values).

Ladder	ST
	ENO:=GP_PINIT(EN,U,s,d); ENO:=ZP_PINIT(EN,U,s,d);

FBD/LD


■Execution condition

Instruction	Execution condition
GP.PINIT ZP.PINIT	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.PINIT	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	16-bit unsigned binary	ANY16
	ZP.PINIT	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	00H to FEH	String	ANY16_OR_STRING_SINGLE
(s)		Start device where the control data is stored	Refer to the control data.	Device name	ANY16[1]
(d)		Device to be turned on one scan upon completion of instruction If the instruction is completed with an error, (d)+1 is also turned on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	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

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	$	
(U)	GP.PINIT	—	—	○	—	—	—	—	○	○	—	—	○
	ZP.PINIT	—	—	○	—	—	—	—	○	—	—	○	○
(s)		—	—	○*3	—	—	—	—	○	—	—	—	—
(d)		○*1	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, C, and FD cannot be used.
*3 FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The instruction completion status is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System

Processing details

- These instructions reset the module parameters and module extension parameters in the buffer memory of the positioning module, and the settings in the module extension parameter file to the factory default settings (initial values). For details, refer to the following.

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

- The setting data initialized include the parameters, positioning data (No. 1 to 600), and block start data (No. 7000 to 7004).
- The execution of the PINIT instruction and whether it has been completed normally or with an error can be checked with the completion device (d) or completion status indication device (d)+1.

- Completion device (d)

This device turns on during the END processing of the scan where the PINIT instruction completed, and turns off during the next END processing.

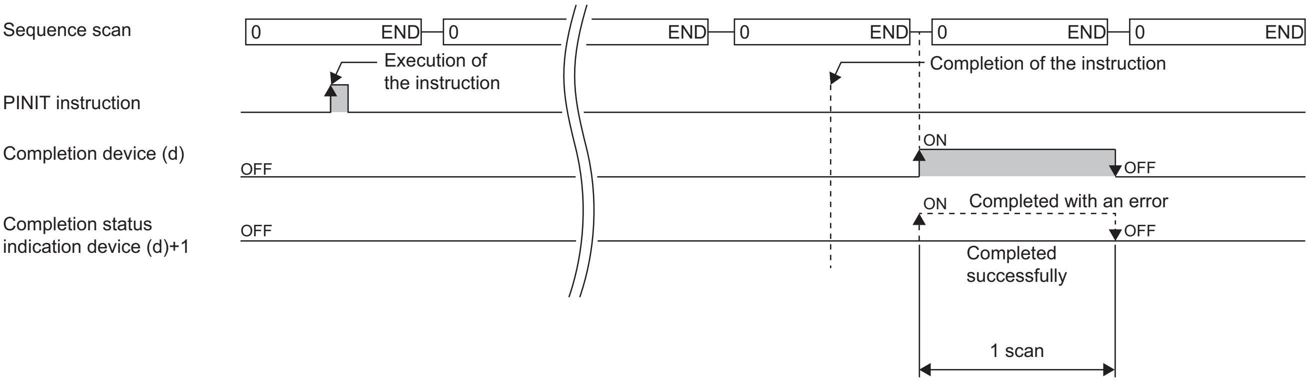
- Completion status indication device (d)+1

This device turns on or off depending on the completion status of the PINIT instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on during the END processing of the scan where the PINIT instruction completed, and turns off during the next END processing (the same on/off operation as the completion device is performed).

- The following figure shows the operation at completion of the PINIT instruction.



Precautions

- The PINIT instruction is executed when the RD75 READY signal [X0] is off. While the RD75 READY signal [X0] is on, the instruction cannot be executed. Before executing the instruction, turn off both the PLC READY signal [Y0] and the RD75 READY signal [X0].
- Data write to the flash ROM of the positioning module can repeat a maximum of one hundred thousand times. Any attempt to write data to the flash memory beyond this count results in failure.
- After the power is turned on or the CPU module is reset once, initialization can repeat a maximum of 25 times if the program is used. Any attempt to write data to the flash ROM memory beyond 25 times results in a flash ROM write count error at error code 1080H, resulting in failure in data write. If one try of initialization results in a flash ROM write count error, check and modify the write program. If a flash ROM write count error occurred, reset the error of the positioning module using "[Cd.5] Axis error reset," or turn on the power or reset the CPU module again.

Operation error

Error code ((s)+1)	Description
1080H	Flash ROM write count error
1931H	Flash ROM write error

23 High Speed Data Logger Module Instructions

Point

This chapter describes the instructions used commonly by MELSEC iQ-R series modules. When using MELSEC-Q series modules, refer to the manual for each module used and create programs.

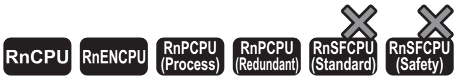
For precautions when using modules, refer to the following.

📖 MELSEC iQ-R Module Configuration Manual

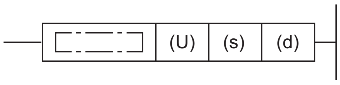
23.1 File Access Instructions

Recipe write

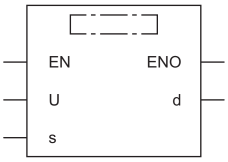
Z(P).RCPWRITE



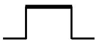
These instructions write the data in the CPU module to the specified recipe file in the SD memory card.

Ladder	ST
	ENO:=Z_RCPWRITE(EN,U,s,d); ENO:=ZP_RCPWRITE(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
Z.RCPWRITE	
ZP.RCPWRITE	

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	String	ANY16_OR_STRING_SINGLE
(s)	Start device where control data is stored	Page 1782 Control data	Device name	ANY16[*1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

The local device and the file register for each program are not available for setting data.

■Applicable devices

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	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s)	—	—	○*2	—	—	—	—	○	—	—	—	—
(d)	○*1	—	○*3	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.

*2 FD cannot be used.

*3 T, ST, C, and FD cannot be used.

■Control data

Operand: (s)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+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
+2	Record number	Record number of data to be written*1	1 to 256	User
+3 to +7	System area	—	—	—
+8 to +23	File name	Specifies the recipe name in which the data of the CPU module is written. *2(Up to 32 characters)	String	User
+24 to +31	System area	—	—	—

*1 Refer to the recipe file format. (📖MELSEC iQ-R High Speed Data Logger Module User's Manual(Application))

*2 Specify the recipe file name in ASCII code.

Processing details

- Specify the recipe file of the SD memory card and then write the data of the CPU module to the recipe file.
- A file from the RECIPE folder of the SD memory card can be specified as the recipe file.
- Normal/error completion of the Z(P).RCPWRITE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the Z(P).RCPWRITE instruction completes, and turns off during the next END processing.

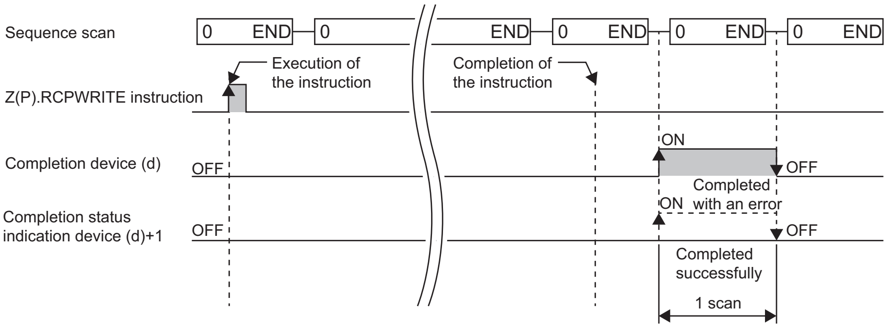
- Completion status indication device (d)+1

The completion device turns on or off depending on the completion status of the Z(P).RCPWRITE instruction.

When completed successfully: The device remains off.

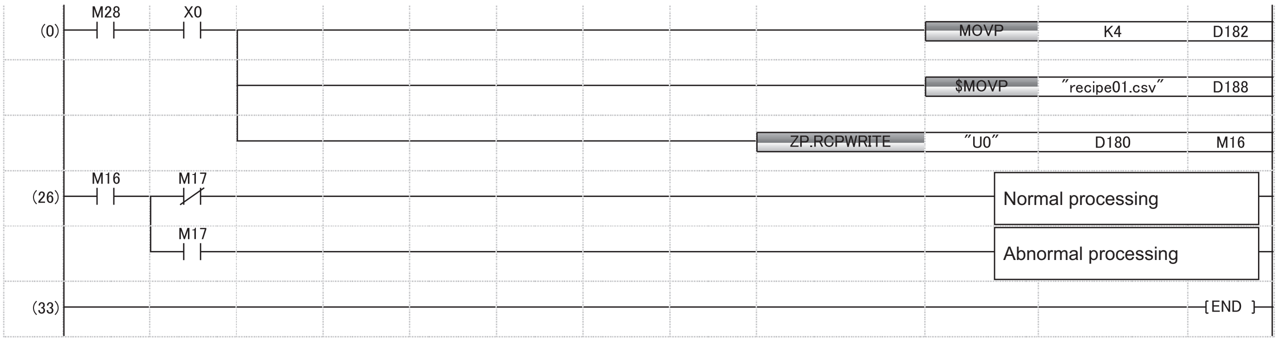
When completed with an error: The device turns on during END processing of the scan where the Z(P).RCPWRITE instruction completes, and turns off during the next END processing.

- The following figure shows how the Z(P).RCPWRITE instruction operates during execution.



■Program example

The following shows an example of a program in which the high speed data logger module mounted in the position of the I/O number X00 to X1F/Y00 to Y1F writes the data of the CPU module in the device value of record 4 of the file (recipe01.CSV) in the RECIPE folder, if M28 is turned ON.



Precautions

- The Z(P).RCPWRITE instruction cannot be executed additionally while another Z(P).RCPWRITE instruction is being executed. (If executed, no processing is performed.) For errors detected at the instruction execution, the completion device (d) and completion status indication device (d)+1 are not turned ON.
- The Z(P).RCPWRITE instruction cannot be executed when the module stop error is being occurred, or the data logging function cannot be performed (X5: OFF). (If attempted, an error occurs at the instruction completion.)
- An error may occur when the instruction (Z(P).RCPREAD) other than the Z(P).RCPWRITE instruction is accessing the same file. When accessing the same file, apply an interlock between the dedicated instructions.
- The Z(P).RCPWRITE instruction cannot be used in the interrupt program.

Operation error

Error code ((s)+1)	Description
1000H to 3FFFH	📖MELSEC iQ-R High Speed Data Logger Module User's Manual(Application)

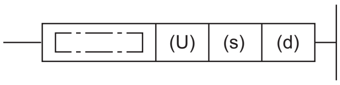
When the instruction completes with an error, the completion status indication device (d)+1 turns on and an error code is stored in the completion status (s)+1.

Recipe read

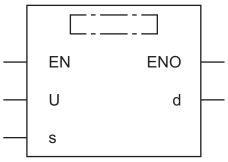
Z(P).RCPREAD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

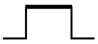
These instructions reads device values of the specified recipe file in the SD memory card to the CPU module.

Ladder	ST
	ENO:=Z_RCPREAD(EN,U,s,d); ENO:=ZP_RCPREAD(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
Z.RCPREAD	
ZP.RCPREAD	

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	String	ANY16_OR_STRING_SINGLE
(s)	Start device where control data is stored	Page 1785 Control data	Device name	ANY16[1]
(d)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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.

The local device and the file register for each program are not available for setting data.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	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	$	(U)
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s)	—	—	○[2]	—	—	—	—	○	—	—	—	—
(d)	○[1]	—	○[3]	—	—	—	—	—	—	—	—	—

[1] FX and FY cannot be used.

[2] FD cannot be used.

[3] T, ST, C, and FD cannot be used.

■Control data

Operand: (s)				
Device	**Item**	**Setting data**	**Setting range**	**Set by**
+0	System area	—	—	—
+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
+2	Record number	Record number of data to be read[*1]	1 to 256	User
+3 to +7	System area	—	—	—
+8 to +23	File name	Specifies the recipe name in which the data of the CPU module is read. [*2](Up to 32 characters)	String	User
+24 to +31	System area	—	—	—

[*1] Refer to the recipe file format. (📖MELSEC iQ-R High Speed Data Logger Module User's Manual(Application))

[*2] Specify the recipe file name in ASCII code.

Processing details

- Specify the recipe file of the SD memory card and then read the device value on the recipe file to the CPU module.
- A file from the RECIPE folder of the SD memory card can be specified as the recipe file.
- Normal/error completion of the Z(P).RCPREAD instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

- Completion device (d)

This device turns on during END processing of the scan where the Z(P).RCPREAD instruction completes, and turns off during the next END processing.

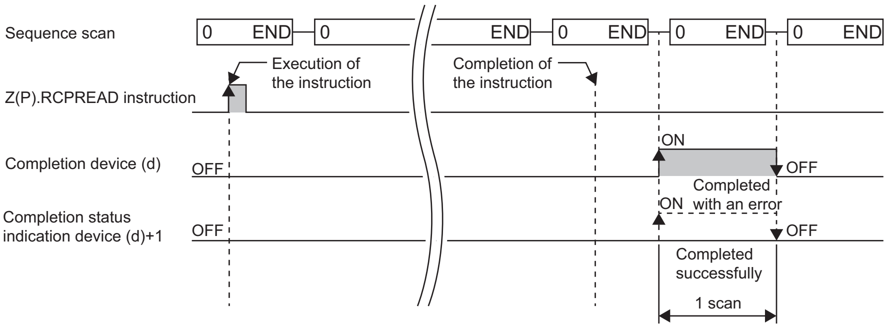
- Completion status indication device (d)+1

The completion device turns on or off depending on the completion status of the Z(P).RCPREAD instruction.

When completed successfully: The device remains off.

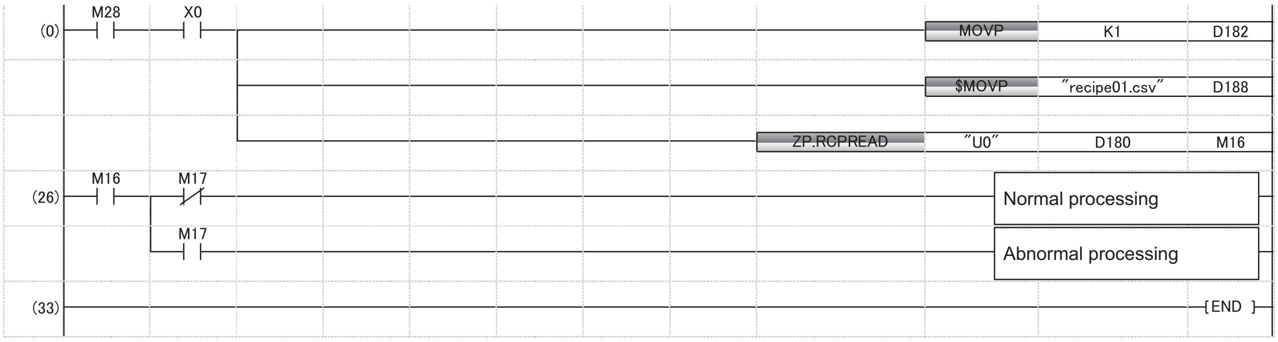
When completed with an error: The device turns on during END processing of the scan where the Z(P).RCPREAD instruction completes, and turns off during the next END processing.

- The following figure shows how the Z(P).RCPREAD instruction operates during execution.



■Program example

The following shows an example of a program in which the high speed data logger module mounted in the position of the I/O number X00 to X1F/Y00 to Y1F, reads the data of the device value of record 1 of the file (recipe01.CSV) in the RECIPE folder to the CPU module, if M28 is turned ON.



Precautions

- The Z(P).RCPREAD instruction cannot be executed additionally while another Z(P).RCPREAD instruction is being executed. (If executed, no processing is performed.) For errors detected at the instruction execution, the completion device (d) and completion status indication device (d)+1 are not turned ON.
- The Z(P).RCPREAD instruction cannot be executed when the module stop error is being occurred, or the data logging function cannot be performed (X5: OFF). (If attempted, an error occurs at the instruction completion.)
- An error may occur when the instruction (Z(P).RCPWRITE) other than the Z(P).RCPREAD instruction is accessing the same file. When accessing the same file, apply an interlock between the dedicated instructions.
- The Z(P).RCPREAD instruction cannot be used in the interrupt program.

Operation error

Error code ((s)+1)	Description
1000H to 3FFFH	📖MELSEC iQ-R High Speed Data Logger Module User's Manual(Application)

When the instruction completes with an error, the completion status indication device (d)+1 turns on and an error code is stored in the completion status (s)+1.

24 C INTELLIGENT FUNCTION MODULE INSTRUCTIONS

Point

For precautions when using modules, refer to the following.

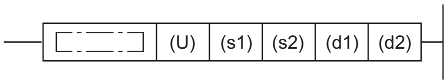
📖 MELSEC iQ-R Module Configuration Manual

24.1 User Function Execution Instruction

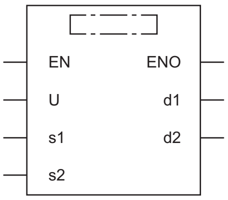
G(P).CEXECUTE

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) ✕ RnSFCPU (Standard) ✕ RnSFCPU (Safety)

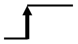
These instructions instruct to execute a function for the pre-registered function.

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 1788 Control data	Device name	ANY16
(s2)	Start device where request data is stored	—[1]	Device name	ANY16
(d1)	Start device for storing response data	—[1]	Device name	ANY16
(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.

■Applicable devices

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	$	
(U)[*1]	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(s2)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(d1)	—	—	○[*3]	—	—	—	—	○	—	—	—	—
(d2)	○[*2]	—	○[*4]	—	—	—	—	—	—	—	—	—

[*1] Index modification is not available.

[*2] FX and FY cannot be used.

[*3] FD cannot be used.

[*4] T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Allowable amount 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

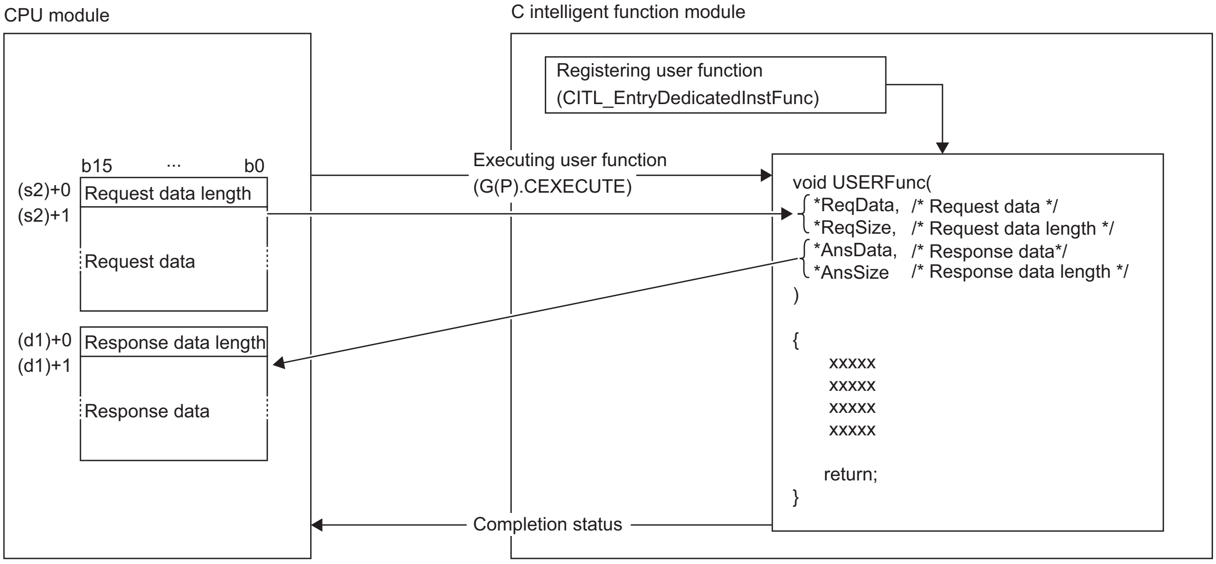
Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the request data length. (Number of words)	1 to 8192	User
+1 to +□	Request data	Specify the request data.	—	User

■Response data

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	Response data length is stored. (Number of words)	0 to 8192	System
+1 to +□	Response data	Response data is stored.	—	System

Processing details

- The request data stored in the device specified by (s2) and later is handed over to the C intelligent function 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 allowed number of response data specified in (s1), only the allowed number of response data will be stored and the remaining will be discarded.However, if the received response data is larger than the allowed number of response data specified in (s1), only the allowed number of response data will be stored and the remaining will be discarded. (Dedicated instruction will be completed successfully.) In this case, the response data length (d1) will be the number of data actually stored.



The user program runs in the following tasks settings.
- Task priority: 100
- Stack size: 40000 bytes
- Task option: VX_FP_TASK

- The completion status of the G(P).CEXECUTE instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

- Completion device (d2)

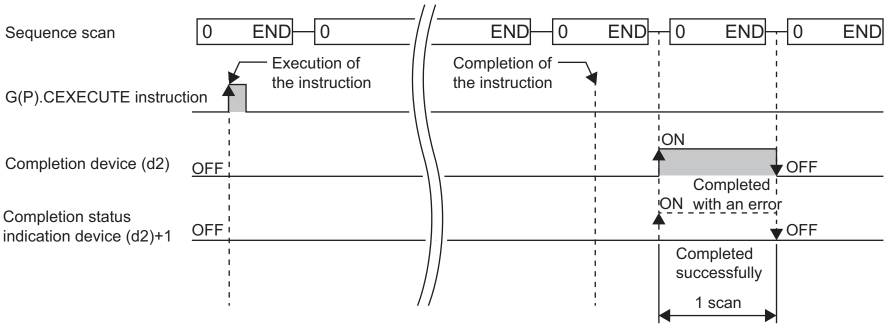
This device turns on during END processing of the scan where the G(P).CEXECUTE instruction completes, and turns off during the next END processing.

- Completion status indication device (d2)+1

When completed successfully: The device remains off.

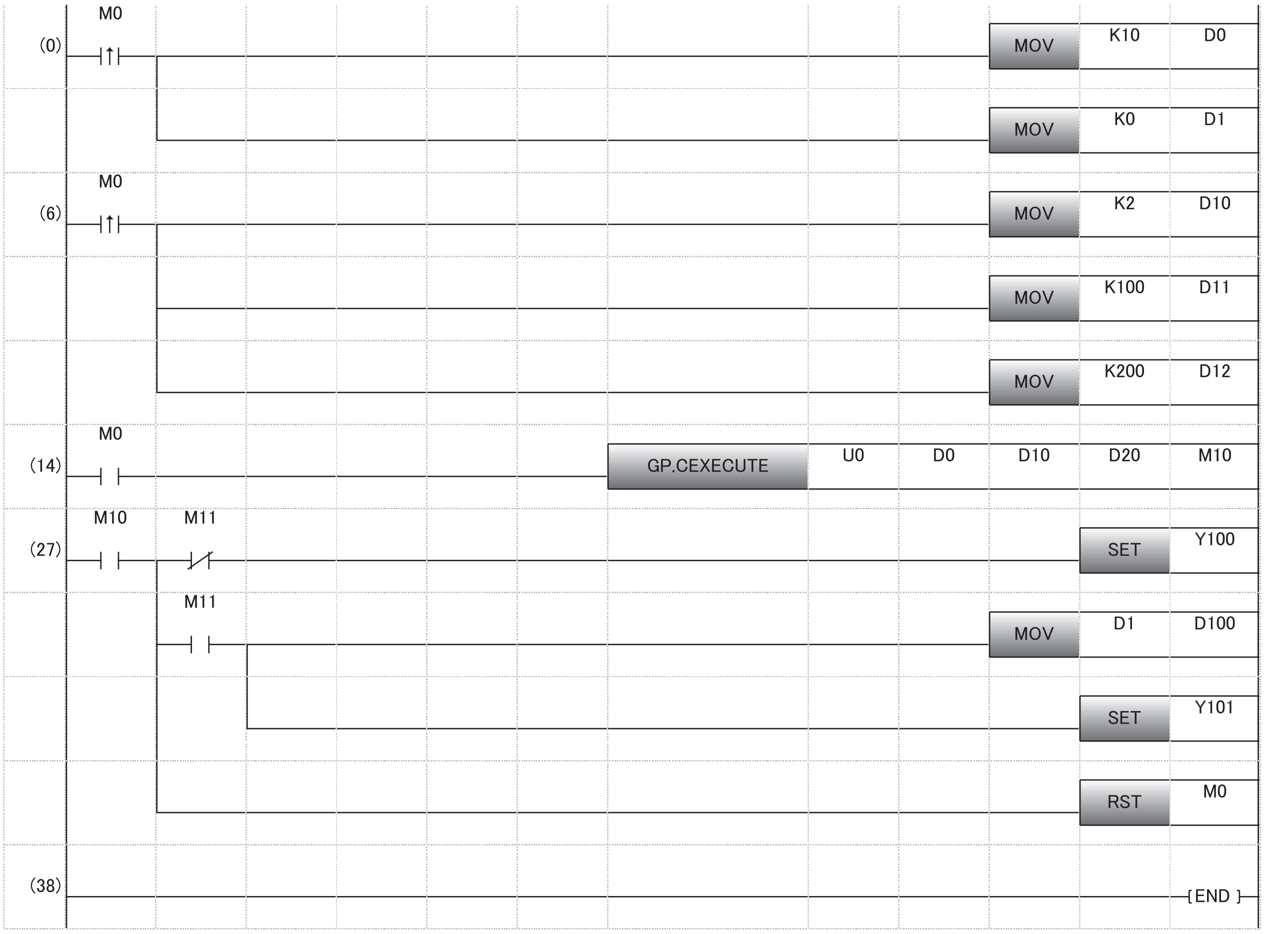
When completed with an error: The device turns on during END processing of the scan where the G(P).CEXECUTE instruction completes, and turns off during the next END processing.

- The following figure shows how the G(P).CEXECUTE instruction operates during execution.



■Program example

The following shows an example of a program that receives response data from D20 after the request data is set from D10 to D12.



Precautions

- Before executing the G(P).CEXECUTE instruction, the user function must be registered using the C intelligent function module dedicated function (CITL_EntryDedicatedInstFunc). If the user function is unregistered, a 'user function unregistered error' (1800H) will be returned. In addition, 1 user function can be executed by the G(P).CEXECUTE instruction. If you want to run multiple functions, prepare the data for identifying functions in request data, then call the functions distinctly using the conditional branches in user function.
- The G(P).CEXECUTE instruction cannot be executed additionally while another G(P).CEXECUTE instruction is being executed. (If attempted, the instruction is not processed.)
- The local device and the file register for each program are not available for setting data.
- Operand must be specified even when request data and response data are not required.

Operation error

Error code ((s1)+1)	**Description**
1000H to 3FFFH	📖 MELSEC iQ-R C Intelligent Function Module User's Manual (Application)

When the instruction completes with an error, the completion status indication device (d2)+1 turns on and an error code is stored in the completion status (s1)+1.

PART 5 STANDARD FUNCTIONS

This part consists of the following chapters.

25 TYPE CONVERSION FUNCTIONS

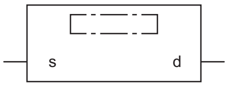
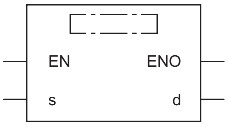
25.1 Converting BOOL to WORD

BOOL_TO_WORD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from BOOL data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BOOL_TO_WORD(s);
		[With EN/ENO]
		d:=BOOL_TO_WORD_E(EN,ENO,s);

Setting data

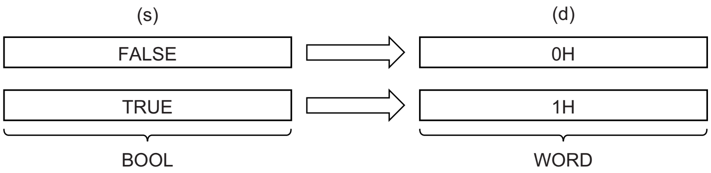
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

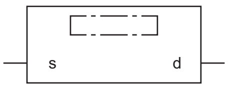
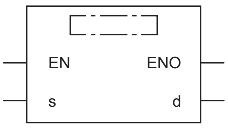
25.2 Converting BOOL to DWORD

BOOL_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=BOOL_TO_DWORD(s);
		[With EN/ENO]
		d:=BOOL_TO_DWORD_E(EN,ENO,s);

Setting data

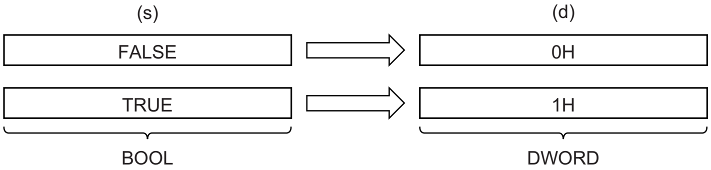
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

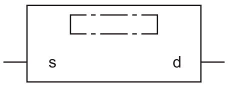
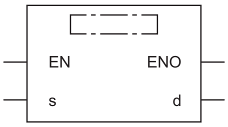
25.3 Converting BOOL to INT

BOOL_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=BOOL_TO_INT(s);
		[With EN/ENO]
		d:=BOOL_TO_INT_E(EN,ENO,s);

Setting data

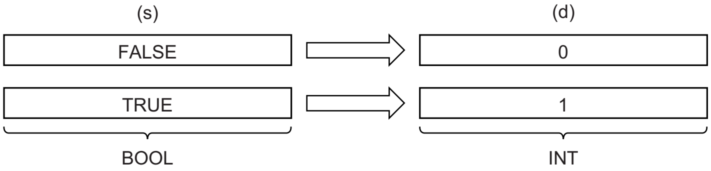
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

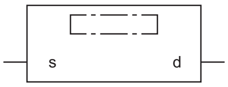
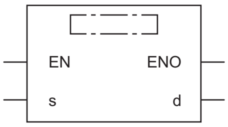
25.4 Converting BOOL to DINT

BOOL_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert a value from BOOL data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=BOOL_TO_DINT(s);
		[With EN/ENO]
		d:=BOOL_TO_DINT_E(EN,ENO,s);

Setting data

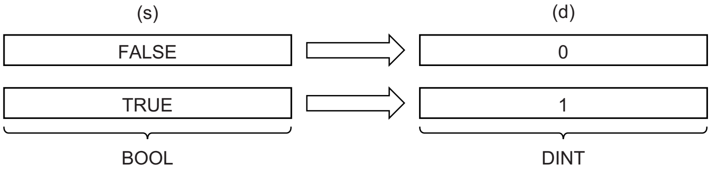
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

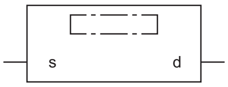
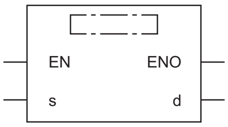
25.5 Converting BOOL to TIME

BOOL_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=BOOL_TO_TIME(s);
		[With EN/ENO]
		d:=BOOL_TO_TIME_E(EN,ENO,s);

Setting data

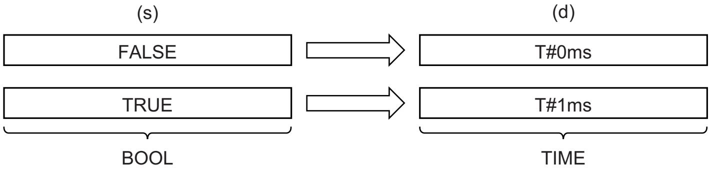
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

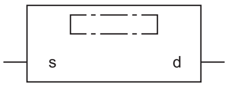
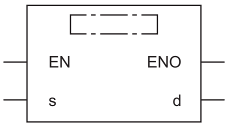
25.6 Converting BOOL to STRING

BOOL_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BOOL_TO_STRING(s);
		[With EN/ENO]
		d:=BOOL_TO_STRING_E(EN,ENO,s);

Setting data

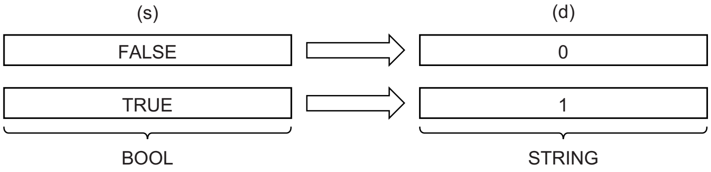
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	STRING

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to STRING data type, and output the converted value from (d).
- When the input value is FALSE, 0 (STRING data type) is output.
- When the input value is TRUE, 1 (STRING data type) is output.



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

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

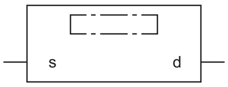
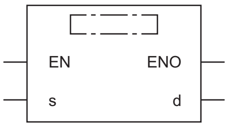
25.7 Converting WORD to BOOL

WORD_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from WORD data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=WORD_TO_BOOL(s);
		[With EN/ENO]
		d:=WORD_TO_BOOL_E(EN,ENO,s);

Setting data

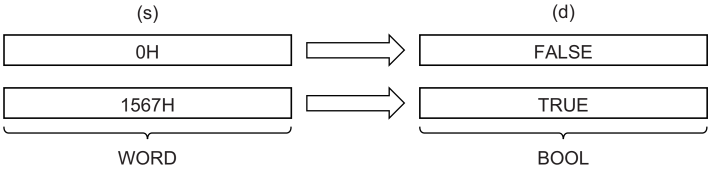
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

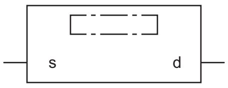
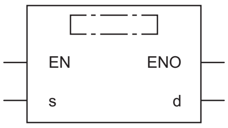
25.8 Converting WORD to DWORD

WORD_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from WORD data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=WORD_TO_DWORD(s);
		[With EN/ENO]
		d:=WORD_TO_DWORD_E(EN,ENO,s);

Setting data

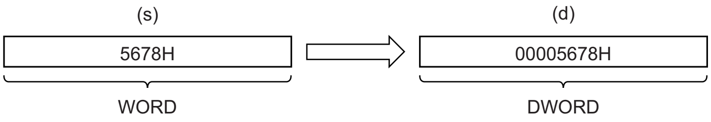
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

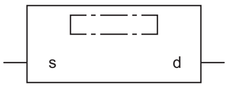
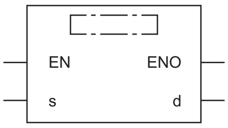
25.9 Converting WORD to INT

WORD_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from WORD data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=WORD_TO_INT(s);
		[With EN/ENO]
		d:=WORD_TO_INT_E(EN,ENO,s);

Setting data

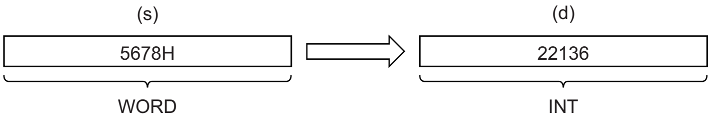
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

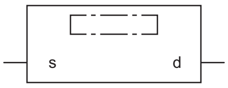
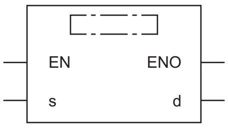
25.10 Converting WORD to DINT

WORD_TO_DINT(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from WORD data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=WORD_TO_DINT(s);
		[With EN/ENO]
		d:=WORD_TO_DINT_E(EN,ENO,s);

Setting data

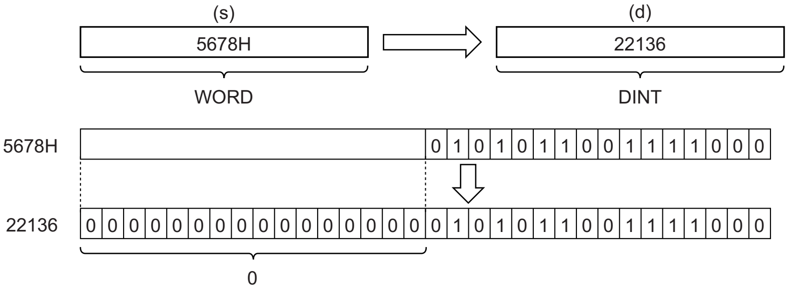
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

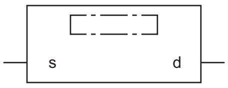
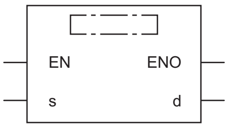
25.11 Converting WORD to TIME

WORD_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=WORD_TO_TIME(s);
		[With EN/ENO]
		d:=WORD_TO_TIME_E(EN,ENO,s);

Setting data

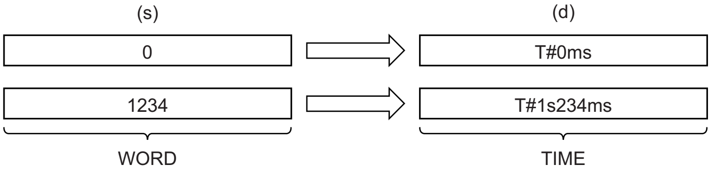
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

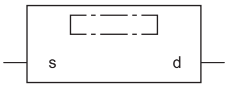
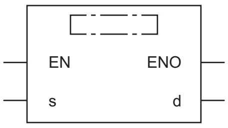
25.12 Converting WORD to STRING

WORD_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=WORD_TO_STRING(s);
		[With EN/ENO]
		d:=WORD_TO_STRING_E(EN,ENO,s);

Setting data

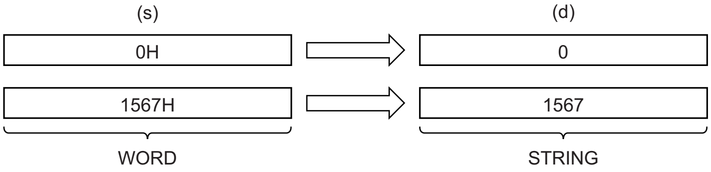
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(4)

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to STRING data type, and output the converted value from (d).



- Input a WORD data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

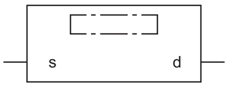
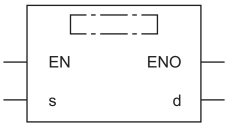
25.13 Converting DWORD to BOOL

DWORD_TO_BOOL(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from DWORD data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DWORD_TO_BOOL(s);
		[With EN/ENO]
		d:=DWORD_TO_BOOL_E(EN,ENO,s);

Setting data

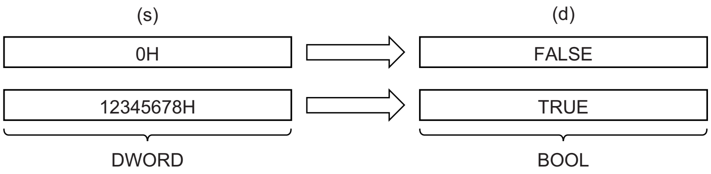
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

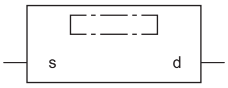
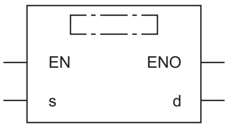
25.14 Converting DWORD to WORD

DWORD_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DWORD data type to WORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DWORD_TO_WORD(s);
		[With EN/ENO]
		d:=DWORD_TO_WORD_E(EN,ENO,s);

Setting data

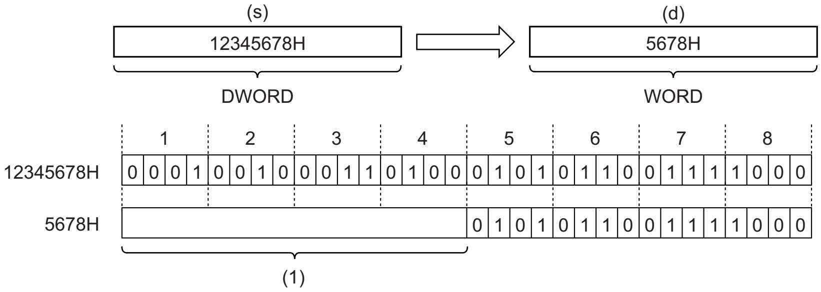
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

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

Operation error

There is no operation error.

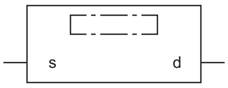
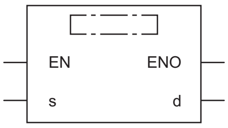
25.15 Converting DWORD to INT

DWORD_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DWORD data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DWORD_TO_INT(s);
		[With EN/ENO]
		d:=DWORD_TO_INT_E(EN,ENO,s);

Setting data

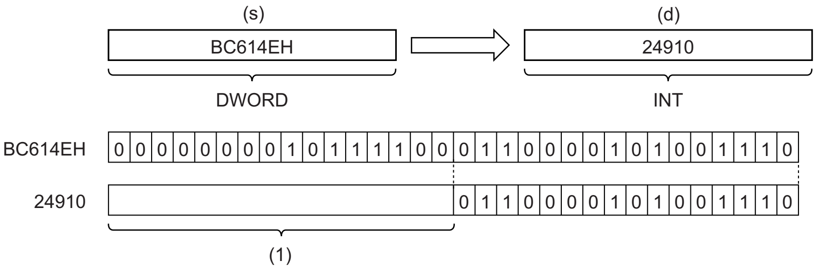
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

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

Operation error

There is no operation error.

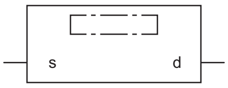
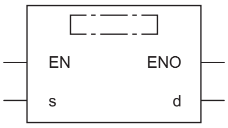
25.16 Converting DWORD to DINT

DWORD_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from DWORD data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DWORD_TO_DINT(s);
		[With EN/ENO]
		d:=DWORD_TO_DINT_E(EN,ENO,s);

Setting data

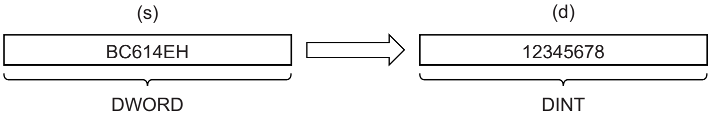
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

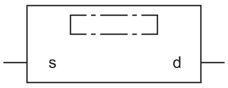
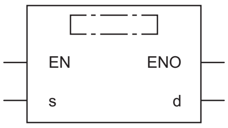
25.17 Converting DWORD to TIME

DWORD_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These functions convert a value from DWORD data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DWORD_TO_TIME(s);
		[With EN/ENO]
		d:=DWORD_TO_TIME_E(EN,ENO,s);

Setting data

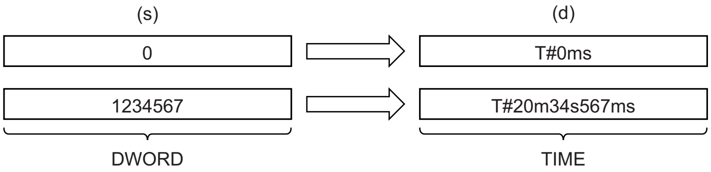
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

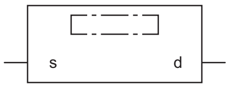
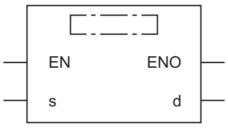
25.18 Converting DWORD to STRING

DWORD_TO_STRING(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions convert a value from DWORD data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DWORD_TO_STRING(s);
		[With EN/ENO]
		d:=DWORD_TO_STRING_E(EN,ENO,s);

Setting data

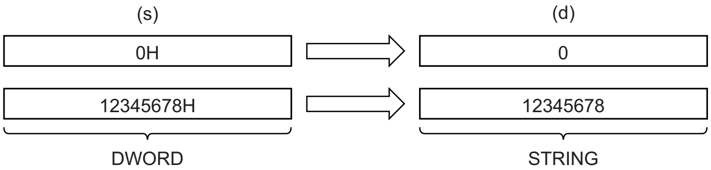
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(8)

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to STRING data type, and output the converted value from (d).



- Input a DWORD data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

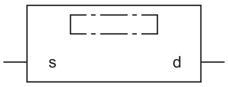
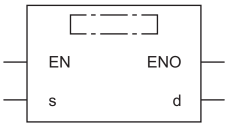
25.19 Converting INT to BOOL

INT_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=INT_TO_BOOL(s);
		[With EN/ENO]
		d:=INT_TO_BOOL_E(EN,ENO,s);

Setting data

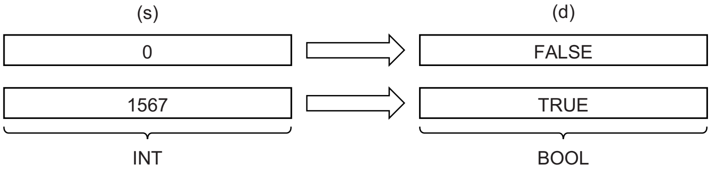
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

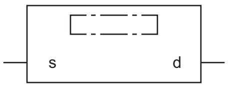
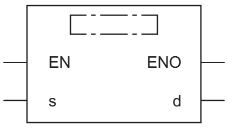
25.20 Converting INT to WORD

INT_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✕

These functions convert a value from INT data type to WORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_WORD(s);
		[With EN/ENO]
		d:=INT_TO_WORD_E(EN,ENO,s);

Setting data

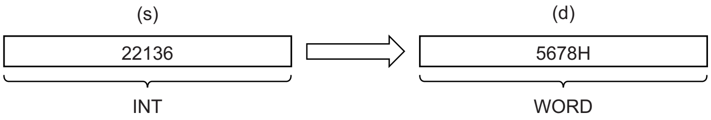
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

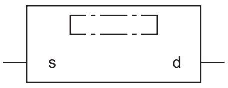
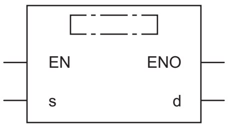
25.21 Converting INT to DWORD

INT_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=INT_TO_DWORD(s);
		[With EN/ENO]
		d:=INT_TO_DWORD_E(EN,ENO,s);

Setting data

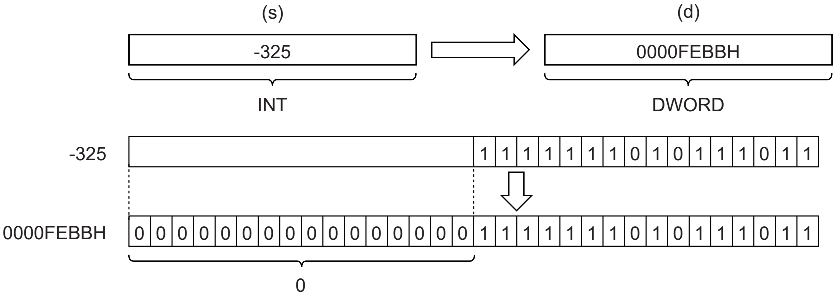
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

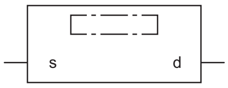
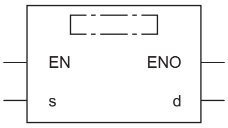
25.22 Converting INT to DINT

INT_TO_DINT(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions convert a value from INT data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_DINT(s);
		[With EN/ENO]
		d:=INT_TO_DINT_E(EN,ENO,s);

Setting data

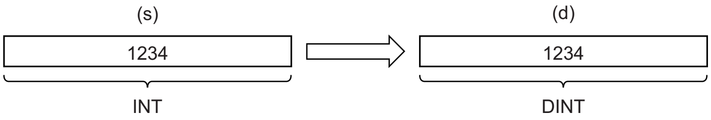
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

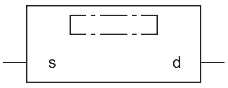
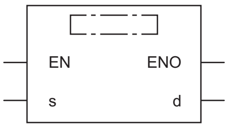
25.23 Converting INT to BCD

INT_TO_BCD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to BCD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_BCD(s);
		[With EN/ENO]
		d:=INT_TO_BCD_E(EN,ENO,s);

Setting data

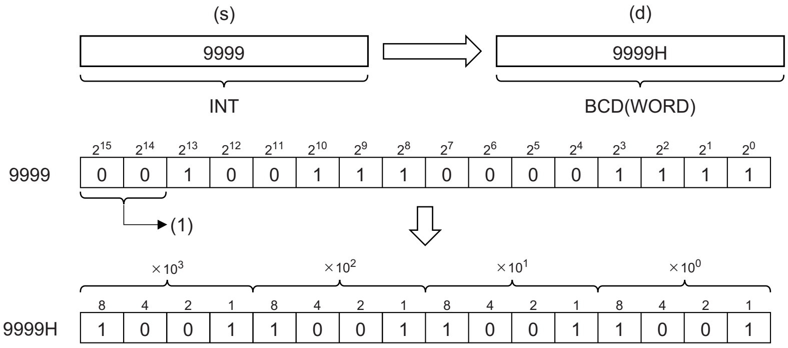
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to BCD (WORD) data type, and output the converted value from (d).



(1) Set 0s.

- Input an INT data type value to (s) within the range of 0 to 9999.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	Data input to (s) is out of the range, 0 to 9999.

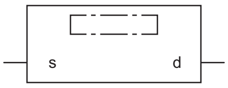
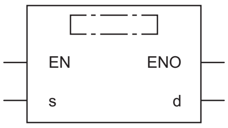
25.24 Converting INT to REAL

INT_TO_REAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These functions convert a value from INT data type to REAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_REAL(s);
		[With EN/ENO]
		d:=INT_TO_REAL_E(EN,ENO,s);

Setting data

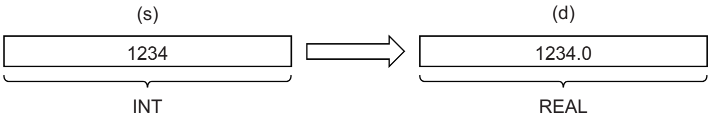
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

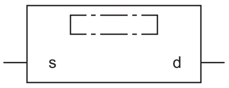
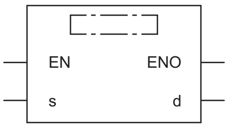
25.25 Converting INT to LREAL

INT_TO_LREAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to LREAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_LREAL(s);
		[With EN/ENO]
		d:=INT_TO_LREAL_E(EN,ENO,s);

Setting data

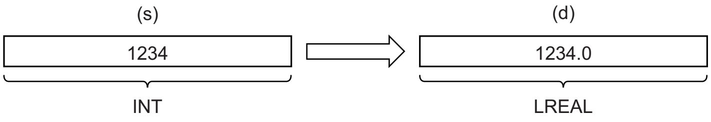
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

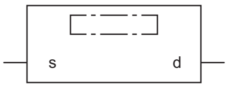
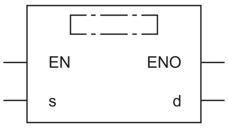
25.26 Converting INT to TIME

INT_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_TIME(s);
		[With EN/ENO]
		d:=INT_TO_TIME_E(EN,ENO,s);

Setting data

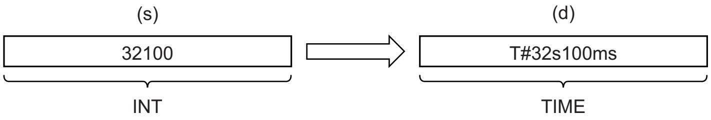
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

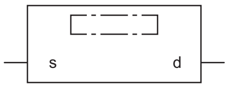
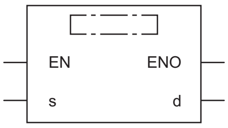
25.27 Converting INT to STRING

INT_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from INT data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=INT_TO_STRING(s);
		[With EN/ENO]
		d:=INT_TO_STRING_E(EN,ENO,s);

Setting data

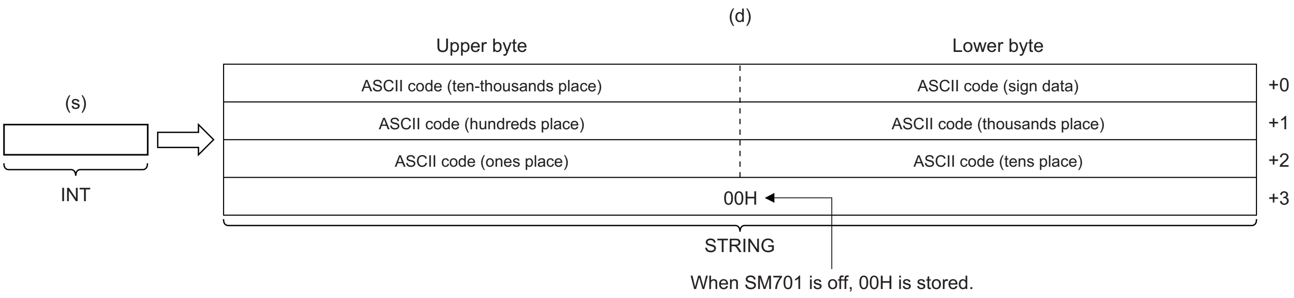
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(6)

Processing details

■Operation processing

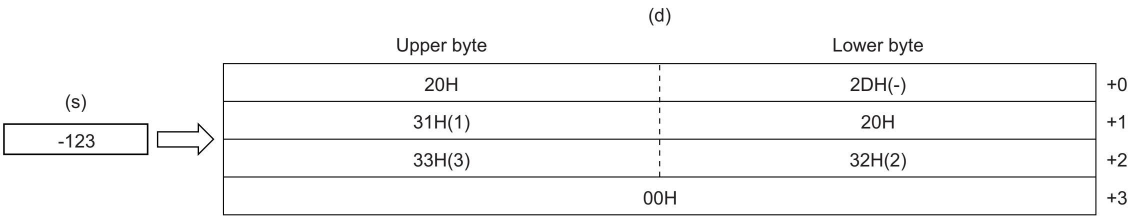
- These functions convert the value input to (s) from INT data type to STRING data type, and output the converted value from (d).



- Input an INT data type value to (s).
- As sign data, 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the value is negative.
- If the number of digits in the input value is less than the number of significant digits, 20H (space) is stored for the upper digit(s).

Ex.

When the value -123 is input



- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string (4th word).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

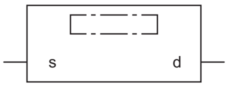
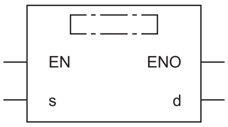
25.28 Converting DINT to BOOL

DINT_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DINT data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_BOOL(s);
		[With EN/ENO]
		d:=DINT_TO_BOOL_E(EN,ENO,s);

Setting data

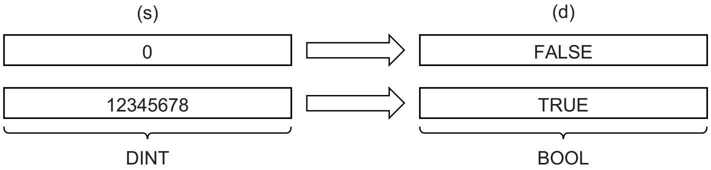
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

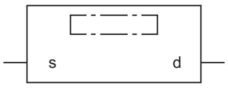
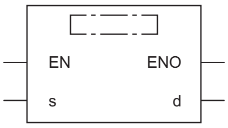
25.29 Converting DINT to WORD

DINT_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These functions convert a value from DINT data type to WORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=DINT_TO_WORD(s);
		[With EN/ENO]
		d:=DINT_TO_WORD_E(EN,ENO,s);

Setting data

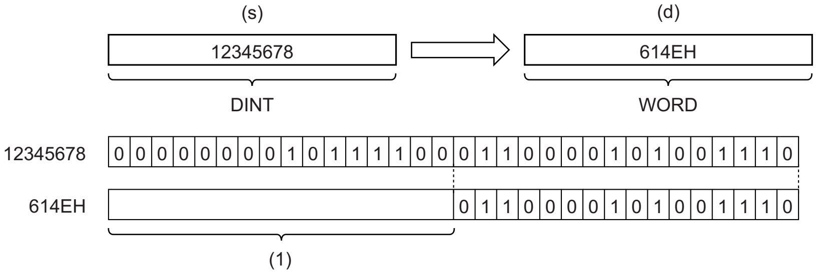
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

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

Operation error

There is no operation error.

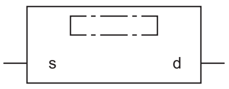
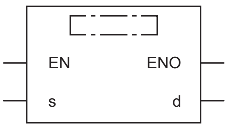
25.30 Converting DINT to DWORD

DINT_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DINT_TO_DWORD(s);
		[With EN/ENO]
		d:=DINT_TO_DWORD_E(EN,ENO,s);

Setting data

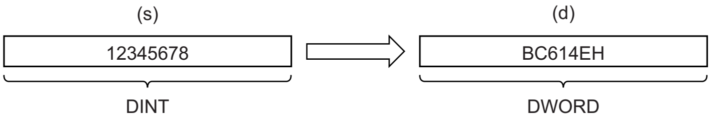
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

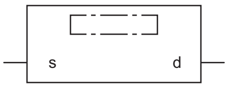
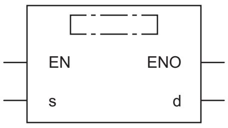
25.31 Converting DINT to INT

DINT_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from DINT data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_INT(s);
		[With EN/ENO]
		d:=DINT_TO_INT_E(EN,ENO,s);

Setting data

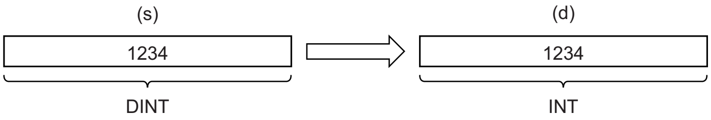
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.

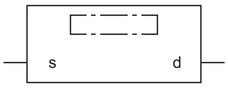
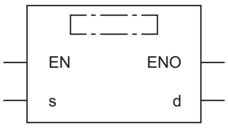
25.32 Converting DINT to BCD

DINT_TO_BCD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from DINT data type to BCD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_BCD(s);
		[With EN/ENO]
		d:=DINT_TO_BCD_E(EN,ENO,s);

Setting data

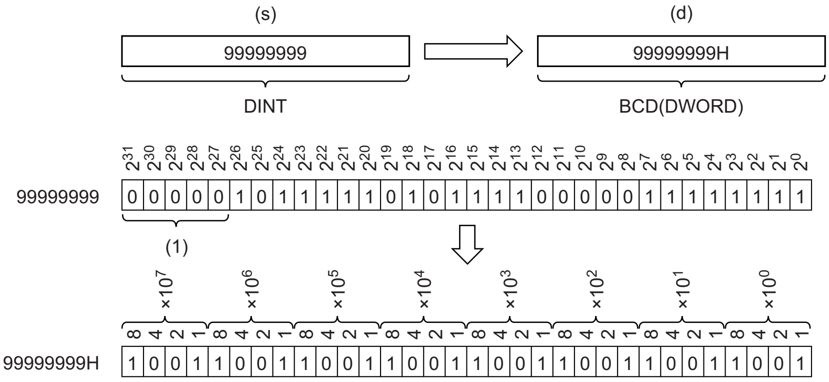
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to BCD (DWORD) data type, and output the converted value from (d).



(1) Set 0s.

- Input a DINT data type value to (s). When (d) is of WORD date type, the input value range is 0 to 9999. When (d) is of DWORD date type, the input value range is 0 to 99999999.
- WORD or DWORD data type can be specified for (d). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	**Operation result**	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (d) is of WORD data type

Error code (SD0)	**Description**
3401H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.
	Data input to (s) is out of the range, 0 to 9999.

- When (d) is of DWORD data type

Error code (SD0)	**Description**
3401H	Data input to (s) is out of the range, 0 to 99999999.

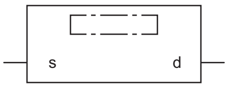
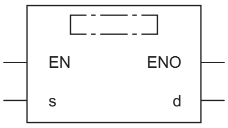
25.33 Converting DINT to REAL

DINT_TO_REAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from DINT data type to REAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_REAL(s);
		[With EN/ENO]
		d:=DINT_TO_REAL_E(EN,ENO,s);

Setting data

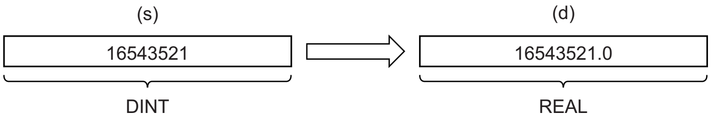
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

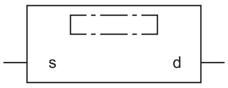
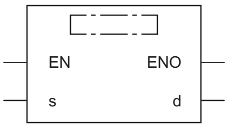
25.34 Converting DINT to LREAL

DINT_TO_LREAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to LREAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_LREAL(s);
		[With EN/ENO]
		d:=DINT_TO_LREAL_E(EN,ENO,s);

Setting data

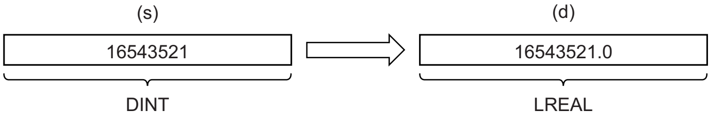
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

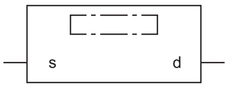
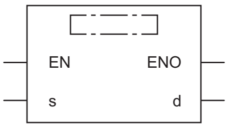
25.35 Converting DINT to TIME

DINT_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_TIME(s);
		[With EN/ENO]
		d:=DINT_TO_TIME_E(EN,ENO,s);

Setting data

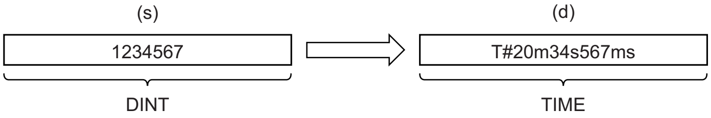
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

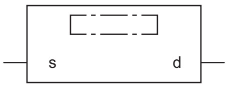
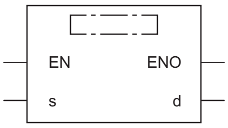
25.36 Converting DINT to STRING

DINT_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DINT data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DINT_TO_STRING(s);
		[With EN/ENO]
		d:=DINT_TO_STRING_E(EN,ENO,s);

Setting data

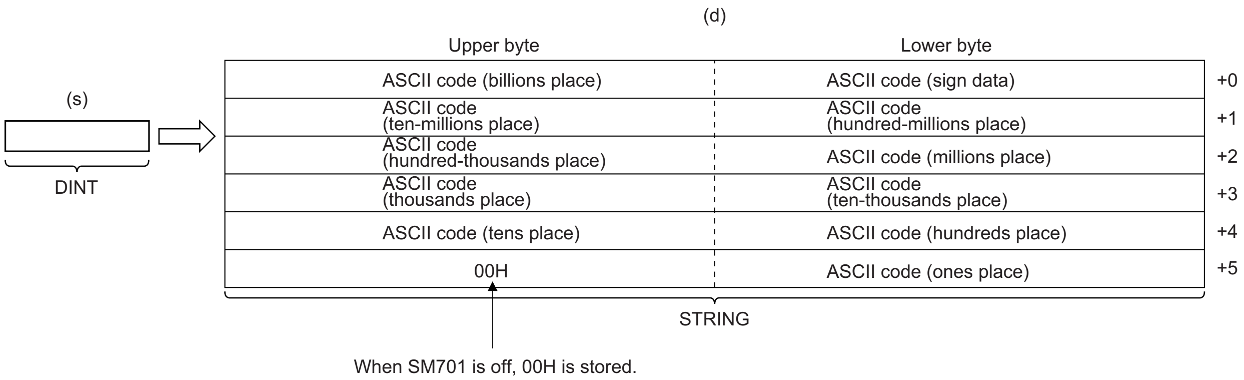
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(11)

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to STRING data type, and output the converted value from (d).



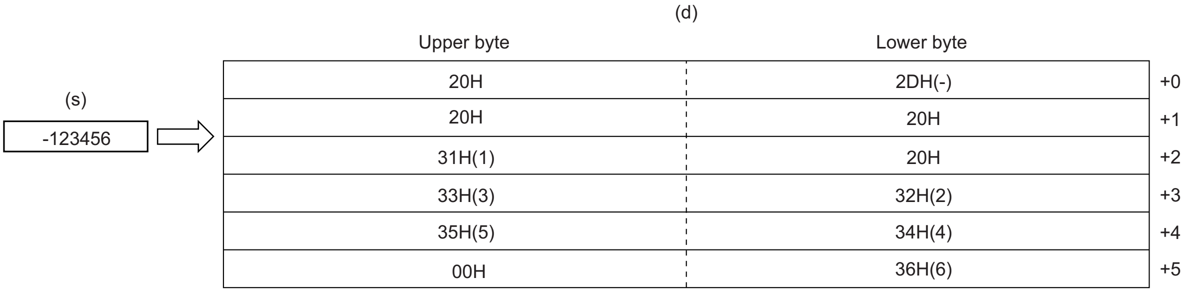
	Upper byte	Lower byte	
	ASCII code (billions place)	ASCII code (sign data)	+0
	ASCII code (ten-millions place)	ASCII code (hundred-millions place)	+1
	ASCII code (hundred-thousands place)	ASCII code (millions place)	+2
	ASCII code (thousands place)	ASCII code (ten-thousands place)	+3
	ASCII code (tens place)	ASCII code (hundreds place)	+4
	00H	ASCII code (ones place)	+5

When SM701 is off, 00H is stored.

- Input a DINT data type value to (s).
- As sign data, 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the value is negative.
- If the number of digits in the input value is less than the number of significant digits, 20H (space) is stored for the upper digit(s).

Ex.

When the value -123456 is input



	Upper byte	Lower byte	
	20H	2DH(-)	+0
	20H	20H	+1
	31H(1)	20H	+2
	33H(3)	32H(2)	+3
	35H(5)	34H(4)	+4
	00H	36H(6)	+5

- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string (upper bytes of the 6th word).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

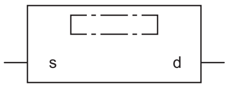
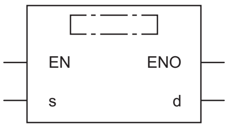
25.37 Converting BCD to INT

BCD_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BCD data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=BCD_TO_INT(s);
		[With EN/ENO]
		d:=BCD_TO_INT_E(EN,ENO,s);

Setting data

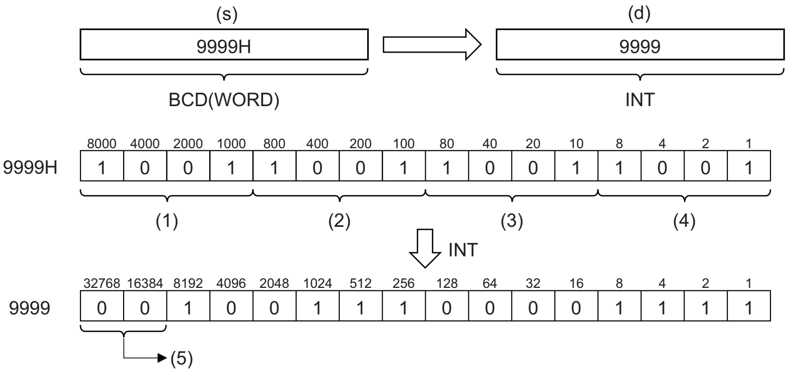
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from BCD (WORD) data type to INT data type, and output the converted value from (d).



(1) Thousands place
(2) Hundreds place
(3) Tens place
(4) Ones place
(5) Filled with 0s.

- Input a WORD data type value to (s) within the range of 0H to 9999H (range of each digit: 0 to 9).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	A value other than 0 to 9 exists at any digit of the value input to (s).

- Turning on SM754 can prevent the above error from being issued. If the specified value is out of the valid range, the BCD_TO_INT(_E) function is not executed regardless of the status (on/off) of SM754.

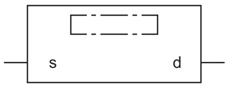
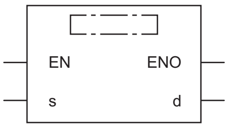
25.38 Converting BCD to DINT

BCD_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BCD data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BCD_TO_DINT(s);
		[With EN/ENO]
		d:=BCD_TO_DINT_E(EN,ENO,s);

Setting data

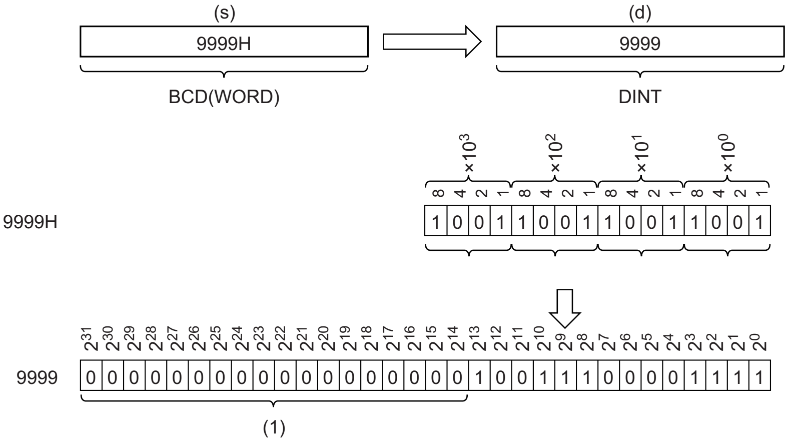
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

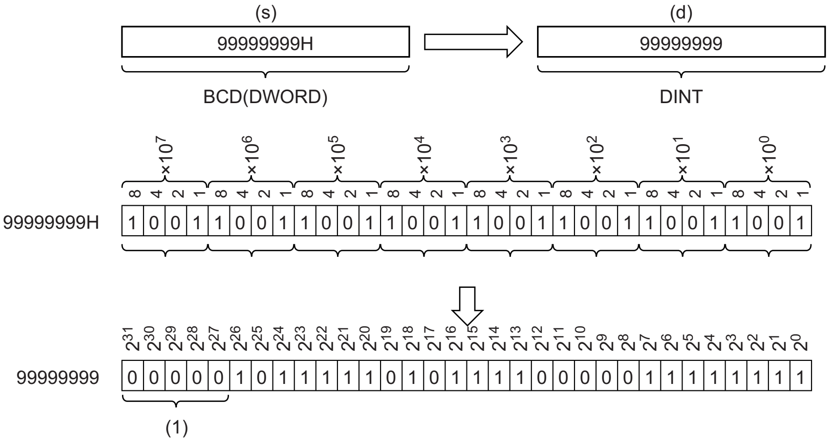
■Operation processing

- These functions convert the value input to (s) from BCD (WORD or DWORD) data type to DINT data type, and output the converted value from (d).
  - When (s) is of WORD data type



(1) Filled with 0s.

  - When (s) is of DWORD data type



(1) Filled with 0s.

- Input a WORD data type value within the range of 0H to 9999H (range of each digit: 0 to 9) or a DWORD date type value within the range of 0H to 99999999H (range of each digit: 0 to 9) to (s).
- WORD or DWORD data type can be specified for (s). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	A value other than 0 to 9 exists at any digit of the value input to (s).

- Turning on SM754 can prevent the above error from being issued. If the specified value is out of the valid range, the BCD_TO_DINT(_E) function is not executed regardless of the status (on/off) of SM754.

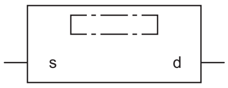
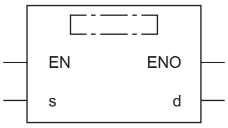
25.39 Converting BCD to STRING

BCD_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions convert a value from BCD data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=BCD_TO_STRING(s);
		[With EN/ENO]
		d:=BCD_TO_STRING_E(EN,ENO,s);

Setting data

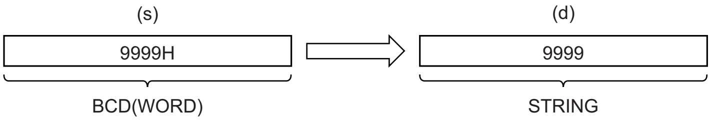
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(8)

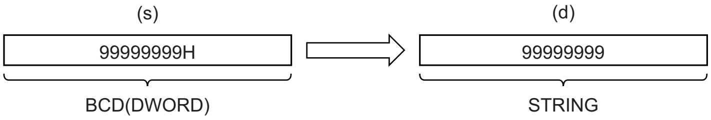
Processing details

■Operation processing

- These functions convert the value input to (s) from BCD (WORD or DWORD) data type to STRING data type, and output the converted value from (d).
  - When (s) is of WORD data type



  - When (s) is of DWORD data type



- WORD or DWORD data type can be specified for (s). BOOL data type cannot be specified.
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of WORD data type

Error code (SD0)	Description
3401H	Data input to (s) is out of the range, 0 to 9999.

- When (s) is of DWORD data type

Error code (SD0)	Description
3401H	Data input to (s) is out of the range, 0 to 99999999.

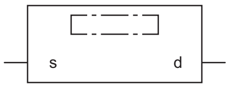
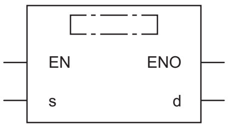
25.40 Converting REAL to INT

REAL_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from REAL data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=REAL_TO_INT(s);
		[With EN/ENO]
		d:=REAL_TO_INT_E(EN,ENO,s);

Setting data

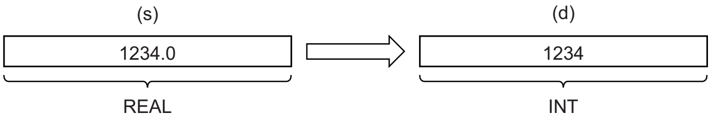
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	The single-precision real number input to (s) is out of the range, -32768 to 32767.
3402H	• 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 ±∞.

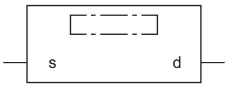
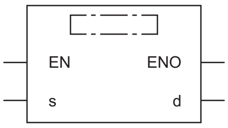
25.41 Converting REAL to DINT

REAL_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These functions convert a value from REAL data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=REAL_TO_DINT(s);
		[With EN/ENO]
		d:=REAL_TO_DINT_E(EN,ENO,s);

Setting data

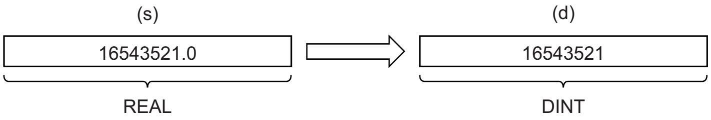
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	The single-precision real number input to (s) is out of the range, -2147483648 to 2147483647.
3402H	• 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 ±∞.

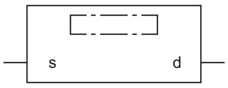
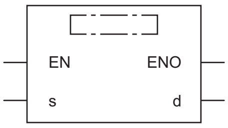
25.42 Converting REAL to LREAL

REAL_TO_LREAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from REAL data type to LREAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=REAL_TO_LREAL(s);
		[With EN/ENO]
		d:=REAL_TO_LREAL_E(EN,ENO,s);

Setting data

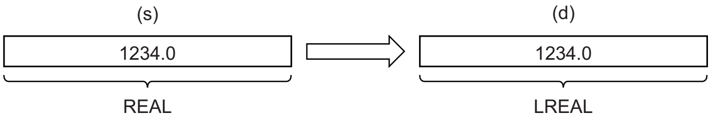
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert 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).
- 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0 or not in 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)
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

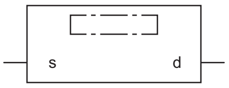
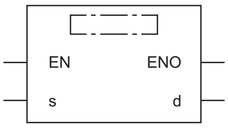
25.43 Converting REAL to STRING

REAL_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions converts a REAL data type value to STRING data type (exponential form).

Ladder, FBD/LD		Structured text

[Without EN/ENO]

s d
	[With EN/ENO]
	
	EN ENO
	s d
		[Without EN/ENO]
		d:=REAL_TO_STRING(s);
		[With EN/ENO]
		d:=REAL_TO_STRING_E(EN,ENO,s);

Setting data

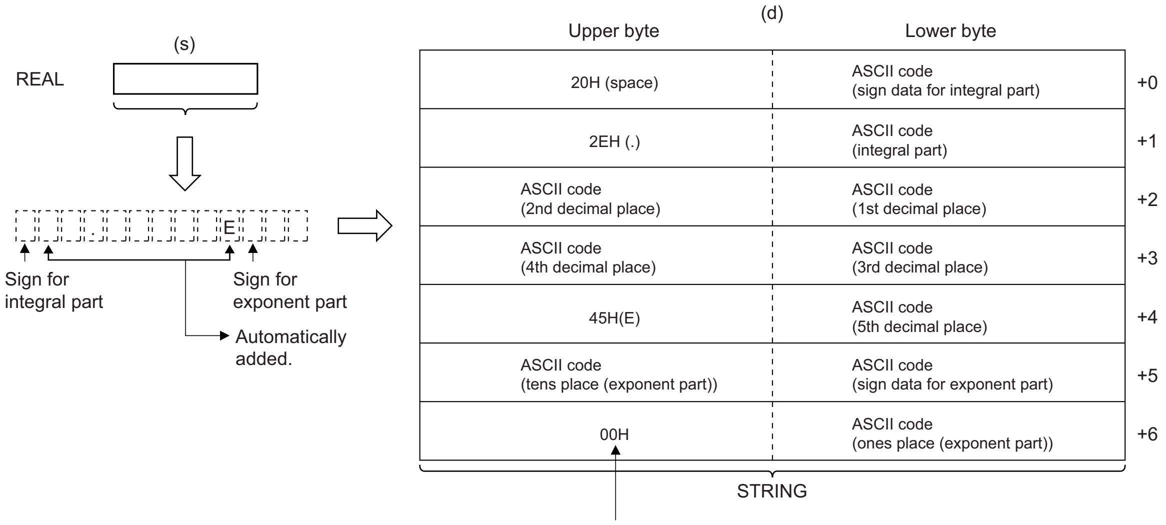
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(13)

Processing details

■Operation processing

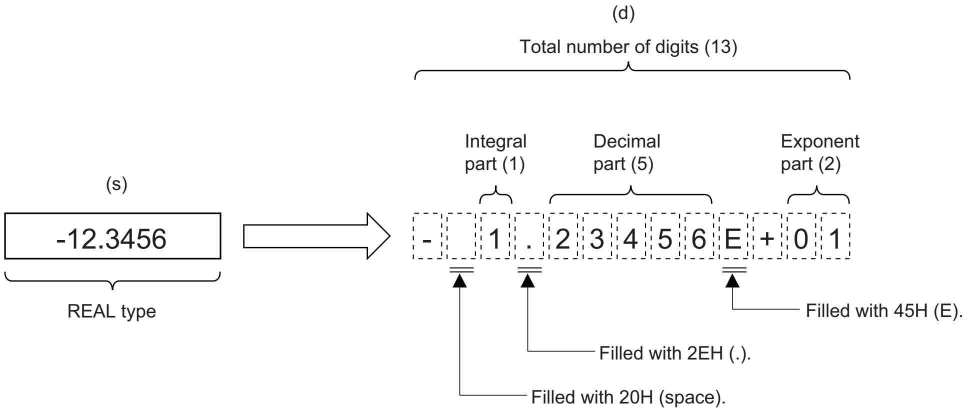
- These functions convert the value input to (s) from REAL data type to STRING data type (exponential form), and output the converted value from (d).



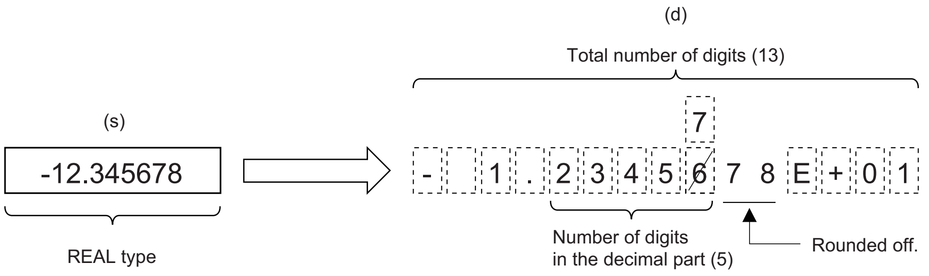
Upper byte	Lower byte	
20H (space)	ASCII code (sign data for integral part)	+0
2EH (.)	ASCII code (integral part)	+1
ASCII code (2nd decimal place)	ASCII code (1st decimal place)	+2
ASCII code (4th decimal place)	ASCII code (3rd decimal place)	+3
45H(E)	ASCII code (5th decimal place)	+4
ASCII code (tens place (exponent part))	ASCII code (sign data for exponent part)	+5
00H	ASCII code (ones place (exponent part))	+6

When SM701 is off, 00H is stored.

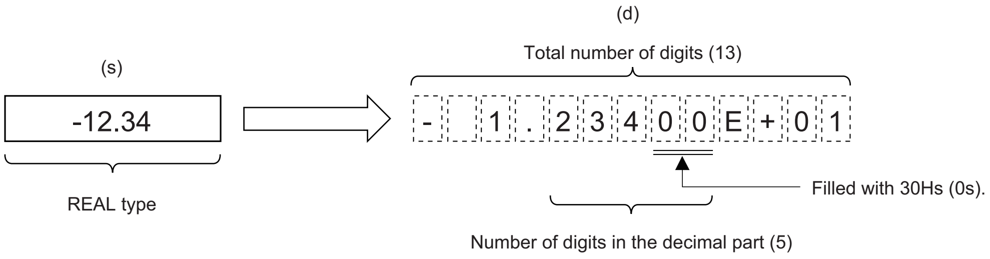
- Input a REAL data type value to (s).
- The converted string data is output from (d) as follows.
  - The number of digits for an integral part, decimal part, and exponent is fixed, integral part: one digit; decimal part: five digits; exponent: two digits.
  - As the second byte, 20H (space) is stored; as the fourth byte, 2EH (.) is stored; and as the 10th byte, 45H (E) is stored automatically.



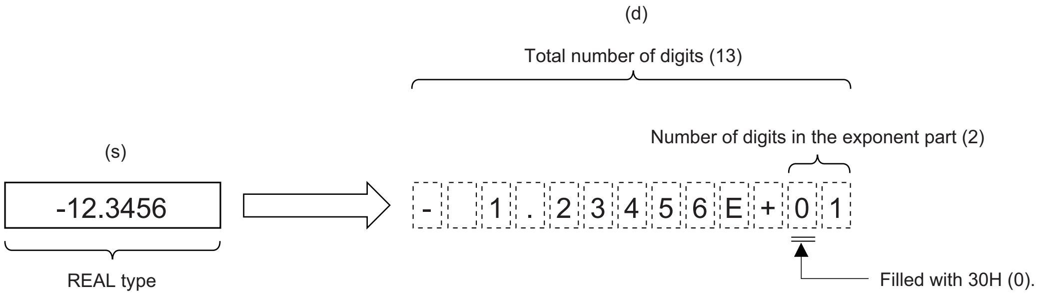
  - As sign data (for integral part), 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the input value is negative.
  - The sixth and later digits of the decimal part are rounded off.



  - If the number of digits in the input value is less than the number of significant digits, 30H (0) is stored in the decimal part.



  - As sign data (for exponent), 2BH (+) is stored if the input value is positive, and 2DH (-) is stored if the input value is negative.
  - When the exponent is one digit, 30H (0) is stored in the tens place of the exponent.



- The NULL code (00H) is automatically stored at the end (i.e. seventh word) of the converted string.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	• The value input to (s) is out of 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 input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3406H	The entire string after conversion cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

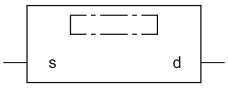
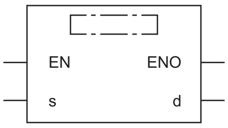
25.44 Converting LREAL to INT

LREAL_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from LREAL data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=LREAL_TO_INT(s);
		[With EN/ENO]
		d:=LREAL_TO_INT_E(EN,ENO,s);

Setting data

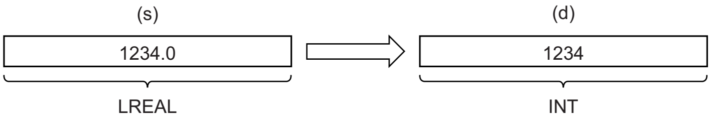
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0 or not in 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 input value is out of the range, -32768 to 32767.

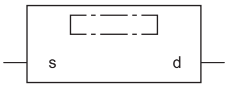
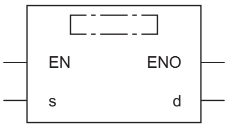
25.45 Converting LREAL to DINT

LREAL_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from LREAL data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LREAL_TO_DINT(s);
		[With EN/ENO]
		d:=LREAL_TO_DINT_E(EN,ENO,s);

Setting data

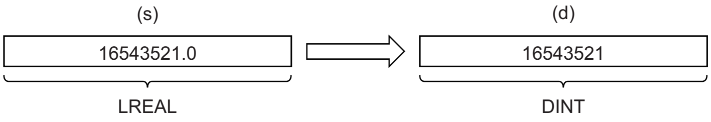
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0 or not in 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 input value is out of the range, -2147483648 to 2147483647.

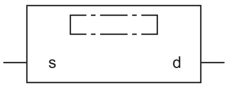
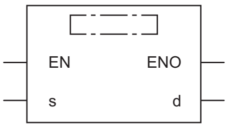
25.46 Converting LREAL to REAL

LREAL_TO_REAL(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions convert a value from LREAL data type to REAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LREAL_TO_REAL(s);
		[With EN/ENO]
		d:=LREAL_TO_REAL_E(EN,ENO,s);

Setting data

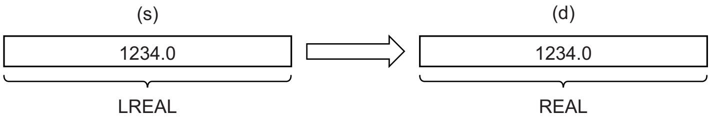
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0 or not in 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)
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

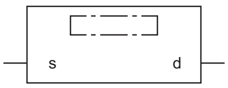
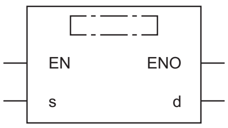
25.47 Converting TIME to BOOL

TIME_TO_BOOL(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from TIME data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_BOOL(s);
		[With EN/ENO]
		d:=TIME_TO_BOOL_E(EN,ENO,s);

Setting data

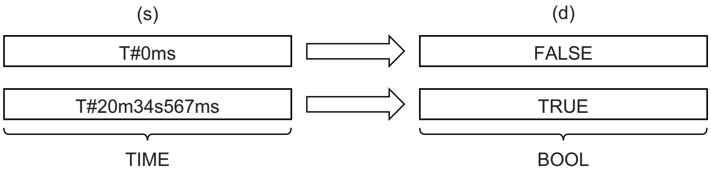
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to BOOL data type, and output the converted value from (d).



■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

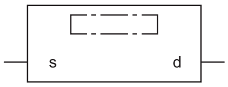
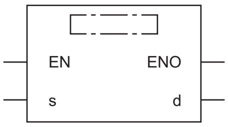
25.48 Converting TIME to WORD

TIME_TO_WORD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from TIME data type to WORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_WORD(s);
		[With EN/ENO]
		d:=TIME_TO_WORD_E(EN,ENO,s);

Setting data

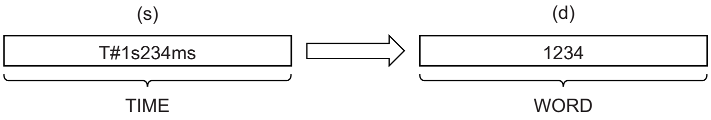
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

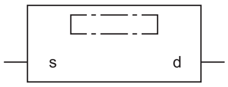
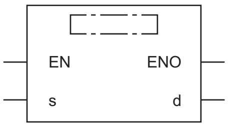
25.49 Converting TIME to DWORD

TIME_TO_DWORD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from TIME data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_DWORD(s);
		[With EN/ENO]
		d:=TIME_TO_DWORD_E(EN,ENO,s);

Setting data

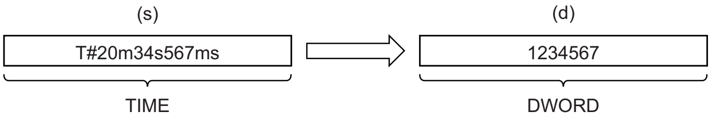
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

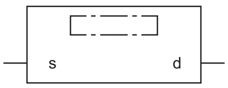
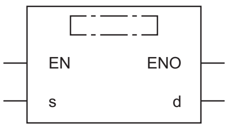
25.50 Converting TIME to INT

TIME_TO_INT(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from TIME data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=TIME_TO_INT(s);
		[With EN/ENO]
		d:=TIME_TO_INT_E(EN,ENO,s);

Setting data

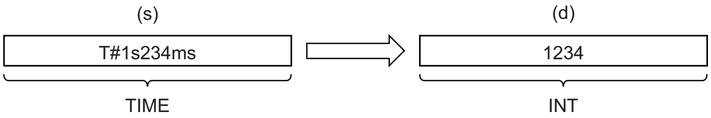
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

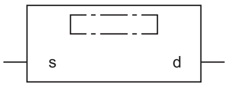
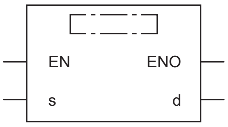
25.51 Converting TIME to DINT

TIME_TO_DINT(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions convert a value from TIME data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_DINT(s);
		[With EN/ENO]
		d:=TIME_TO_DINT_E(EN,ENO,s);

Setting data

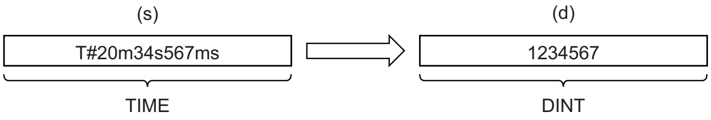
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

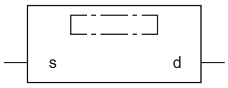
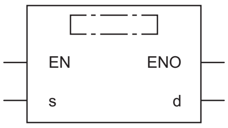
25.52 Converting TIME to STRING

TIME_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from TIME data type to STRING data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_STRING(s);
		[With EN/ENO]
		d:=TIME_TO_STRING_E(EN,ENO,s);

Setting data

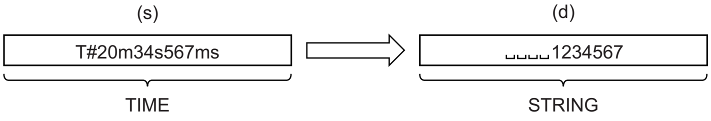
■Description, type, data type

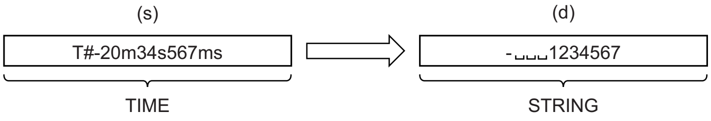
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING STRING(11)

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to STRING data type, and output the converted value from (d).





- Input a TIME data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.
- The operation result will be as follows.
  - As the first character, 20H (space) is stored if the output value is positive, and 2DH (-) is stored if the output value is negative.
  - At the left of the number of significant digits, 20H (space) is stored.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

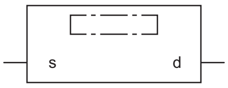
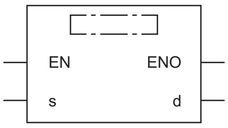
25.53 Converting STRING to BOOL

STRING_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These functions convert a value from STRING data type to BOOL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_BOOL(s);
		[With EN/ENO]
		d:=STRING_TO_BOOL_E(EN,ENO,s);

Setting data

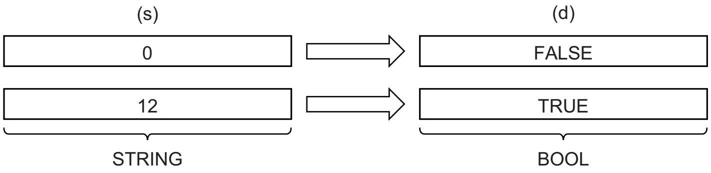
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(1)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type (decimal form/exponential form) to BOOL data type, and output the converted value from (d).



■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

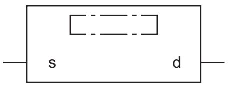
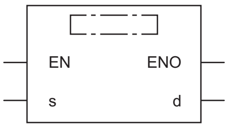
There is no operation error.

STRING_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to WORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=STRING_TO_WORD(s);
		[With EN/ENO]
		d:=STRING_TO_WORD_E(EN,ENO,s);

Setting data

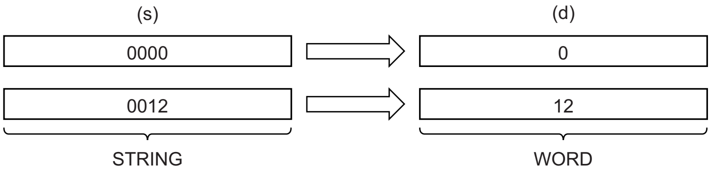
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(4)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to WORD data type, and output the converted value from (d).



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

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An ASCII code other than 30H to 39H and 41H to 46H is input.

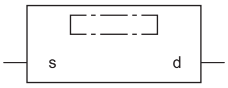
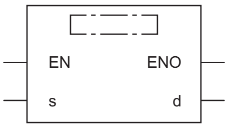
25.55 Converting STRING to DWORD

STRING_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from STRING data type to DWORD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_DWORD(s);
		[With EN/ENO]
		d:=STRING_TO_DWORD_E(EN,ENO,s);

Setting data

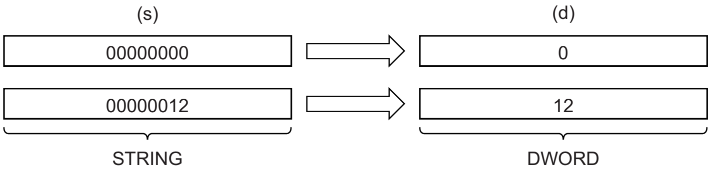
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(8)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to DWORD data type, and output the converted value from (d).



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

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An ASCII code other than 30H to 39H and 41H to 46H is input.

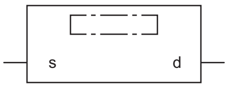
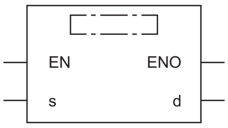
25.56 Converting STRING to INT

STRING_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to INT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_INT(s);
		[With EN/ENO]
		d:=STRING_TO_INT_E(EN,ENO,s);

Setting data

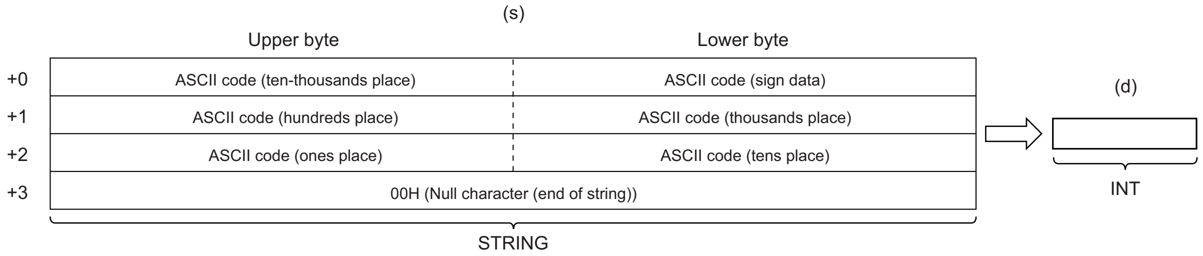
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(6)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to INT data type, and output the converted value from (d).



- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 20H, 2DH, and 00H
  - STRING data type value: -32768 to 32767

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An out-of-range value is input to (s). • ASCII code: other than 30H to 39H, 20H, and 00H • STRING data type value: other than -32768 to 32767

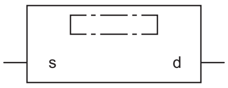
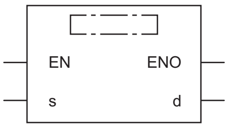
25.57 Converting STRING to DINT

STRING_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from STRING data type to DINT data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=STRING_TO_DINT(s);
		[With EN/ENO]
		d:=STRING_TO_DINT_E(EN,ENO,s);

Setting data

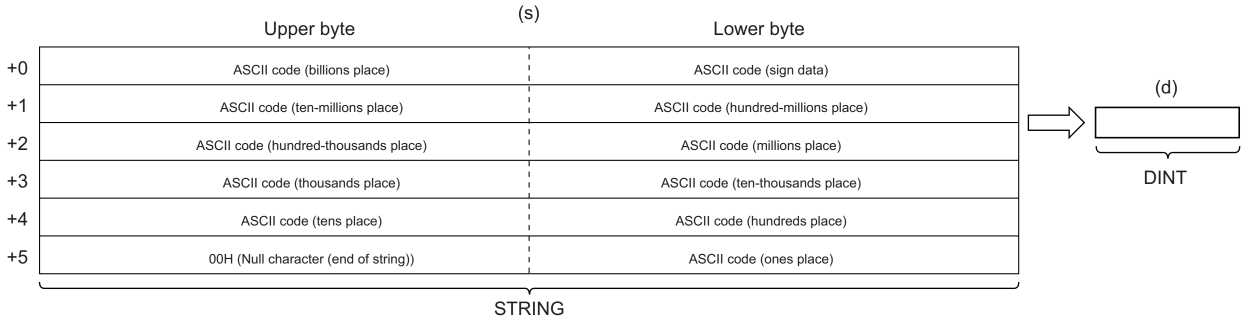
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(11)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to DINT data type, and output the converted value from (d).



	Upper byte	Lower byte
+0	ASCII code (billions place)	ASCII code (sign data)
+1	ASCII code (ten-millions place)	ASCII code (hundred-millions place)
+2	ASCII code (hundred-thousands place)	ASCII code (millions place)
+3	ASCII code (thousands place)	ASCII code (ten-thousands place)
+4	ASCII code (tens place)	ASCII code (hundreds place)
+5	00H (Null character (end of string))	ASCII code (ones place)

- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 20H, 2DH, and 00H
  - STRING data type value: -2147483648 to 2147483647

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An out-of-range value is input to (s). • ASCII code: other than 30H to 39H, 20H, and 00H • STRING data type value: other than -2147483648 to 2147483647

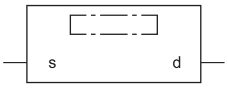
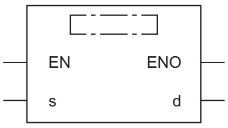
25.58 Converting STRING to BCD

STRING_TO_BCD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions convert a value from STRING data type to BCD data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_BCD(s);
		[With EN/ENO]
		d:=STRING_TO_BCD_E(EN,ENO,s);

Setting data

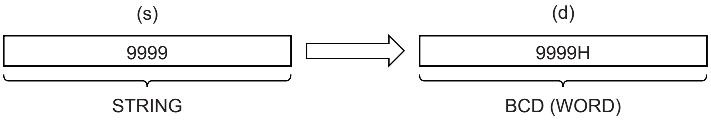
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(8)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to BCD (WORD) data type, and output the converted value from (d).



- The ASCII code 20H (space) that exists in the the string is ignored.
- The ASCII codes 20H (space) and 30H (0) that exist in the string are counted as one character as well.
- Input a STRING data type value to (s) within the following range.

ASCII code: 30H to 39H, 20H, 00H

- If the string input has less than four characters, the string will be supplemented with zero(s). (Zero(s) is/are added at the end of the string.) For this reason, if a string shorter than four characters is to be converted, input a character string padded with 0s (e.g. '0001' for '1').
- If the string length exceeds 4 characters, the four left characters are regarded as the targets to convert.

Input string	Conversion target string	Output (BCD data type)
'1'	'1000'	1000H (4096D)
'12'	'1200'	1200H(4608D)
'123'	'1230'	1230H(4656D)
'1234'	'1234'	1234H(4660D)
'12345'	'1234'	1234H(4660D)

- WORD or DWORD data type can be specified for (d). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An ASCII code other than 30H to 39H, 20H, or 00H is input.

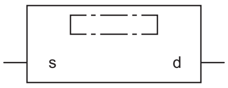
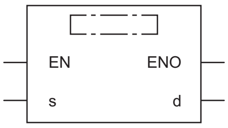
25.59 Converting STRING to REAL

STRING_TO_REAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from STRING data type to REAL data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_REAL(s);
		[With EN/ENO]
		d:=STRING_TO_REAL_E(EN,ENO,s);

Setting data

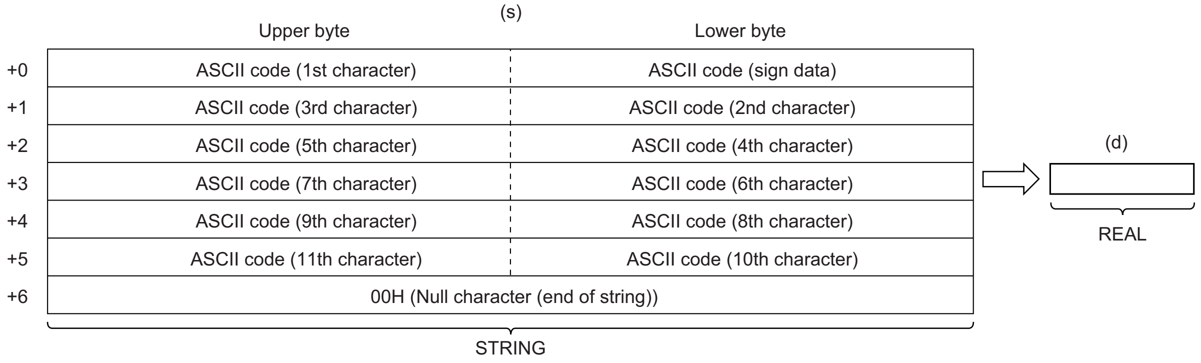
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(24)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type (decimal form/exponential form) to REAL data type, and output the converted value from (d).

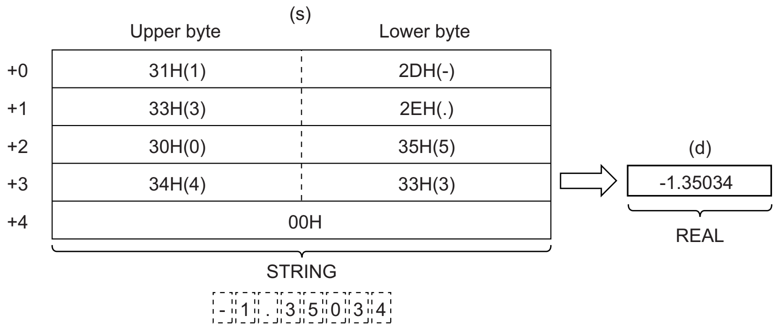


(s)

	Upper byte	Lower byte
+0	ASCII code (1st character)	ASCII code (sign data)
+1	ASCII code (3rd character)	ASCII code (2nd character)
+2	ASCII code (5th character)	ASCII code (4th character)
+3	ASCII code (7th character)	ASCII code (6th character)
+4	ASCII code (9th character)	ASCII code (8th character)
+5	ASCII code (11th character)	ASCII code (10th character)
+6	00H (Null character (end of string))	

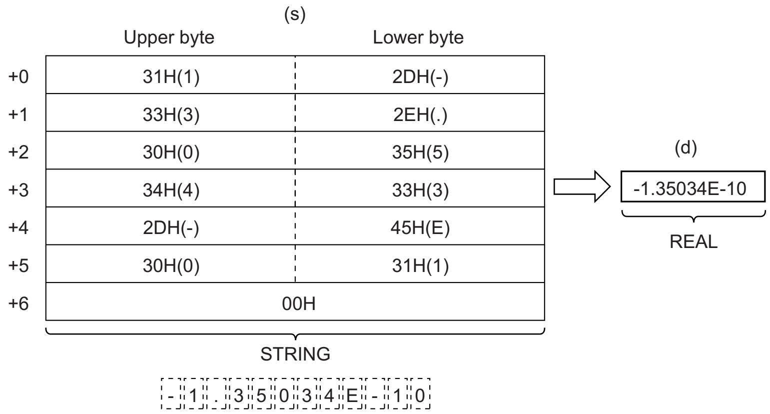
STRING

- The STRING data type value both in decimal form and exponential form can be converted.
  - When (s) is in decimal form



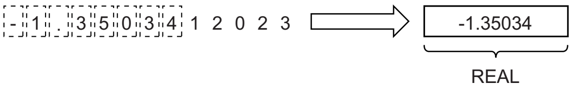
	(s) Upper byte	(s) Lower byte
+0	31H(1)	2DH(-)
+1	33H(3)	2EH(.)
+2	30H(0)	35H(5)
+3	34H(4)	33H(3)
+4	00H	

  - When (s) is in exponential form

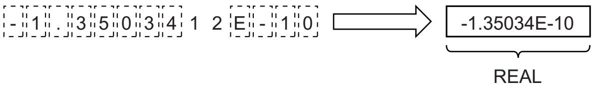


	(s) Upper byte	(s) Lower byte
+0	31H(1)	2DH(-)
+1	33H(3)	2EH(.)
+2	30H(0)	35H(5)
+3	34H(4)	33H(3)
+4	2DH(-)	45H(E)
+5	30H(0)	31H(1)
+6	00H	

- The number of significant digits of the STRING data type value is six. (The sign, decimal point, and exponent are not included.) The seventh and later digits are rounded down when the data is converted.
  - When (s) is in decimal form

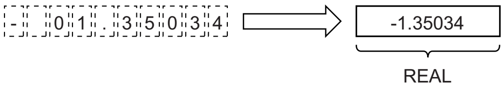


  - When (s) is in exponential form

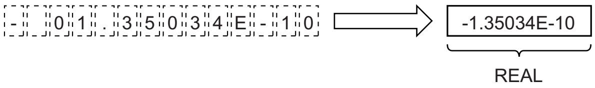


- In decimal form, when 2BH (+) is specified as sign data or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.
- In exponential form, when 2BH (+) is specified as sign data for the exponent or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.

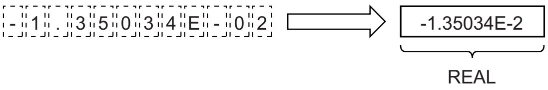
- The ASCII code 20H (space) or 30H (0) that exists before the first numerical value 0 in the STRING data type value is ignored.
  - When (s) is in decimal form



  - When (s) is in exponential form



- The ASCII code 30H (0) that exists between E and a numerical value in the STRING data type value is ignored (in exponential form only).



- The ASCII code 20H (space) that exists in the the string is ignored.
- Up to 24 characters can be input. The ASCII codes 20H (space) and 30H (0) that exist in the string are counted as one character as well.
- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 45H, 2BH, 2DH, 2EH, 20H, and 00H

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the device area (between the specified device number and the last device number) specified by (s).
3401H	An out-of-range value is input to (s). • The integral part or decimal part contains a character other than 30H(0) to 39H(9). • More than one 2EH (.) exists in the specified string. • The exponent of the specified character string contains a character other than 45H(E), 65H(e), 2BH(+), and 2DH(-). • The specified character string contains more than one exponent 45H(E) or 65H(e). • The exponent in the specified string contains a numerical value consisting of three digits or more. • The exponent of the specified character string contains more than one sign 2BH(+) or 2DH(-). • The specified string (in the integral part if the decimal format is used or in the mantissa if the exponent format is used) contains more than one sign data of 2BH(+) or 2DH(-). • The number of characters in the device specified by (s) and later is 0 or exceeds 24.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)|<2^{128}$

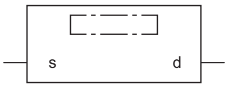
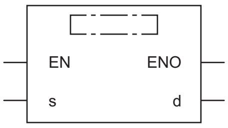
25.60 Converting STRING to TIME

STRING_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from STRING data type to TIME data type.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=STRING_TO_TIME(s);
		[With EN/ENO]
		d:=STRING_TO_TIME_E(EN,ENO,s);

Setting data

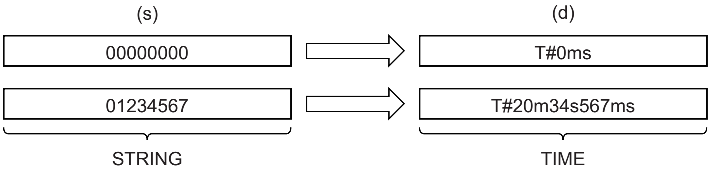
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(11)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to TIME data type, and output the converted value from (d).



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

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3401H	An ASCII code other than 30H to 39H, 20H, and 00H is input.
	The STRING data type value input is out of the following range: -2147483648 to 4147483647

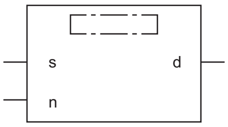
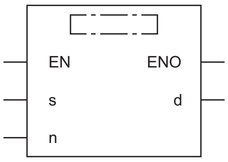
25.61 Converting Bit Array to INT

BITARR_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert the specified number of bits in a bit array to an INT data type value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BITARR_TO_INT(s,n);
		[With EN/ENO]
		d:=BITARR_TO_INT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (BitArr)	Input (An element can be specified by a variable.)	Input variable	Boolean array element
n	Number of bits (4, 8, 12, or 16)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	ANY16

Processing details

■Operation processing

- These functions convert the number of bits specified by (n) starting from the bit array element input to (s) to any 16-bit data type value, and output the converted value from (d).
- Zeros (0s) are set for all the bits exceeding the specified number of bits.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

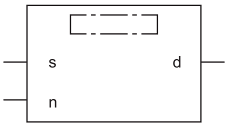
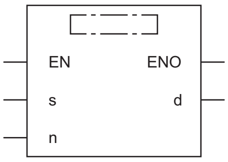
There is no operation error.

25.62 Converting Bit Array to DINT

BITARR_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert the specified number of bits in a bit array to a DINT data type value.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] d:=BITARR_TO_DINT(s,n) [With EN/ENO] d:=BITARR_TO_DINT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (BitArr)	Input (An element can be specified by a variable.)	Input variable	Boolean array element
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	ANY32

Processing details

■Operation processing

- These functions convert the number of bits specified by (n) starting from the bit array element input to (s) to any 32-bit data type value, and output the converted value from (d).
- Zeros (0s) are set for all the bits exceeding the specified number of bits.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

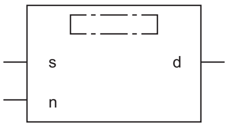
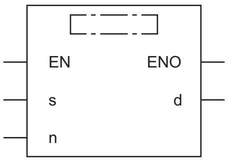
25.63 Converting INT to Bit Array

INT_TO_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the lower n bits of the INT data type value to the bit array.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=INT_TO_BITARR(s,n);
		[With EN/ENO]
		d:=INT_TO_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	ANY16
n	Number of bits (4, 8, 12, or 16)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output (An element can be specified by a variable.)	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the lower (n) bits of ANY16 type data specified by (s) to (d).
- The output bits beyond the specified number of bits are not changed.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

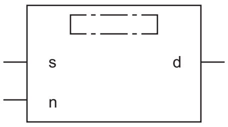
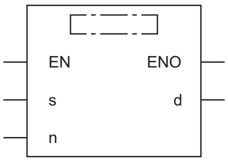
25.64 Converting DINT to Bit Array

DINT_TO_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the lower n bits of the DINT data type value to the bit array.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_BITARR(s,n);
		[With EN/ENO]
		d:=DINT_TO_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	ANY32
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output (An element can be specified by a variable.)	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the lower (n) bits of any 32-bit data type value specified by (s) to (d).
- The output bits beyond the specified number of bits are not changed.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

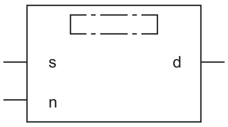
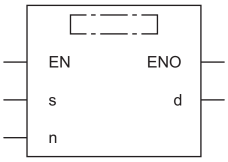
25.65 Copying the Bit Array

CPY_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions copy the bit array by the specified number of bits.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=CPY_BITARR(s,n);
		[With EN/ENO]
		d:=CPY_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s(BitArrIn)	Input	Input variable	Boolean array element
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the bit array (number of (n) bits) specified by (s) to (d).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

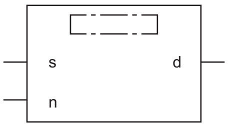
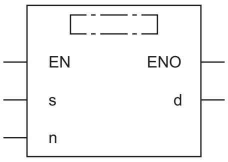
25.66 Reading the Specified Bit of the Word Label

GET_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions read a value from the specified bit of a word label.

Ladder[1]		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=GET_BIT_OF_INT(s,n);
		[With EN/ENO]
		d:=GET_BIT_OF_INT_E(EN,ENO,s,n);

[1] FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	INT
n	Number of bits (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions output a value in the (n)th bit of (s).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as GET_BIT_OF_INT can be created.

The following example reads the value in bit 5 (b5) of g_int1 into g_bool1 the same as when GET_BIT_OF_INT is used.

Ladder

ST

```
g_bool1 := g_int1.5;
```

FBD/LD

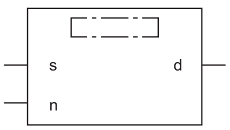
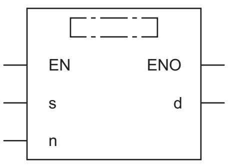
25.67 Writing the Specified Bit of the Word Label

SET_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions write a value to the specified bit of a word label.

Ladder[1]		**Structured text**

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=SET_BIT_OF_INT(s,n);
		[With EN/ENO]
		d:=SET_BIT_OF_INT_E(EN,ENO,s,n);

[1] FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	BOOL
n	Number of bits (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Input/output	Input/output variable	INT

Processing details

■Operation processing

- These functions write a BOOL data type value specified by (s) to the (n)th bit of (d).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Precautions

When using SET_BIT_OF_INT(_E) in ST, create a program which assigns the return value to a variable.

```
//The return value of SET_BIT_OF_INT is assigned to a variable and used.
g_int1 := SET_BIT_OF_INT(TRUE, 0);
g_bool1 := GET_BIT_OF_INT(g_int1, 0);
```

The return value of SET_BIT_OF_INT(_E) also works as input/output. Do not directly assign it to the input argument of another instruction, function, or function block.

```
//In the following program, the value of the first argument of GET_BIT_OF_INT is undefined.
g_bool1 := GET_BIT_OF_INT( SET_BIT_OF_INT(TRUE, 0), 0);
```

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as SET_BIT_OF_INT can be created.

The following example changes the value in bit 5 (b5) of g_int1 to the value of g_bool1 the same as when SET_BIT_OF_INT is used.

Ladder

ST

```
g_int1.5 := g_bool1;
```

FBD/LD

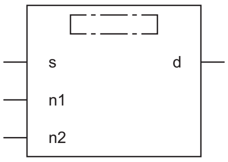
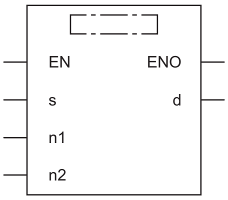
25.68 Copying the Specified Bit of the Word Label

CPY_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions copy the specified bit of the word label to the specified bit of another word label.

Ladder[1]		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=CPY_BIT_OF_INT(s,n1,n2);
		[With EN/ENO]
		d:=CPY_BIT_OF_INT_E(EN,ENO,s,n1,n2);

[1] FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	INT
n1	Number of bits in input variable (0 to 15)	Input variable	INT
n2	Number of bits in output variable (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Input/output	Input/output variable	INT

Processing details

■Operation processing

- These functions copy the value in the (n1)th bit of the word specified by (s) to the (n2)th bit of (d).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Precautions

When using CPY_BIT_OF_INT(_E) in ST, create a program which assigns the return value to a variable.

```
//The return value of CPY_BIT_OF_INT is assigned to a variable and used.
g_int2 := CPY_BIT_OF_INT(g_int1,5,3);
g_bool1 := GET_BIT_OF_INT(g_int2,3);
```

The return value of CPY_BIT_OF_INT(_E) also works as input/output. Do not directly assign it to the input argument of another instruction, function, or function block.

```
//In the following program, the value of the first argument of GET_BIT_OF_INT is undefined.
g_bool1 := GET_BIT_OF_INT( CPY_BIT_OF_INT(g_int1,5,3), 3);
```

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as CPY_BIT_OF_INT can be created.

The following example changes the value in bit 3 (b3) of g_int2 to the value of bit 5 (b5) of g_int1 the same as when CPY_BIT_OF_INT is used.

Ladder

ST

```
g_int2.3 := g_int1.5;
```

FBD/LD

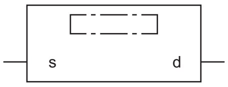
25.69 Getting the Start Data

GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the start data of the specified data as BOOL, INT, or WORD type data.

Ladder, FBD/LD	Structured text

	d:=GET_BOOL_ADDR(s)
	d:=GET_INT_ADDR(s);
	d:=GET_WORD_ADDR(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s	Input	Input variable	ANY
d	Output	Output variable	BOOL/INT/WORD

Processing details

■Operation processing

- The GET_BOOL_ADDR function outputs the start data of the data specified by (s) as BOOL type data.
- The GET_INT_ADDR function outputs the start data of the data specified by (s) as INT type data.
- The GET_WORD_ADDR function outputs the start data of the data specified by (s) as WORD type data.

Standard function	Input data type	Output data type
GET_BOOL_ADDR	BOOL ARRAY OF BOOL	BOOL
GET_INT_ADDR	INT DINT WORD REAL TIME STRING ARRAY OF INT ARRAY OF DINT ARRAY OF WORD ARRAY OF DWORD ARRAY OF REAL ARRAY OF TIME	INT
GET_WORD_ADDR		WORD

■Operation result

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

Operation error

There is no operation error.

26 SINGLE VARIABLE FUNCTIONS

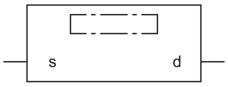
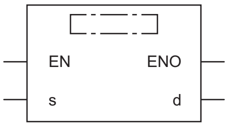
26.1 Calculating the Absolute Value

ABS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the absolute value of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ABS(s);
		[With EN/ENO]
		d:=ABS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

Processing details

■Operation processing

- These functions output 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. When ABS_E is used, ENO outputs TRUE.)

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of REAL data type

Error code (SD0)	Description
3402H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

- When (s) is of LREAL data type

Error code (SD0)	Description
3402H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

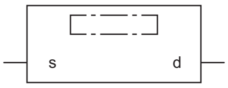
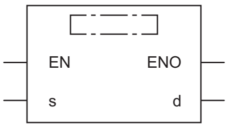
26.2 Calculating the Square Root

SQRT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions calculate the square root of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SQRT(s);
		[With EN/ENO]
		d:=SQRT_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3405H	The input value is negative.

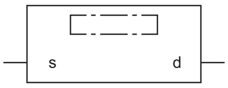
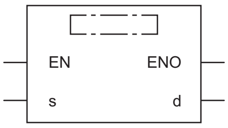
26.3 Calculating the Natural Logarithm

LN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions output the natural logarithm (logarithm with base e) of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=LN(s);
		[With EN/ENO]
		d:=LN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3405H	The input value is negative.

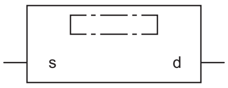
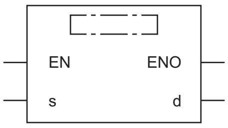
26.4 Calculating the Common Logarithm

LOG(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the common logarithm (logarithm with base 10) of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LOG(s);
		[With EN/ENO]
		d:=LOG_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of REAL data type

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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 (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

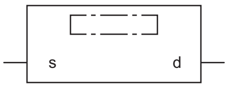
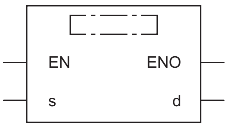
26.5 Calculating the Exponent

EXP(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the exponent of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=EXP(s);
		[With EN/ENO]
		d:=EXP_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

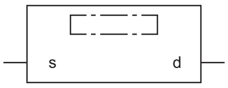
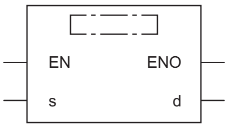
26.6 Calculating the Sine

SIN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the sine of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SIN(s);
		[With EN/ENO]
		d:=SIN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.

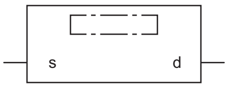
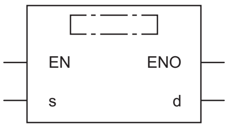
26.7 Calculating the Cosine

COS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the cosine of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=COS(s);
		[With EN/ENO]
		d:=COS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.

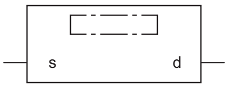
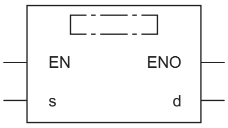
26.8 Calculating the Tangent

TAN(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions output the tangent of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=TAN(s);
		[With EN/ENO]
		d:=TAN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.

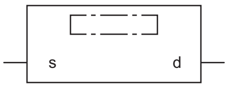
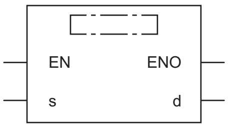
26.9 Calculating the Arc Sine

ASIN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the arc sine (SIN^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ASIN(s);
		[With EN/ENO]
		d:=ASIN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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:

$B=\text{SIN}^{-1} A$

- Input a REAL data type value to (s) within the following range.

ASIN(_E): -1.0 to 1.0

- The value (angle) is output from (d) in radians ($\text{angle}\times\pi/180$).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.
3405H	The value input with ASIN(_E) is other than -1.0 to 1.0.

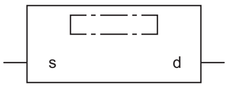
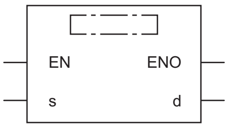
26.10 Calculating the Arc Cosine

ACOS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the arc cosine (COS^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ACOS(s);
		[With EN/ENO]
		d:=ACOS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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(_E): -1.0 to 1.0

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.
3405H	The value input with ACOS(_E) is other than -1.0 to 1.0.

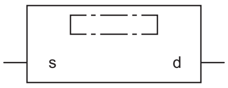
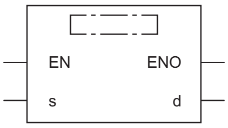
26.11 Calculating the Arc Tangent

ATAN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the arc tangent (TAN^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ATAN(s);
		[With EN/ENO]
		d:=ATAN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
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(_E): $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The input value is -0.

27 ARITHMETIC OPERATION FUNCTIONS

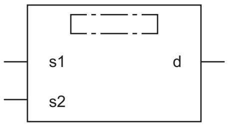
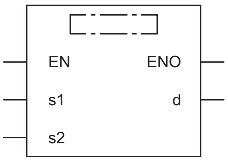
27.1 Addition

ADD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the sum of input values ((s1)+(s2)+···+(s28)).

Ladder, FBD/LD*1		Structured text*1

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ADD(s1,s2);
		[With EN/ENO]
		d:=ADD_E(EN,ENO,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

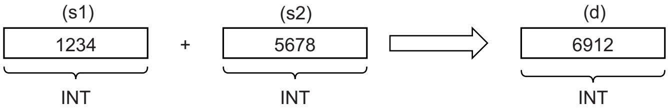
Processing details

■Operation processing

- These functions add the INT, DINT, 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, 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: INT	Data type: DINT	Data type: REAL/LREAL
• Even if an underflow or overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [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.	• Even if an underflow or overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [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.	An operation error occurs and an undefined value is output.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) to (s28) are of REAL data type

Error code (SD0)	Description
3402H	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 ±∞.
3403H	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 (SD0)	Description
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

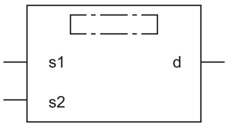
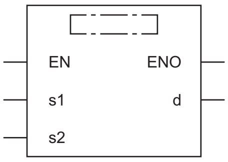
27.2 Multiplication

MUL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the product of input values ((s1)×(s2)×···×(s28)).

Ladder, FBD/LD[1]		Structured text[1]

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=MUL(s1,s2);
		[With EN/ENO]
		d:=MUL_E(EN,ENO,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

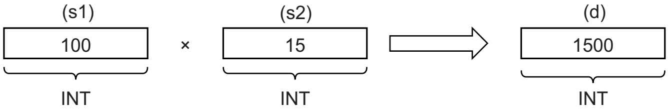
Processing details

■Operation processing

- These functions multiply the INT, DINT, 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, 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: INT	Data type: DINT	Data type: REAL/LREAL
• Even if an underflow or overflow occurs, no operation error is issued. When MUL_E is used, ENO outputs TRUE. • Even if the operation result is outside the INT data type range, the INT data type value is output; (In this case, the output value is of INT data type with the upper 16 bits deleted although the operation result is a DINT data type value.) • If the operation result is outside the INT data type range, convert the input value to the DINT data type by using the INT_TO_DINT function, and then perform operation.	• Even if an underflow or overflow occurs, no operation error is issued. When MUL_E is used, ENO outputs TRUE. • Even if the operation result is outside the DINT data type range, the DINT data type value is output; (In this case, the output value is of DINT data type with the upper 32 bits deleted although the operation result is 64-bit data.) • If the operation result is outside the DINT data type range, convert the input value to the REAL data type by using the DINT_TO_REAL function, and then perform operation.	An operation error occurs and an undefined value is output.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	**Operation result**	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

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 (SD0)	**Description**
3402H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	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 (SD0)	**Description**
3402H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

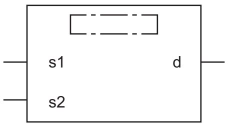
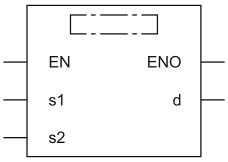
27.3 Subtraction

SUB(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the difference between input values ((s1)-(s2)).

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SUB(s1,s2);
		[With EN/ENO]
		d:=SUB_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

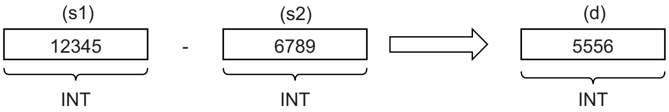
Processing details

■Operation processing

- These functions perform subtraction between the INT, DINT, 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, 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: INT	Data type: DINT	Data type: REAL/LREAL
• Even if an underflow or overflow occurs, no operation error is issued. When SUB_E is used, ENO outputs TRUE. [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.	• Even if an underflow or overflow occurs, no operation error is issued. When SUB_E is used, ENO outputs TRUE. [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.	An operation error occurs and an undefined value is output.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) and (s2) are of REAL data type

Error code (SD0)	Description
3402H	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 ±∞.
3403H	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 (SD0)	Description
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

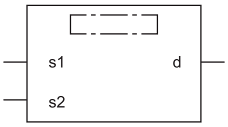
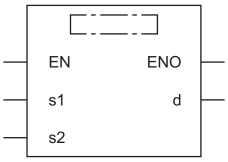
27.4 Division

DIV(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the quotient of input values ((s1)÷(s2)).

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DIV(s1,s2);
		[With EN/ENO]
		d:=DIV_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Dividend	Input variable	ANY_NUM
s2 (IN2)	Divisor	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

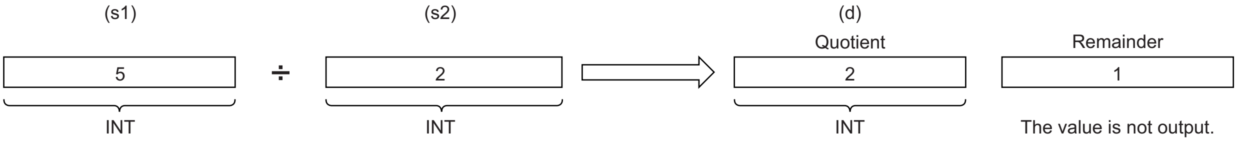
Processing details

■Operation processing

- These functions perform division between the INT, DINT, 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, REAL, or LREAL data type value to (s1) and (s2). provided that the value input to (s2) shall be other than 0.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) and (s2) are of INT data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of REAL data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.
3402H	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 ±∞.
3403H	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

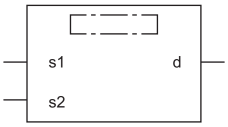
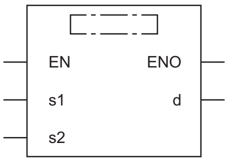
Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $|(d)|<2^{1024}$

27.5 Remainder

MOD(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the remainder of input values ((s1)÷(s2)).

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design)) [With EN/ENO] d:=MOD_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Dividend	Input variable	ANY_INT
s2 (IN2)	Divisor	Input variable	ANY_INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_INT

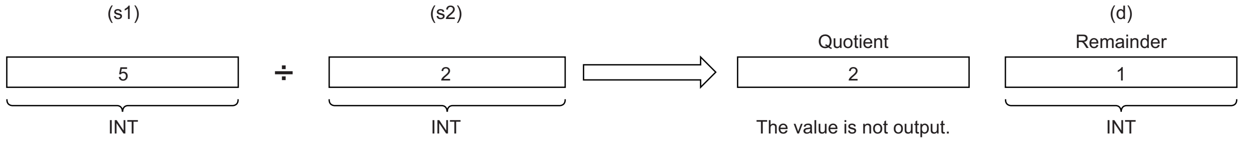
Processing details

■Operation processing

- These functions perform division between the INT or DINT 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 or DINT data type value to (s1) and (s2), provided that the value input to (s2) shall be other than 0.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) and (s2) are of INT data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

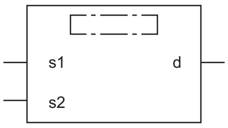
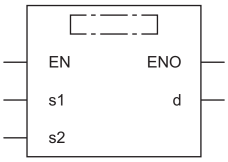
27.6 Exponentiation

EXPT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These functions output the exponentiation of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=EXPT(s1,s2);
		[With EN/ENO]
		d:=EXPT_E(EN,ENO,s1,s2);

Setting data

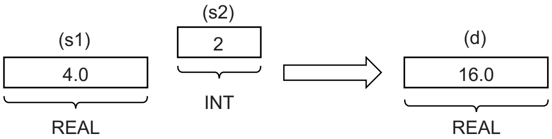
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Base	Input variable	ANY_REAL
s2 (IN2)	Exponent (power)	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions exponentiate the REAL or LREAL data type value input to (s1) with the exponent (INT, DINT, REAL, or LREAL data type) input to (s2), and output the operation result from (d).



■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

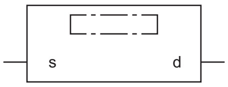
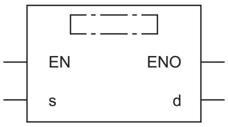
27.7 Assignment (Move Operation)

MOVE(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the assignment value of an input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=MOVE(s);
		[With EN/ENO]
		d:=MOVE_E(EN,ENO,s);

Setting data

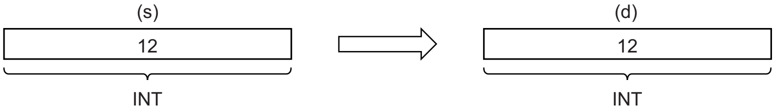
■Description, type, data type

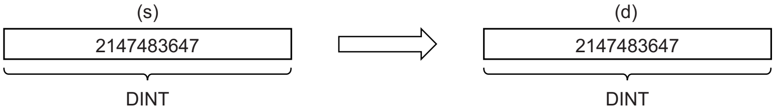
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

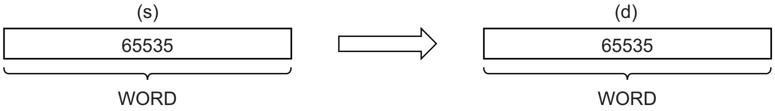
Processing details

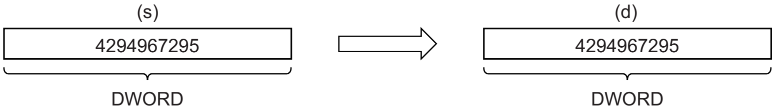
■Operation processing

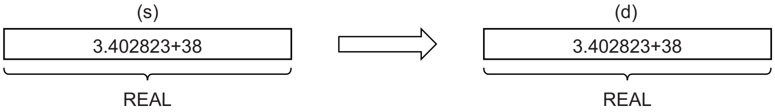
- These functions assign the value of the input variable specified by (s) to the output variable specified by (d).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s) and (d). The values input to (s) and (d) must be of the same data type.

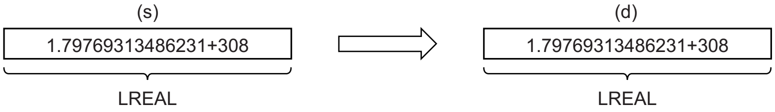












■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) in the device/label memory.
3405H	The number of characters in the string input to (s) exceeds 16383.
3406H	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.)

28 BIT SHIFT FUNCTIONS

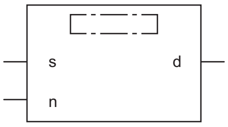
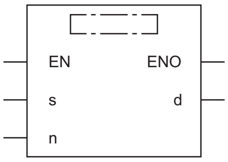
28.1 Shifting Data to the Left by n Bit(s)

SHL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions shift the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=SHL(s,n);
		[With EN/ENO]
		d:=SHL_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

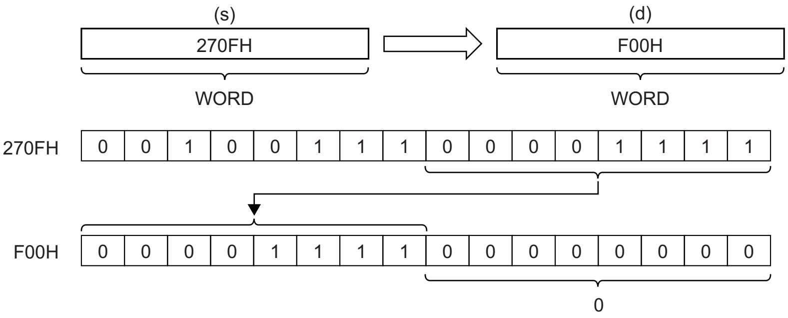
Processing details

■Operation processing

- These functions shift the WORD or DWORD data type value input to (s) to the left by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be shifted in (n).

Ex.

Data type of (s): WORD, Value input to (n): 8



- The (n) bit(s) from the least significant bit is/are filled with 0(s).
- Input a WORD or DWORD data type value to (s).
- Input an INT data type value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): WORD	Data type of (s): DWORD
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

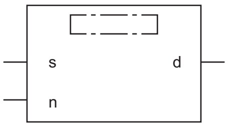
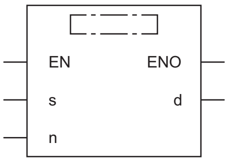
28.2 Shifting Data to the Right by n Bit(s)

SHR(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions shift the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SHR(s,n);
		[With EN/ENO]
		d:=SHR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

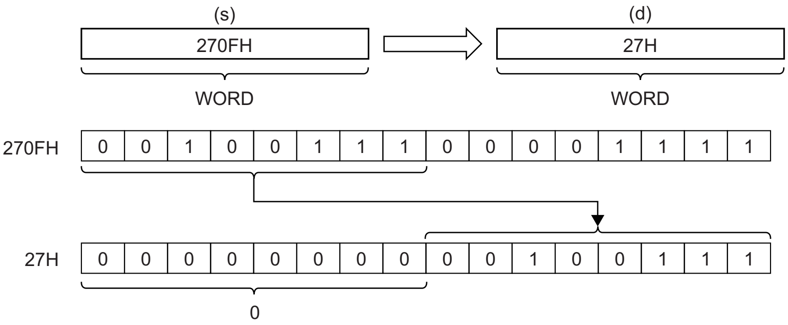
Processing details

■Operation processing

- These functions shift the WORD or DWORD data type value input to (s) to the right by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be shifted in (n).

Ex.

Data type of (s): WORD, Value input to (n): 8



- The (n) bit(s) from the most significant bit is/are filled with 0(s).
- Input a WORD or DWORD data type value to (s).
- Input an INT data type value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): WORD	Data type of (s): DWORD
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

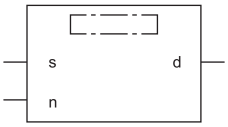
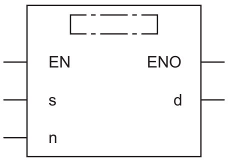
28.3 Rotating Data to the Left by n Bit(s)

ROL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions rotate the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ROL(s,n);
		[With EN/ENO]
		d:=ROL_E(EN,ENO,s,n);

Setting data

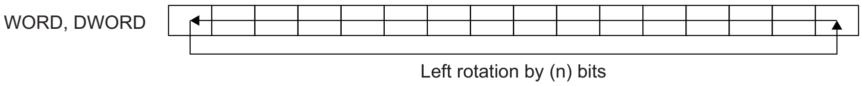
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

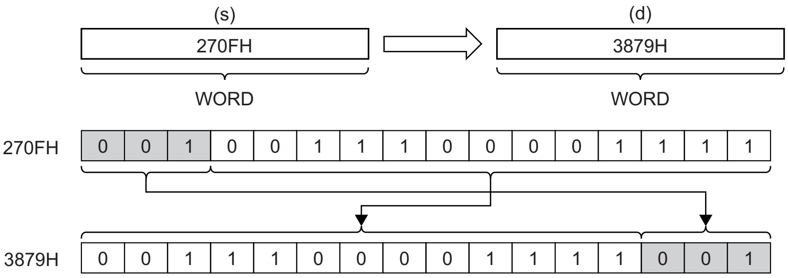
■Operation processing

- These functions rotate the WORD or DWORD data type value input to (s) to the left by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be rotated in (n).



Ex.

Data type of (s): WORD, Value input to (n): 3 (The data rotates to the left by 3 bits.)



- Input a WORD or DWORD data type value to (s).
- Input an INT data type value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): WORD	Data type of (s): DWORD
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

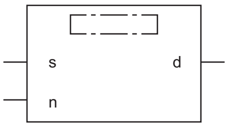
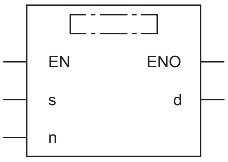
28.4 Rotating Data to the Right by n Bit(s)

ROR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions rotate the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ROR(s,n);
		[With EN/ENO]
		d:=ROR_E(EN,ENO,s,n);

Setting data

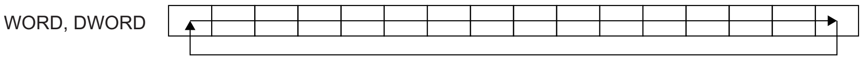
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

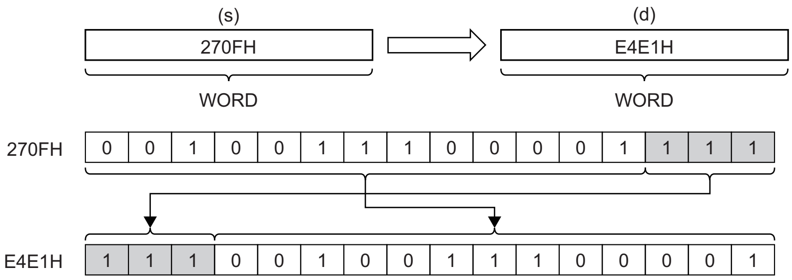
- These functions rotate the WORD or DWORD data type value input to (s) to the right by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be rotated in (n).



Right rotation by (n) bits

Ex.

Data type of (s): WORD, Value input to (n): 3 (The data rotates to the right by 3 bits.)



- Input a WORD or DWORD data type value to (s).
- Input an INT data type value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): WORD	Data type of (s): DWORD
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

29 BOOLEAN FUNCTIONS

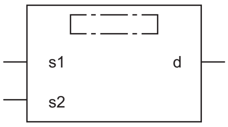
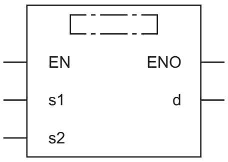
29.1 AND Operation, OR Operation, and XOR Operation

AND(_E), OR(_E), XOR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- AND(_E): These functions output the logical product of input values.
- OR(_E): These functions output the logical sum of input values.
- OR(_E): These functions output the exclusive logical sum of input values.

Ladder, FBD/LD[1]		**Structured text**[1]

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design))
		[With EN/ENO]
		d:=AND_E(EN,ENO,s1,s2);
		d:=OR_E(EN,ENO,s1,s2);
		d:=XOR_E(EN,ENO,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

1. AND(_E)

- These functions perform an AND operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: WORD

(s1)	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1
								AND								
(s2)	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
								⇩								
(d)	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

2. OR(_E)

- These functions perform an OR operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: WORD

(s1)	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1
								OR								
(s2)	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0
								⇩								
(d)	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1

3. XOR(_E)

- These functions perform an XOR operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

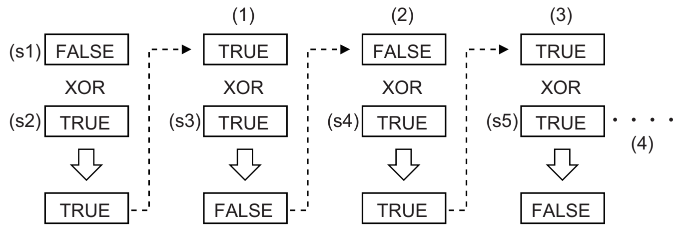
Data type: WORD

(s1)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
								XOR								
(s2)	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	0
								⇩								
(d)	1	0	1	1	0	0	0	1	0	1	0	1	1	0	1	0

- If three or more (s) settings exist, (s3) will be XORed with the result of XOR between (s1) and (s2). In addition, if (s4) exists, (s4) will be XORed with the result of subjecting (s) to XOR. After this, XOR will repeat for the number of (s) settings.

Ex.

Data type: BOOL



(1) Number of s's: 3

(2) Number of s's: 4

(3) Number of s's: 5

The XOR operation is repeated by the number of s's.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

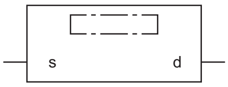
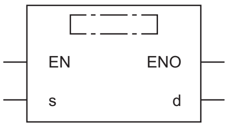
There is no operation error.

29.2 NOT Operation

NOT(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the logical NOT of input values.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design)) [With EN/ENO] d:=NOT_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)[*1]	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

*1 DX cannot be used.

Processing details

■Operation processing

- These functions perform 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

(s1)	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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

30 SELECTION FUNCTIONS

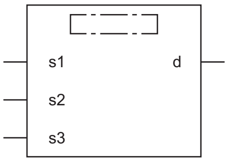
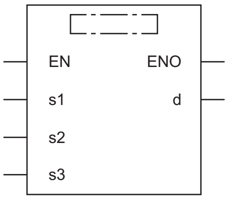
30.1 Selecting a Value

SEL(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the selected input value.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=SEL(s1,s2,s3);
		[With EN/ENO]
		d:=SEL_E(EN,ENO,s1,s2,s3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (G)	Output condition (TRUE: s3 output, FALSE: s2 output)	Input variable	BOOL
s2 (IN0)	Input	Input variable	ANY
s3 (IN1)			
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

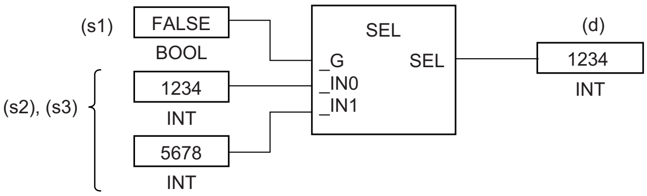
Processing details

■Operation processing

- These functions output either the (s2) or (s3) input value, in the same data type as (s2) or (s3), from (d) according to the value input to (s1).
- If the value input to (s1) is FALSE (=0), the (s2) input value is output from (d).
- If the value input to (s1) is TRUE (=1), the (s3) input value is output from (d).

Ex.

Data type of (s2) and (s3): INT (Argument names (s2) and (s3) correspond to the bit value (0 or 1) of (s1).)



- Input a BOOL data type value to (s1).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s2) and (s3).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s2) and (s3) are of STRING data type

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s2).
	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s3).
3406H	The entire string cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

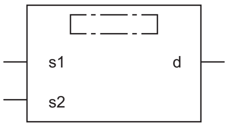
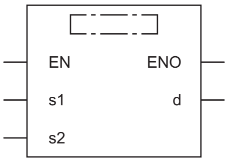
30.2 Selecting the Maximum/Minimum Value

MAX(_E), MIN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- MAX(_E): These functions output the maximum input value.
- MIN(_E): These functions output the minimum input value.

Ladder, FBD/LD*1		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=MAX(s1,s2);
		d:=MIN(s1,s2);
		[With EN/ENO]
		d:=MAX_E(EN,ENO,s1,s2);
		d:=MIN_E(EN,ENO,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_ELEMENTARY

Processing details

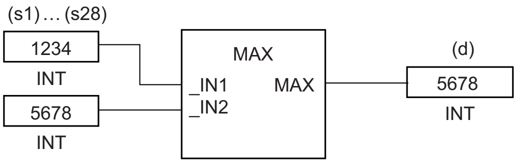
■Operation processing

- MAX(_E)

These functions output 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

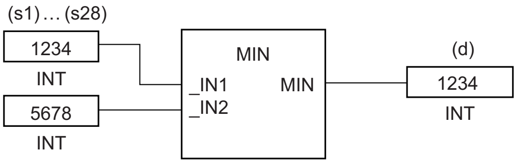


- MIN(_E)

These functions output 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	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.)

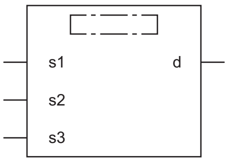
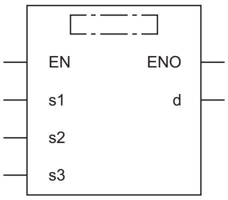
30.3 Controlling the Upper/Lower Limit

LIMIT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗RnSFCPU (Safety)

These functions output an input value that has been controlled in terms of the upper and lower limits.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LIMIT(s1,s2,s3);
		[With EN/ENO]
		d:=LIMIT_E(EN,ENO,s1,s2,s3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (MN)[*1]	Lower limit value (minimum output threshold value)	Input variable	ANY_ELEMENTARY
s2 (IN)[*1]	Input value to be controlled with the upper and lower limits	Input variable	ANY_ELEMENTARY
s3 (MX)[*1]	Upper limit value (maximum output threshold value)	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_ELEMENTARY

*1 DX cannot be used.

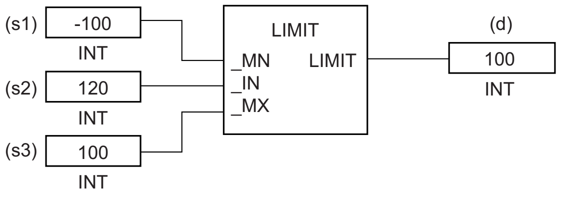
Processing details

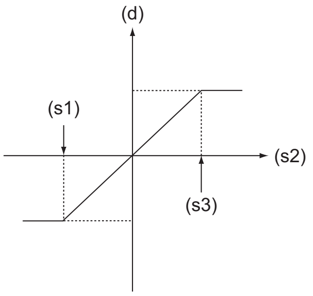
■Operation processing

- These functions output the value, in the same data type as (s1), (s2), or (s3), from (d) according to the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value input to (s1), (s2), and (s3).
  - If the input values are (s1)>(s3), the value input to (s3) is output from (d).
  - If the input values are (s2)<(s1), the value input to (s1) is output from (d).
  - If the input values are (s1)≤(s2)≤(s3), the value input to (s2) is output from (d).

Ex.

Data type: INT





- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value to (s1), (s2), and (s3), provided that the input value is (s1) < (s3).
- 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1), (s2), and (s3) are of INT or WORD data type

Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

- When (s1), (s2), and (s3) are of DINT, DWORD, or TIME data type

Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

- When (s1), (s2), or (s3) are of BOOL data type

Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of REAL data type

Error code (SD0)	Description
3402H	The value input to (s1) is out of the following range: $-2^{128}<(s1)\leq-2^{-126}$, 0, $2^{-126}\leq(s1)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is out of the following range: $-2^{128}<(s2)\leq-2^{-126}$, 0, $2^{-126}\leq(s2)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s3) is out of the following range: $-2^{128}<(s3)\leq-2^{-126}$, 0, $2^{-126}\leq(s3)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of LREAL data type

Error code (SD0)	Description
3402H	The value input to (s1) is out of the following range: $-2^{1024}<(s1)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s1)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s3) is out of the following range: $-2^{1024}<(s3)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s3)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of STRING data type

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s1), (s2), or (s3).
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).
	The number of characters in the strings input to (s1), (s2), and (s3) exceeds 16383.
3406H	The entire string cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

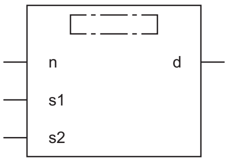
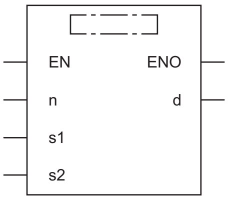
30.4 Multiplexer

MUX(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output one of the input values.

Ladder, FBD/LD[1]		Structured text[1]

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=MUX(n,s1,s2);
		[With EN/ENO]
		d:=MUX_E(EN,ENO,n,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
n(K)	Output value selection	Input variable	INT
s1 (IN0) to s28 (IN27)	Input	Input variable	ANY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

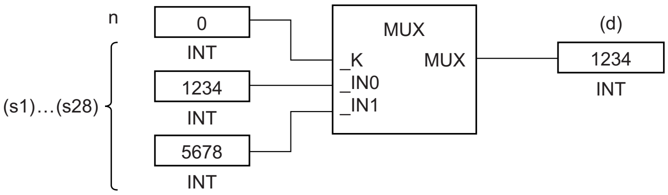
Processing details

■Operation processing

- These functions output one of the values input to (s1) to (s28), in the same data type as (s), from (d) according to the value input to (n).
- If the (n) input value is 0, the value input to (s1) is output from (d).
- If the (n) input value is (n)-1, the value input to (sn) is output from (d).

Ex.

Data type: INT



- If a value outside the range of the number of pins in (s) is input to (n), an undefined value is output from (d). (No operation error occurs. When MUX_E is used, ENO outputs FALSE.)
- Input an INT data type value to (n) within the range of 0 to 27, provided that it is within the range of the number of pins in (s).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	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.)

31 COMPARISON FUNCTIONS

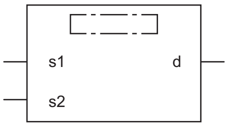
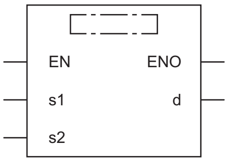
31.1 Comparing Data

GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the comparison result of input values.

Ladder, FBD/LD*1		Structured text*1

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]*2
		d:=GT(s1,s2);
		d:=GE(s1,s2);
		d:=EQ(s1,s2);
		d:=LE(s1,s2);
		d:=LT(s1,s2);
		[With EN/ENO]
		d:=GT_E(EN,ENO,s1,s2);
		d:=GE_E(EN,ENO,s1,s2);
		d:=EQ_E(EN,ENO,s1,s2);
		d:=LE_E(EN,ENO,s1,s2);
		d:=LT_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

*2 The engineering tool with version "1.035M" or later supports the ST. The function is described as an operator when an engineering tool with an earlier version is used. (📖 MELSEC iQ-R Programming Manual (Program Design))

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)*1	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

*1 DX cannot be used.

Processing details

■Operation processing

- These functions perform comparison operation between the input values to (s), and output the operation result, in the BOOL data type, from (d).
  - GT(_E): Performs comparison of $[(s1)>(s2)]\&[(s2)>(s3)]\&\cdots\&[(s)_{(n-1)}>(s)_{(n)}]$.
    - If all values satisfy $(s)_{(n-1)}>(s)_{(n)}$, TRUE is output.
    - If one the values satisfies $(s)_{(n-1)}\leq(s)_{(n)}$, FALSE is output.
  - GE(_E): Performs comparison of $[(s1)\geq(s2)]\&[(s2)\geq(s3)]\&\cdots\&[(s)_{(n-1)}\geq(s)_{(n)}]$.
    - If all values satisfy $(s)_{(n-1)}\geq(s)_{(n)}$, TRUE is output.
    - If one the values satisfies $(s)_{(n-1)}<(s)_{(n)}$, FALSE is output.
  - EQ(_E): Performs comparison of $[(s1)=(s2)]\&[(s2)=(s3)]\&\cdots\&[(s)_{(n-1)}=(s)_{(n)}]$.
    - If all values satisfy $(s)_{(n-1)}=(s)_{(n)}$, TRUE is output.
    - If one the values satisfies $(s)_{(n-1)}\neq(s)_{(n)}$, FALSE is output.
  - LE(_E): Performs comparison of $[(s1)\leq(s2)]\&[(s2)\leq(s3)]\&\cdots\&[(s)_{(n-1)}\leq(s)_{(n)}]$.
    - If all values satisfy $(s)_{(n-1)}\leq(s)_{(n)}$, TRUE is output.
    - If one the values satisfies $(s)_{(n-1)}>(s)_{(n)}$, FALSE is output.
  - LT(_E): Performs comparison of $[(s1)<(s2)]\&[(s2)<(s3)]\&\cdots\&[(s)_{(n-1)}<(s)_{(n)}]$.
    - If all values satisfy $(s)_{(n-1)}<(s)_{(n)}$, TRUE is output.
    - If one the values satisfies $(s)_{(n-1)}\geq(s)_{(n)}$, FALSE is output.
- Input an INT, DINT, REAL, LREAL, BOOL, WORD, DWORD, TIME, or STRING type data value to (s). No WSTRING type Unicode string can be specified.

- 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	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.)

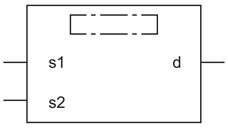
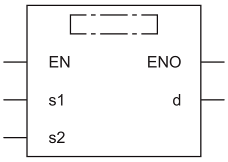
31.2 Comparing Data

NE(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the comparison result of input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]*1
		d:=NE(s1,s2);
		[With EN/ENO]
		d:=NE_E(EN,ENO,s1,s2);

*1 The engineering tool with version "1.035M" or later supports the ST. The function is described as an operator when an engineering tool with an earlier version is used. (📖 MELSEC iQ-R Programming Manual (Program Design))

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1, s2	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

Processing details

■Operation processing

- These functions perform comparison operation between the input values to (s), and output the operation result, in the BOOL data type, from (d).
  - NE(_E): Performs comparison of [(s1)≠(s2)].
    - If (s1)≠(s2), TRUE is output.
    - If (s1)=(s2), FALSE is output.
- Input an INT, DINT, REAL, LREAL, BOOL, WORD, DWORD, TIME, or STRING type data value to (s). No WSTRING type Unicode string can be specified.
- 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

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) in the device/label memory.
3405H	The number of characters in the string input to (s) exceeds 16383.
3406H	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.)

32 STRING FUNCTIONS

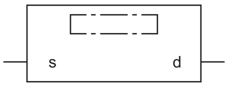
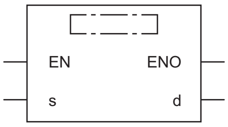
32.1 Detecting a String Length

LEN(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These functions detect and output the length of the string input.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LEN(s);
		[With EN/ENO]
		d:=LEN_E(EN,ENO,s);

Setting data

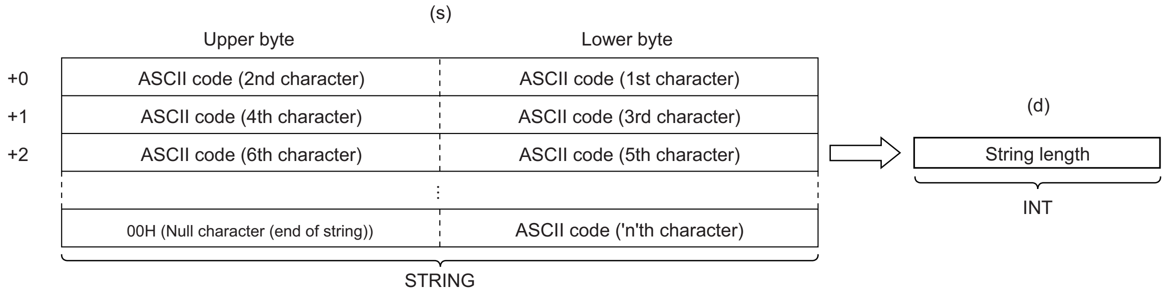
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions detect the length of the string input to (s), and output the length from (d).



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

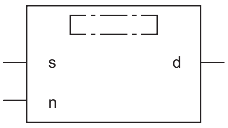
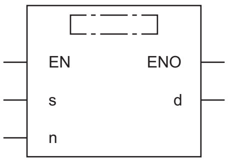
32.2 Extracting String Data From the Left/Right

LEFT(_E), RIGHT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- LEFT(_E): These functions extract and output the specified number of characters, starting from the left end of the string input.
- RIGHT(_E): These functions extract and output the specified number of characters, starting from the right end of the string input.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LEFT(s,n);
		d:=RIGHT(s,n);
		[With EN/ENO]
		d:=LEFT_E(EN,ENO,s,n);
		d:=RIGHT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n (L)	Number of characters to be extracted	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

Processing details

■Operation processing

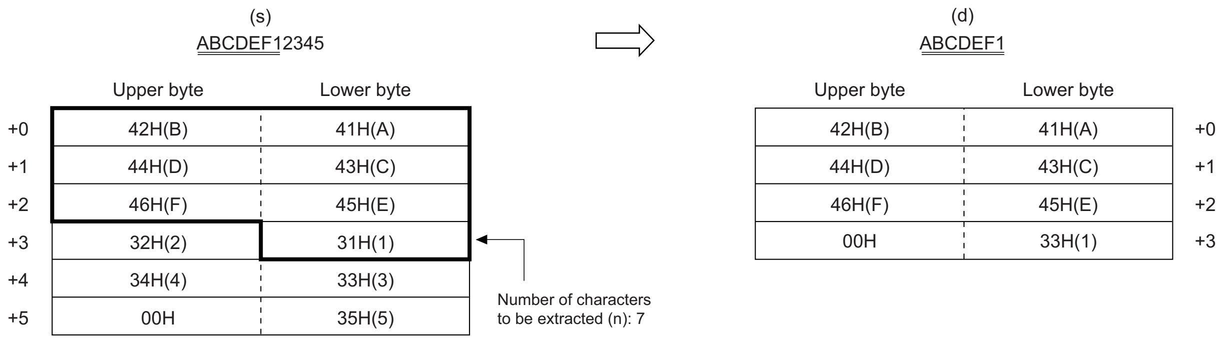
- LEFT(_E)

These functions extract the specified number of characters, starting from the left end of the string input to (s), and output the operation result from (d).

Specify the number of characters to be extracted in (n).

Ex.

When (n)=7



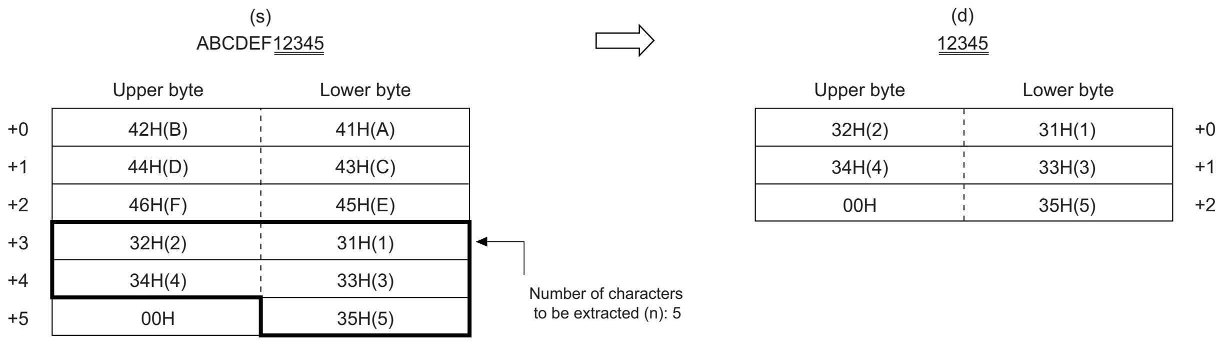
- RIGHT(_E)

These functions extract the specified number of characters, starting from the right end of the string input to (s), and output the operation result from (d).

Specify the number of characters to be extracted in (n).

Ex.

When (n)=5



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

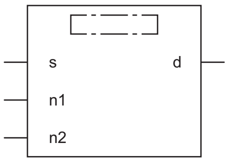
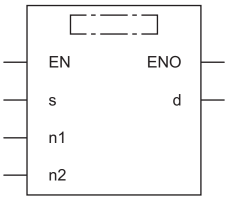
32.3 Extracting String Data

MID(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions extract and output the specified number of characters, starting from the specified position of the string input.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=MID(s,n1,n2);
		[With EN/ENO]
		d:=MID_E(EN,ENO,s,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be extracted	Input variable	INT
n2 (P)	Extraction target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

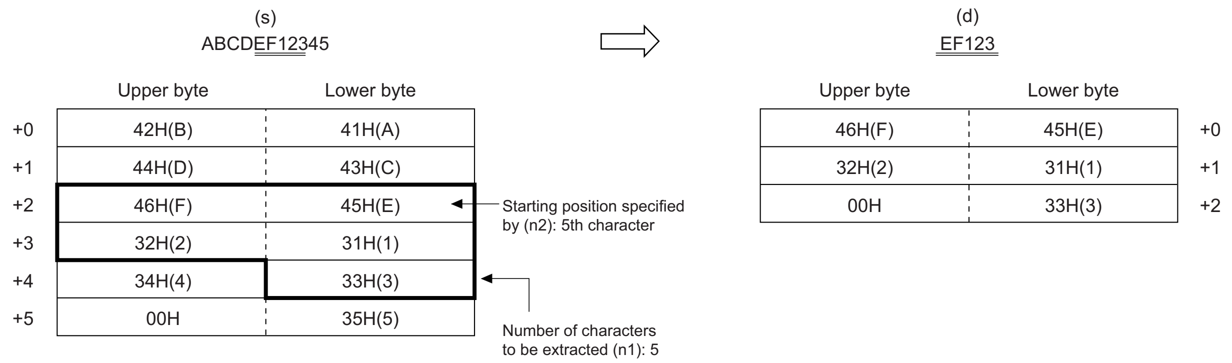
Processing details

■Operation processing

- These functions extract the specified number of characters, starting from the specified position of the string input to (s), and output the operation result from (d).
- Specify the number of characters to be extracted in (n1).
- Specify the start position of the string to be extracted in (n2).

Ex.

When (n1)=5, (n2)=5



	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	46H(F)	45H(E)
+3	32H(2)	31H(1)
+4	34H(4)	33H(3)
+5	00H	35H(5)

Upper byte	Lower byte	
46H(F)	45H(E)	+0
32H(2)	31H(1)	+1
00H	33H(3)	+2

- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) in the device/label memory.
3405H	The number of characters in the string input to (s) exceeds 16383.
	Out-of-range data is set to (n1) or (n2). • The value input to (n1) or (n2) is 0 or smaller. • The value input to (n2) is other than the valid values (-1, 0, 1 or bigger). • The value input to (n1) exceeds the number of characters in (s). • The sum of (n1) and (n2) exceeds the number of characters in (s).

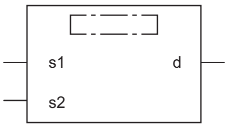
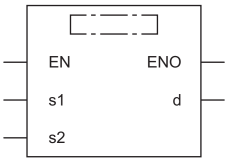
32.4 Concatenating String Data

CONCAT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions concatenate character strings, and output the operation result.

Ladder, FBD/LD[*1]		**Structured text**[*1]

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=CONCAT(s1,s2);
		[With EN/ENO]
		d:=CONCAT_E(EN,ENO,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
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

Processing details

■Operation processing

- These functions concatenate the strings input to (s2) to (s28) to the end of the string input to (s1), and output the operation result from (d).
- The (s2) to (s28) strings are concatenated successively, ignoring 00H, which indicates the end of the (s1) string.
- If the string after concatenation exceeds 255 bytes, the substring up to the 255th byte will be output.

(s1) ABCDE + (s2) 123456 ⇨ (d) ABCDE123456

(s1)

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	00H	45H(E)

(s2)

	Upper byte	Lower byte
+0	32H(2)	31H(1)
+1	34H(4)	33H(3)
+2	36H(6)	35H(5)
+3	00H	

(d)

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	31H(1)	45H(E)
+3	33H(3)	32H(2)
+4	35H(5)	34H(4)
+5	00H	36H(6)

- Input a STRING data type value to (s1) and (s2) to (s28) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The device numbers are overlapping between (s1) to (s28) and (d).
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
	The number of characters in the strings input to (s1) to (s28) is 0.
	The number of characters of the character string in the device specified by (d) exceeds 16383.
3406H	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

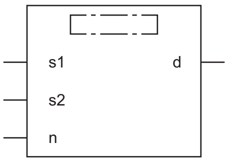
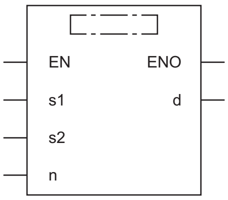
32.5 Inserting String Data

INSERT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions insert a character string into another string, and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INSERT(s1,s2,n);
		[With EN/ENO]
		d:=INSERT_E(EN,ENO,s1,s2,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
n (P)	Insertion target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

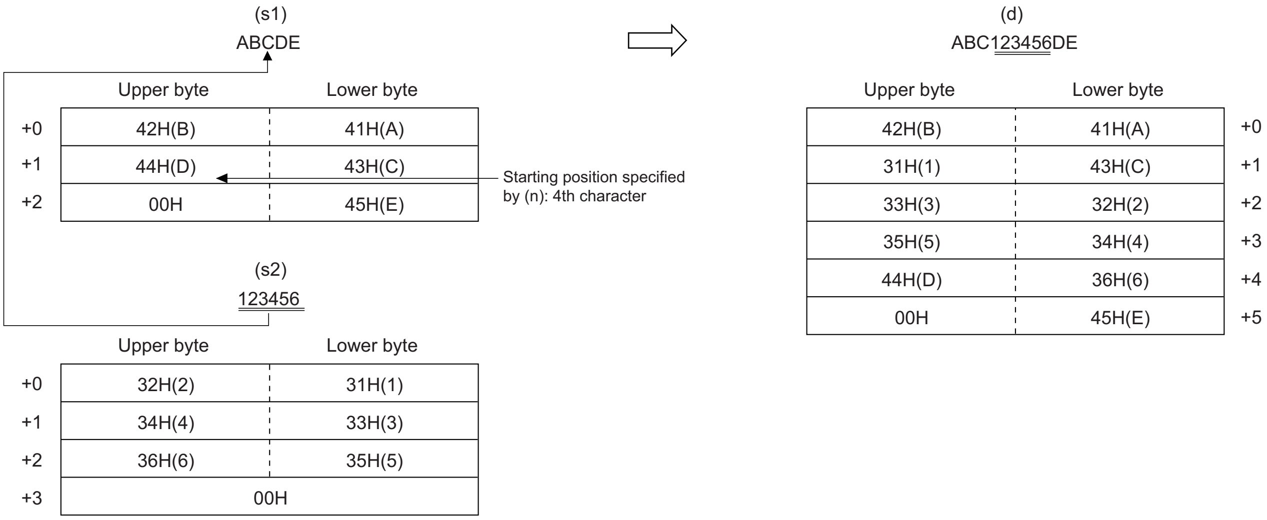
Processing details

■Operation processing

- These functions insert the string input to (s2) into the insertion start position, i.e. the 'n'th character position from the beginning of the string input to (s1), and output the operation result from (d).
- After the (s2) string is inserted into the (s1) string, 00H, which indicates the end of the (s2) string, is ignored.
- If the string after insertion exceeds 255 bytes, the substring up to the 255th byte will be output.

Ex.

When (n)=4



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.
- Input an INT data type value to (n) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s1).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area specified by (s1) to (s28) in the device/label memory.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The device numbers are overlapping between (s1) to (s28) and (d).
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
	The number of characters in the strings input to (s1) to (s28) is 0.
	The number of characters of the character string in the device specified by (d) exceeds 16383.
3406H	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

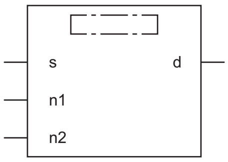
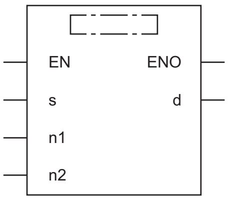
32.6 Deleting String Data

DELETE(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These functions delete the specified range in a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DELETE(s,n1,n2);
		[With EN/ENO]
		d:=DELETE_E(EN,ENO,s,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be deleted	Input variable	INT
n2 (P)	Deletion target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

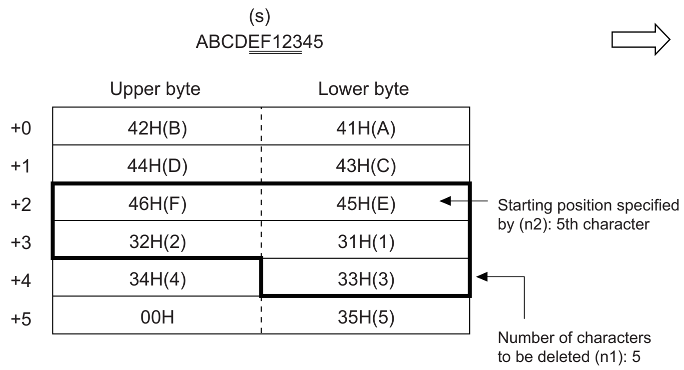
Processing details

■Operation processing

- These functions delete the specified number of characters, starting from the desired position of the string input to (s), and output the remaining substring from (d).
- Specify the number of characters to be deleted in (n1).
- Specify the start position of the string to be deleted in (n2).

Ex.

When (n1)=5, (n2)=5



(s)
ABCDEF12345

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	46H(F)	45H(E)
+3	32H(2)	31H(1)
+4	34H(4)	33H(3)
+5	00H	35H(5)

Starting position specified by (n2): 5th character

Number of characters to be deleted (n1): 5

(d)
ABCD45

Upper byte	Lower byte	
42H(B)	41H(A)	+0
44H(D)	43H(C)	+1
35H(5)	34H(4)	+2
00H		+3

- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s).
3405H	The number of characters in the string input to (s) exceeds 255.
	The value input to (n1) is out of the range, 0 to 255.
	The value input to (n2) is out of the range, 1 to 255.
	The value input to (n1) exceeds the number of characters in (s).
	The value input to (n2) exceeds the number of characters in (s).
3406H	The entire string after delete processing cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d).

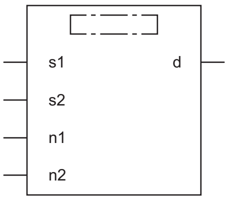
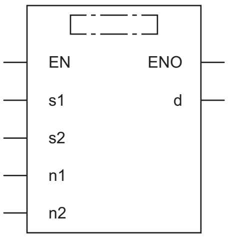
32.7 Replacing String Data

REPLACE(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions replace the specified range in a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=REPLACE(s1,s2,n1,n2);
		[With EN/ENO]
		d:=REPLACE_E(EN,ENO,s1,s2,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be replaced	Input variable	INT
n2 (P)	Replacement target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

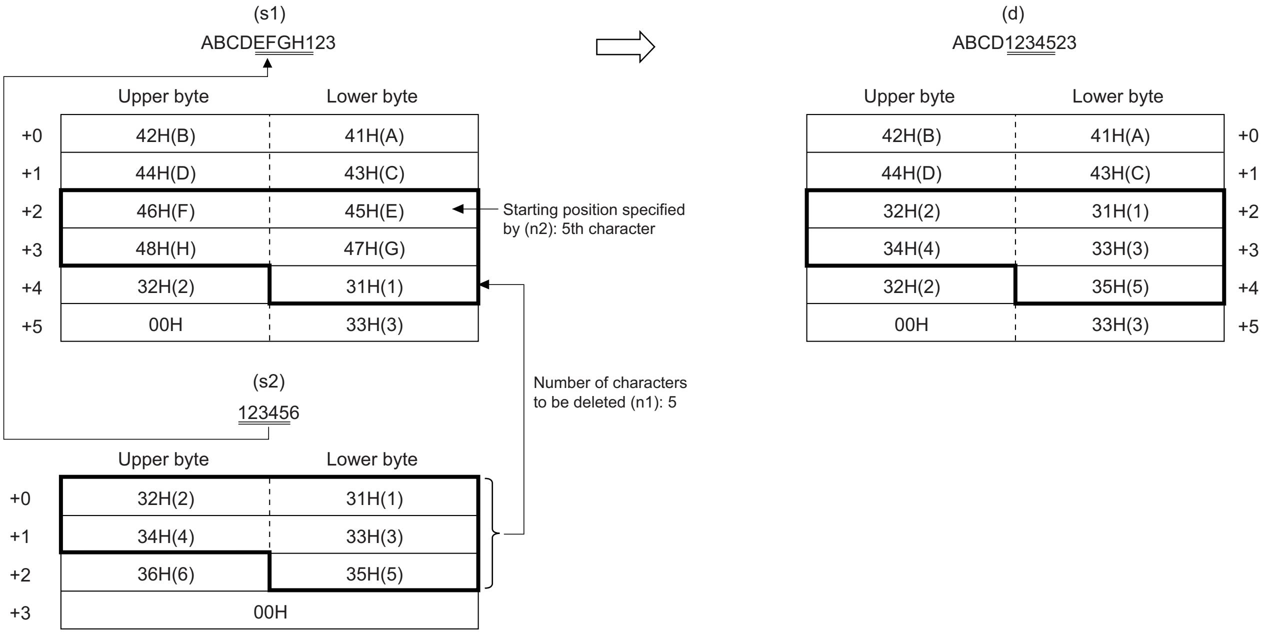
Processing details

■Operation processing

- These functions replace the specified number of characters starting from the desired position of the string input to (s1) with the string input to (s2), and output the operation result from (d).
- Specify the number of characters to be replaced in (n1).
- Specify the start position of the string to be replaced in (n2).

Ex.

When (n1)=5, (n2)=5



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s1).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s1).

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

---

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s1).
	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s2).
3405H	The number of characters in the string input to (s1) exceeds 255.
	The number of characters in the string input to (s2) exceeds 255.
	The value input to (n1) is out of the range, 0 to 255.
	The value input to (n2) is out of the range, 1 to 255.
	The value input to (n1) exceeds the number of characters in (s2).
	The value input to (n2) exceeds the number of characters in (s1).
3406H	The entire string after delete processing cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d).

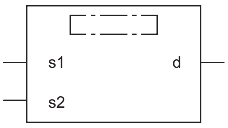
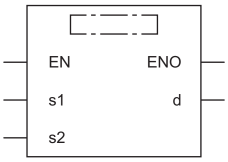
32.8 Searching String Data

FIND(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions search a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=FIND(s1,s2);
		[With EN/ENO]
		d:=FIND_E(EN,ENO,s1,s2);

Setting data

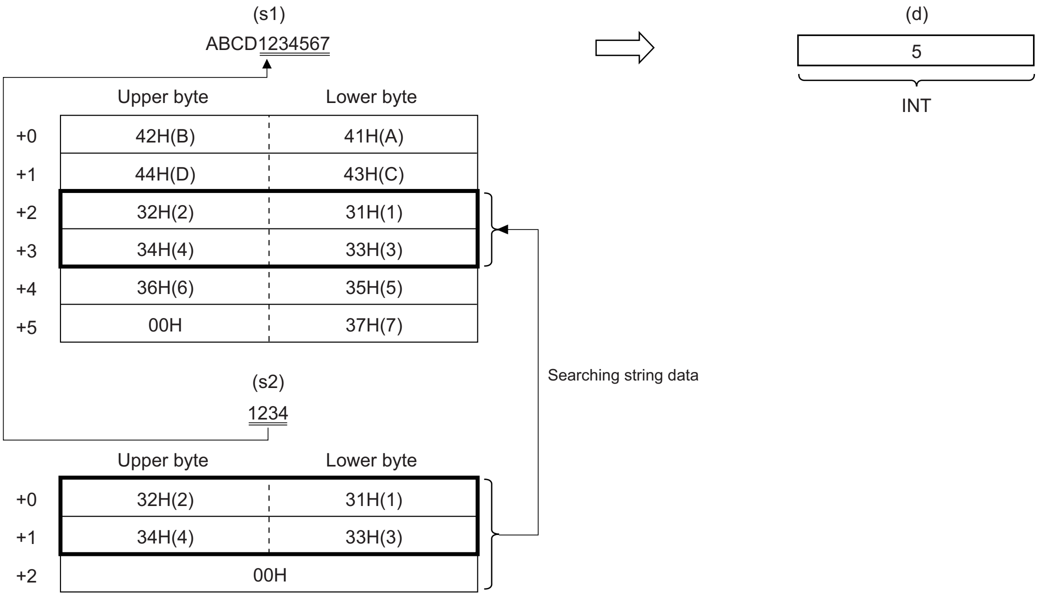
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions search the string input to (s2) from the beginning of the string input to (s1), and output the search result from (d).
- The start character position of the first string found is output as the search result.
- If the (s2) string is not found in the (s1) string, 0 will be output.



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

33 TIME DATA TYPE FUNCTIONS

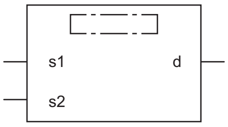
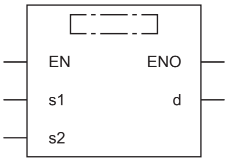
33.1 Addition

ADD_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the sum ((s1)+(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ADD_TIME(s1,s2);
		[With EN/ENO]
		d:=ADD_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

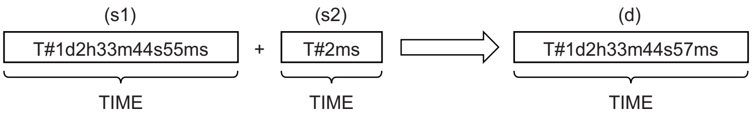
Processing details

■Operation processing

- These functions perform addition of the TIME data type values input to (s1) and (s2) ((s1)+(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

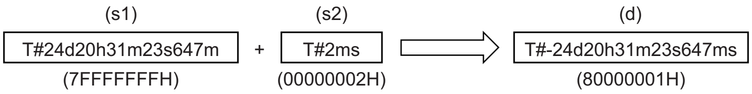
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=T#2ms (2 milliseconds)



- Input a TIME data type value to (s1) and (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When ADD_TIME_E is used, ENO outputs TRUE.

Ex.

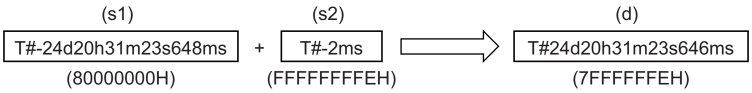
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

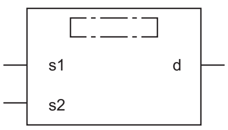
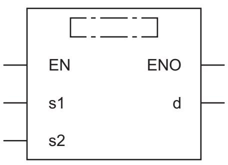
33.2 Subtraction

SUB_TIME(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** **RnSFCPU (Safety)**

These functions output the difference ((s1)-(s2)) between the TIME data type input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=SUB_TIME(s1,s2);
		[With EN/ENO]
		d:=SUB_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

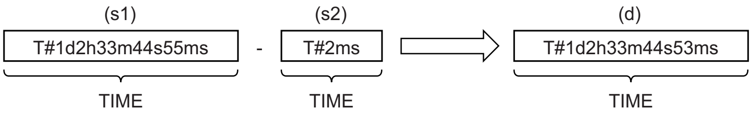
Processing details

■Operation processing

- These functions perform subtraction between the TIME data type values input to (s1) and (s2) ((s1)-(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

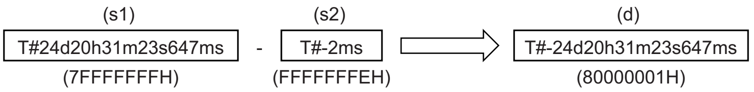
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=T#2ms (2 milliseconds)



- Input a TIME data type value to (s1) and (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When SUB_TIME_E is used, ENO outputs TRUE.

Ex.

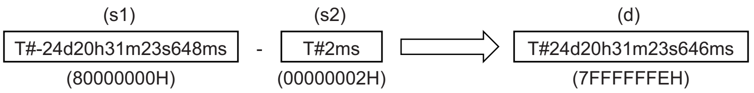
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[*1]	Undefined value

[*1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

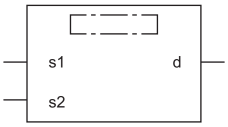
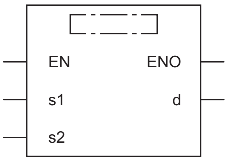
33.3 Multiplication

MUL_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the product ((s1)×(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=MUL_TIME(s1,s2);
		[With EN/ENO]
		d:=MUL_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Input	Input variable	TIME
s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

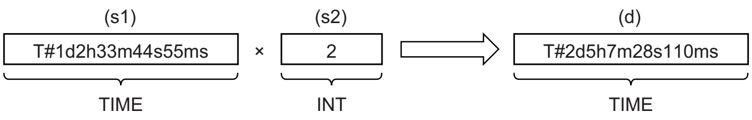
Processing details

■Operation processing

- These functions perform multiplication between the TIME data type values input to (s1) and (s2) ((s1)×(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

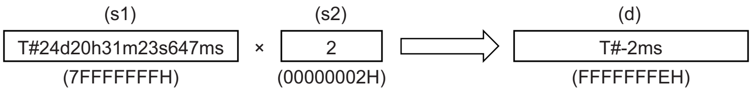
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=2



- Input a TIME data type value to (s1).
- Input an INT, DINT, REAL, or LREAL data type value to (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When MUL_TIME_E is used, ENO outputs TRUE. (In this case, the output value is of TIME data type with the upper 32 bits deleted although the operation result is 64-bit data.)

Ex.

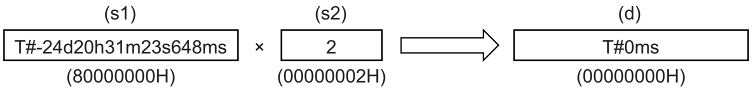
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s2) is of LREAL data type

Error code (SD0)	Description
3402H	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The single-precision real number input to (s2) is out of the range, -2147483648 to 2147483647.

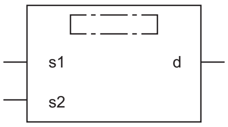
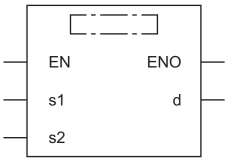
33.4 Division

DIV_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the quotient ((s1)÷(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DIV_TIME(s1,s2);
		[With EN/ENO]
		d:=DIV_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Input	Input variable	TIME
s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

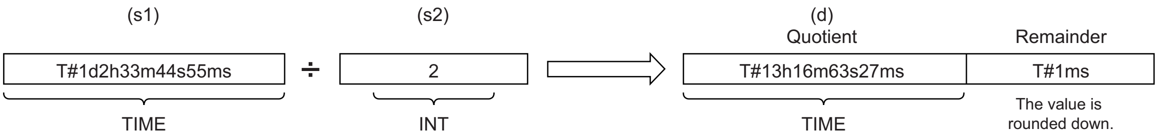
Processing details

■Operation processing

- These functions perform division between the TIME data type values input to (s1) and (s2) ((s1)÷(s2)), and output the operation result, in the TIME data type, from (d). The remainder is rounded down.

Ex.

When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=2



- Input a TIME data type value to (s1).
- Input an INT, DINT, REAL, or LREAL data type value to (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

1. Function without EN/ENO

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

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE[1]	Undefined value

[1] If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3400H	The value input to (s2) is 0. (The value is divided by zero.)

- When (s2) is of LREAL data type

Error code (SD0)	Description
3402H	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
3405H	The data input to (s2) is out of the range, -2147483648 to 2147483647.

MEMO

PART 6

STANDARD FUNCTION BLOCKS

This part consists of the following chapters.

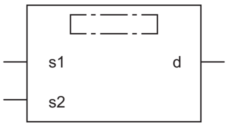
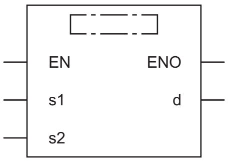
34 BISTABLE FUNCTION BLOCKS

34.1 Bistable Function Block (Set-Dominant)

SR(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(S1:=s1,R:=s2,Q1:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,S1:=s1,R:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (S1)	Set command	Input variable	BOOL
s2 (R)	Reset command	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

■Operation processing

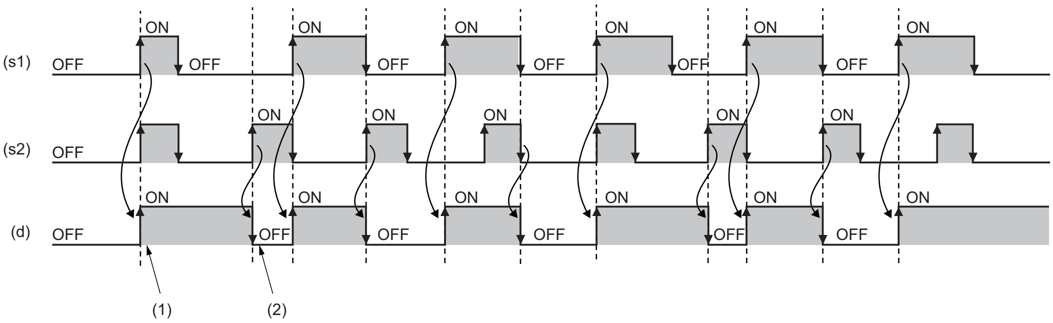
- When (s1) turns on, (d) is set. Turning on (s2) while (s1) is off resets (d).
- Even when (s2) turns on while (s1) is on, (d) is not reset.

■Operation result

1. Function block without EN/ENO

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

- Timing chart



(1) When (s1) turns on, (d) turns on.

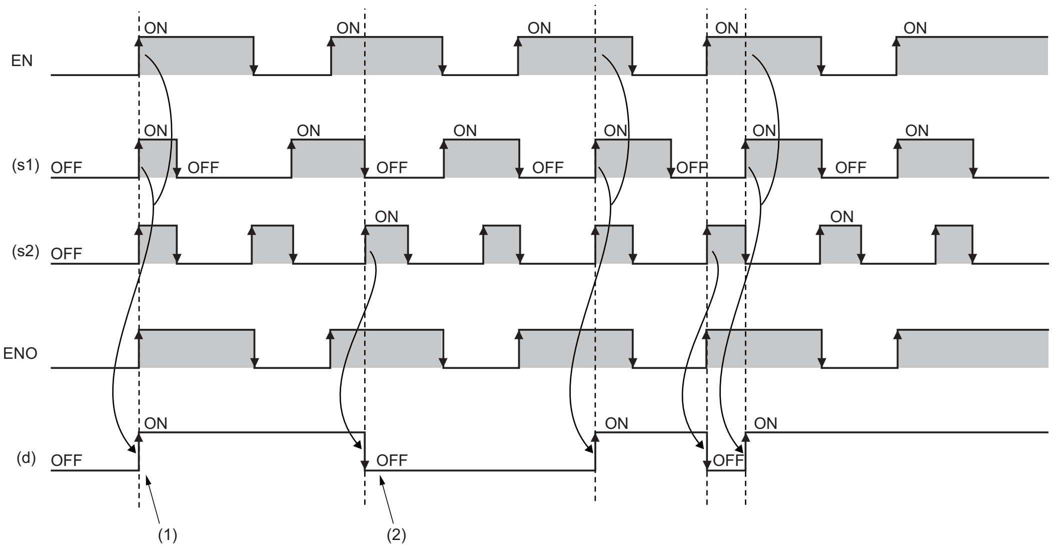
(2) When (s2) turns on while (s1) is off, (d) turns off.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) When (s1) turns on while EN is on, (d) turns on.

(2) When (s2) turns on while EN is on and (s1) is off, (d) turns off.

Operation error

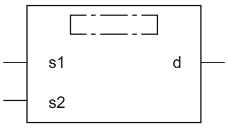
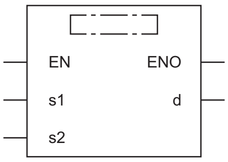
There is no operation error.

34.2 Bistable Function Block (Reset-Dominant)

RS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(S:=s1,R1:=s2,Q1:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,S:=s1,R1:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (S)	Set command	Input variable	BOOL
s2 (R1)	Reset command	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

■Operation processing

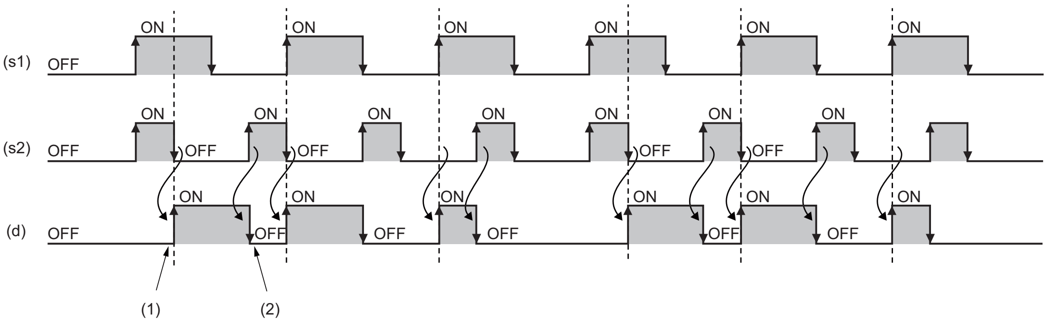
- When (s1) turns on, (d) is set. When (s2) turns on, (d) is reset.
- Even when (s1) turns on while (s2) is on, (d) is not reset.

■Operation result

1. Function block without EN/ENO

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.

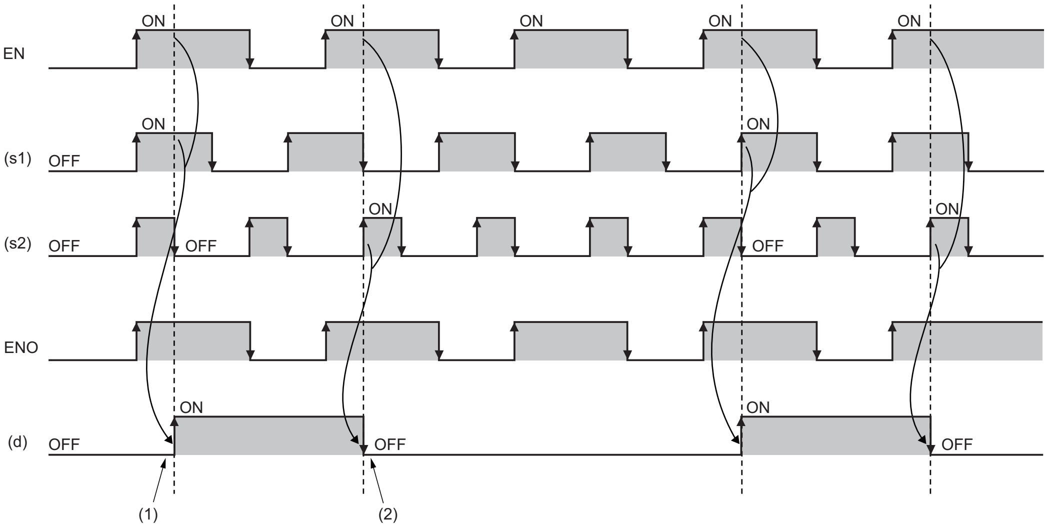
(2) When (s2) turns on, (d) turns off.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) When (s2) turns off while EN is on and (s1) is on, (d) turns on.

(2) When (s2) turns on while EN is on, (d) turns off.

Operation error

There is no operation error.

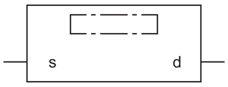
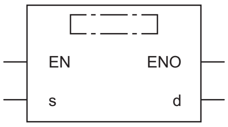
35 EDGE DETECTION FUNCTION BLOCKS

35.1 Detecting a Rising Edge

R_TRIG(_E)

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ✕ **RnSFCPU (Safety)**

These function blocks detect a signal rising edge, and outputs the pulse signal.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CLK:=s,Q:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (CLK)	Rising edge detection input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

■Operation processing

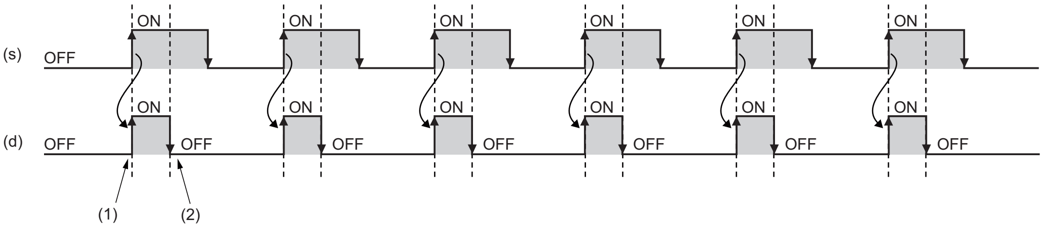
When (s) turns on, (d) turns on only for one scan.

■Operation result

1. Function block without EN/ENO

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

- Timing chart



(1) (d) turns on at the rising edge of (s).

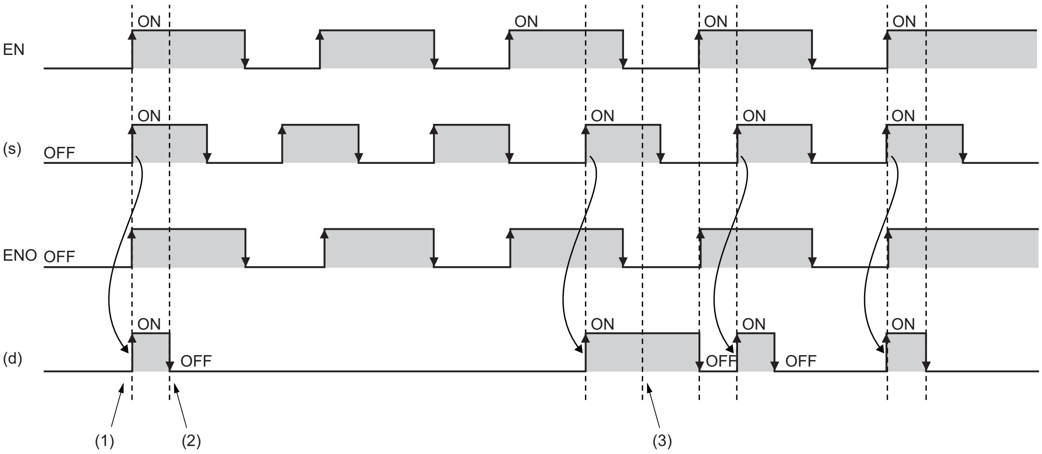
(2) (d) turns off in the next scan.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) (d) turns on at the rising edge of (s) while EN is on.

(2) (d) turns off in the next scan.

(3) If EN is off, (d) holds the output value of the last scan.

Operation error

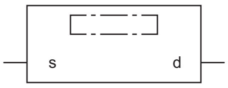
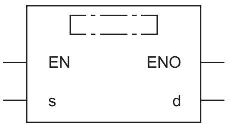
There is no operation error.

35.2 Detecting a Falling Edge

F_TRIG(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

These function blocks detect a signal falling edge, and outputs the pulse signal.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CLK:=s,Q:=d); [With EN/ENO] Instance name(EN:= en,ENO:=eno,CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (CLK)	Falling edge detection input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

■Operation processing

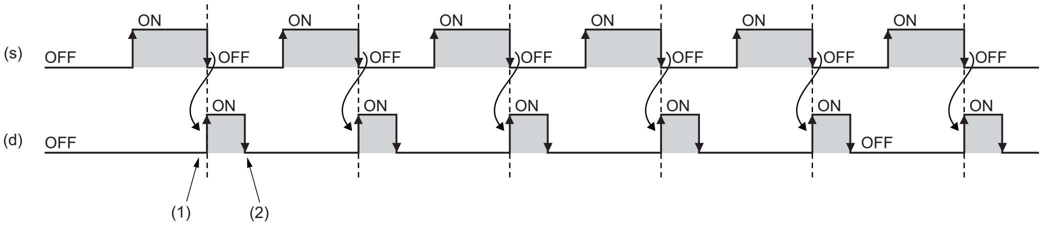
When (s) turns off, (d) turns on only for one scan.

■Operation result

1. Function block without EN/ENO

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

- Timing chart



(1) (d) turns on at the falling edge of (s).

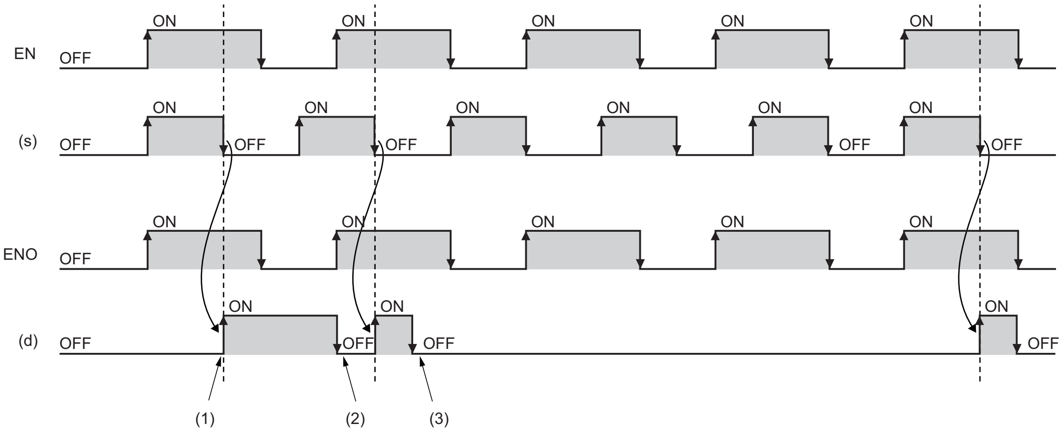
(2) (d) turns off in the next scan.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) (d) turns on at the falling edge of (s) while EN is on.

(2) (d) turns off in the next scan.

(3) If EN is off, (d) holds the output value of the last scan.

Operation error

There is no operation error.

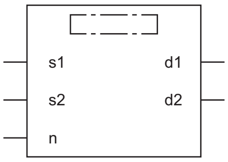
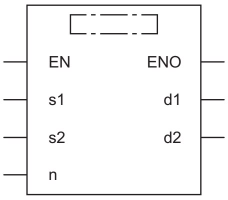
36 COUNTER/TIMER FUNCTION BLOCKS

36.1 Up Counter

CTU(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

These function blocks count up the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (CU)	Count signal input	Input variable	BOOL
s2 (R)	Count value reset	Input variable	BOOL
n (PV)	Maximum count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	End of count	Output variable	BOOL
d2 (CV)	Count value	Output variable	INT

Processing details

■Operation processing

1. Counting up

- When (s1) changes from off to on, the value in (d2) is counted up by one.
- When (d2) reaches the (n) value, (d1) turns on and the counting stops.
- Set the maximum counter value to (n). When (s2) turns on, (d1) turns off and (d2) is set to 0.

2. Maximum count value

The valid setting range of (n) is 0 to 32767.

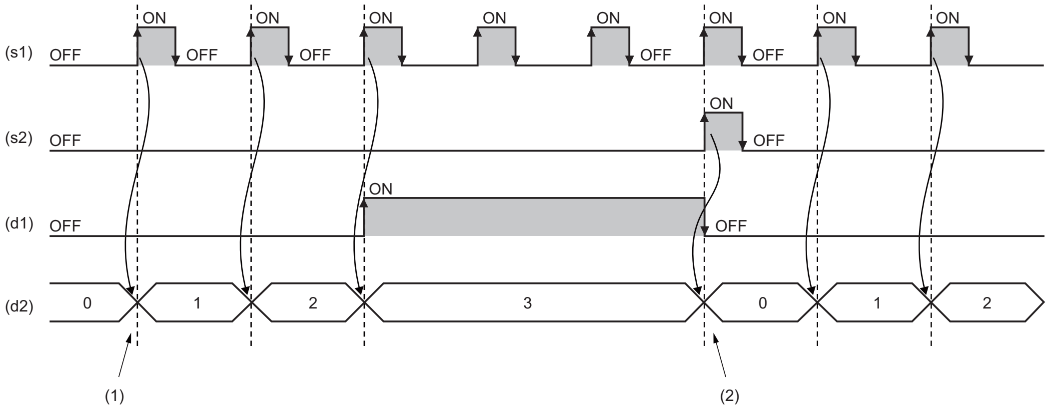
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1) and (d2).

- Timing chart

When (n)=3



(1) When (s1) turns on, the value in (d2) is counted up.

(2) When (s2) turns on, the value in (d2) is initialized.

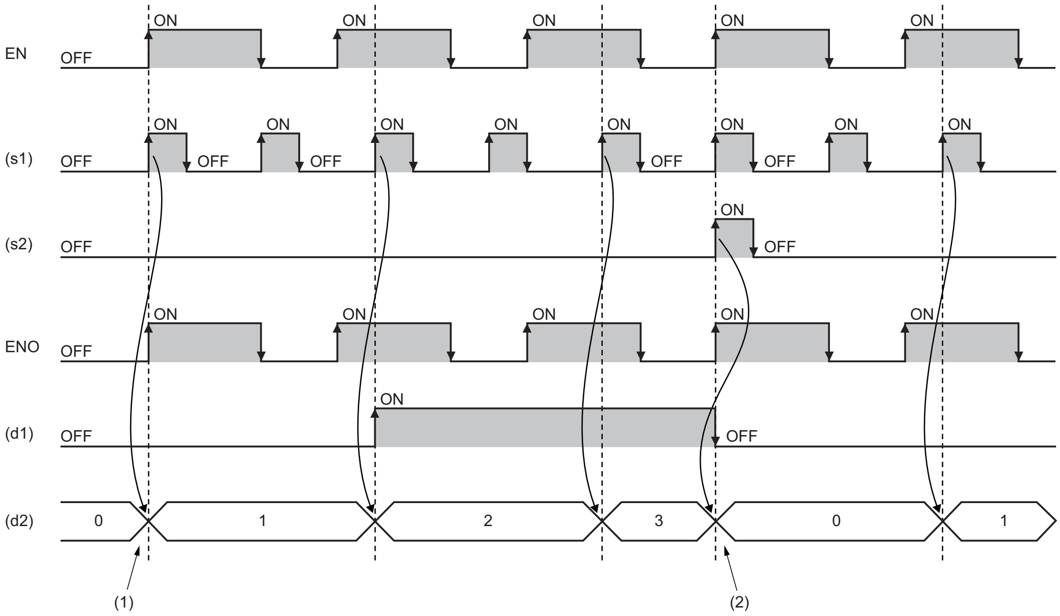
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When (n)=3



(1) When (s1) turns on while EN is on, the value in (d2) is counted up.

(2) When (s2) turns on while EN is on, the value in (d2) is initialized.

Operation error

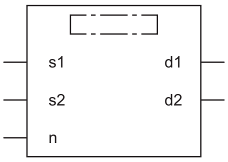
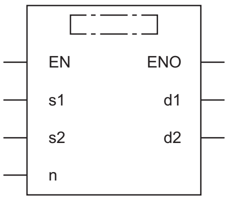
There is no operation error.

36.2 Down Counter

CTD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety) ✗

These function blocks count down the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1(CD)	Count signal input	Input variable	BOOL
s2 (LD)	Count value set	Input variable	BOOL
n (PV)	Start count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	End of count	Output variable	BOOL
d2 (CV)	Count value	Output variable	INT

Processing details

■Operation processing

1. Counting down

- When (s1) changes from off to on, the value in (d2) is counted down by one.
- When (d2) is 0, (d1) turns on and the counting stops.
- Set the start count value to (n). When (s2) turns on, (d1) turns off and (n) is set to (d2).

2. Start count value

The valid setting range of (n) is 0 to 32767.

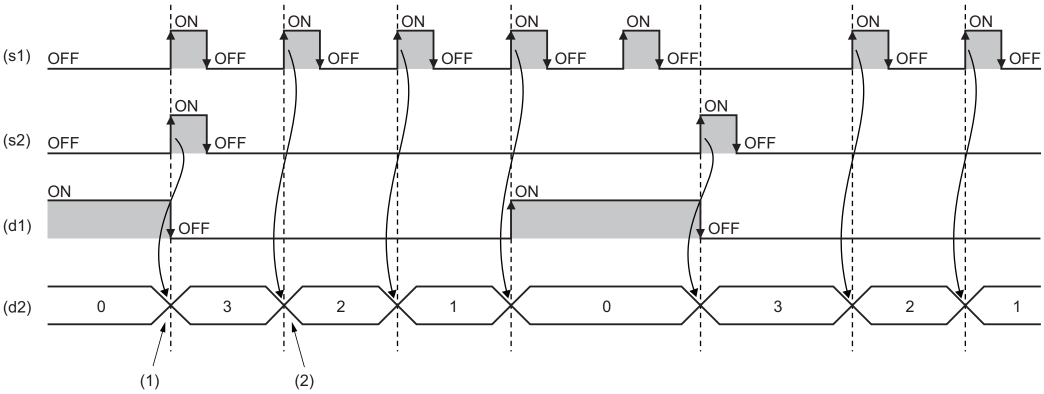
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1) and (d2).

- Timing chart

When (n)=3



(1) When (s2) turns on, the value in (d2) is initialized.

(2) When (s1) turns on, the value in (d2) is counted down.

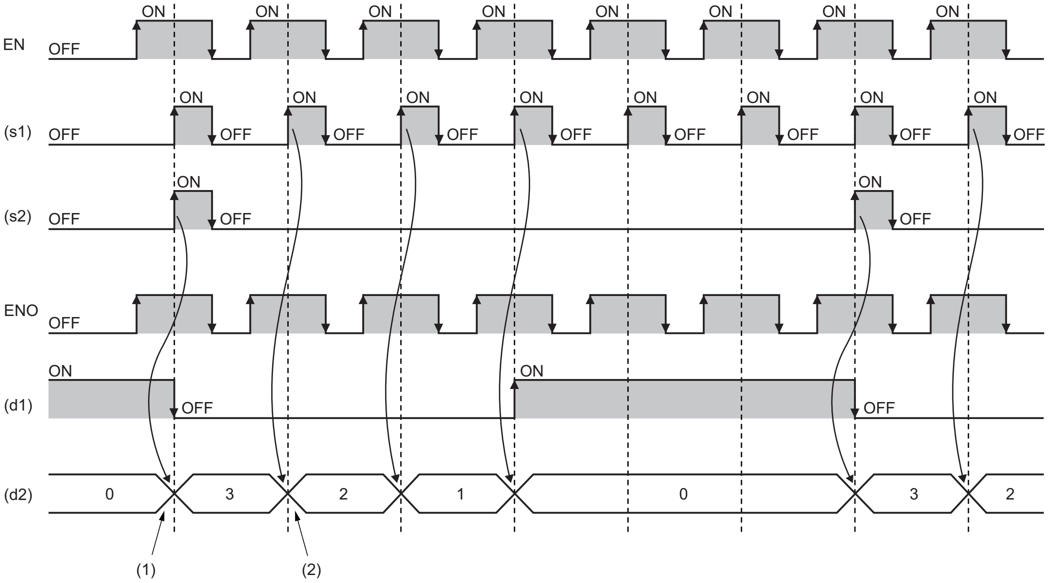
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When (n)=3



(1) When (s2) turns on while EN is on, the value in (d2) is initialized.

(2) When (s1) turns on while EN is on, the value in (d2) is counted down.

Operation error

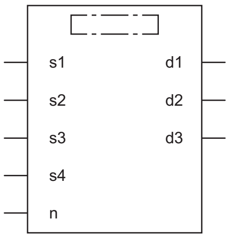
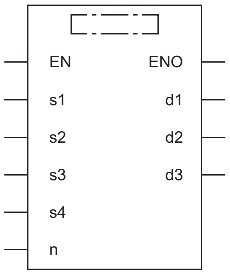
There is no operation error.

36.3 Up/Down Counter

CTUD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These function blocks count up or down the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (CU)	Count signal input (for counting-up)	Input variable	BOOL
s2 (CD)	Count signal input (for counting-down)	Input variable	BOOL
s3 (R)	Count value reset	Input variable	BOOL
s4 (LD)	Count value set	Input variable	BOOL
n (PV)	Maximum count value or start count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (QU)	End of count (for counting-up)	Output variable	BOOL
d2 (QD)	End of count (for counting-down)	Output variable	BOOL
d3 (CV)	Current count value	Output variable	INT

Processing details

■Operation processing

In (n), set the maximum count value for up count and the start count value for down count. The valid range of (n) is 0 to 32767.

1. Counting up

- When (s1) changes from off to on, the value in (d3) is counted up by one.
- When (d3) reaches the (n) value, (d1) turns on and the counting stops.
- When (s3) turns on, (d1) turns off and (d3) is set to 0.

2. Counting down

- When (s2) changes from off to on, the value in (d3) is counted down by one.
- When (d3) is 0, (d2) turns on and the counting stops.
- When (s4) turns on, (d2) turns off and (n) is set to (d3).

3. Others

- When (s1) and (s2) change from off to on simultaneously, the value in (d3) is counted up by one with priority given to (s1).
- When (s3) and (s4) turn on simultaneously, (d3) is set to 0 with priority given to (s3).

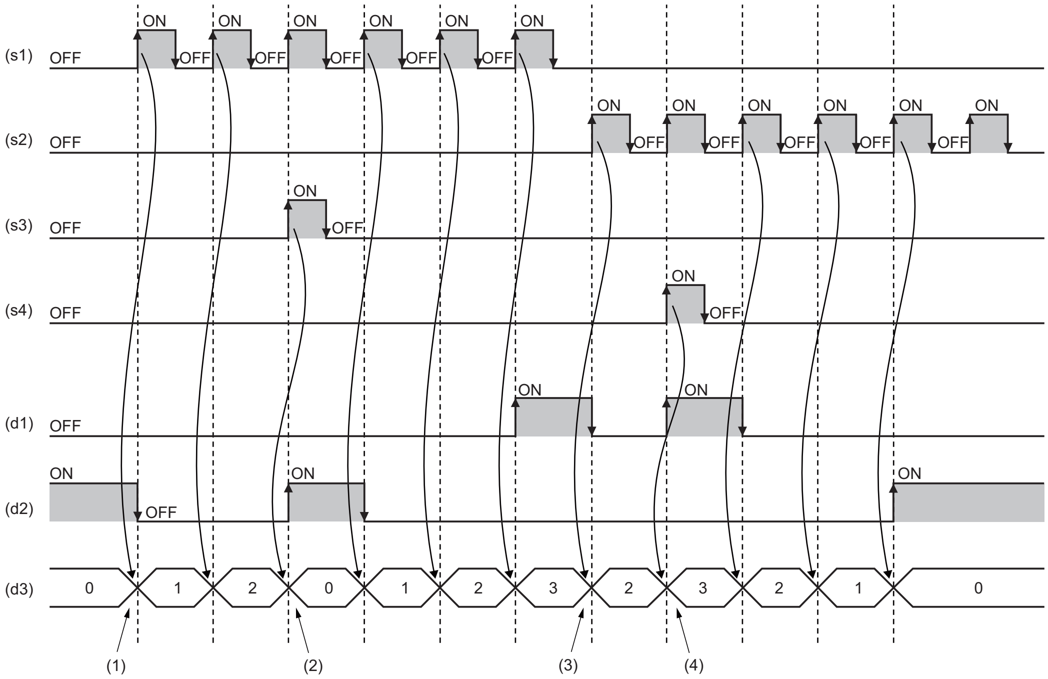
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1), (d2), and (d3).

- Timing chart

When (n)=3



(1) When (s1) turns on, the value in (d3) is counted up.
(2) When (s3) turns on, the value in (d3) is initialized.
(3) When (s2) turns on, the value in (d3) is counted down.
(4) When (s4) turns on, the value in (d3) is initialized.

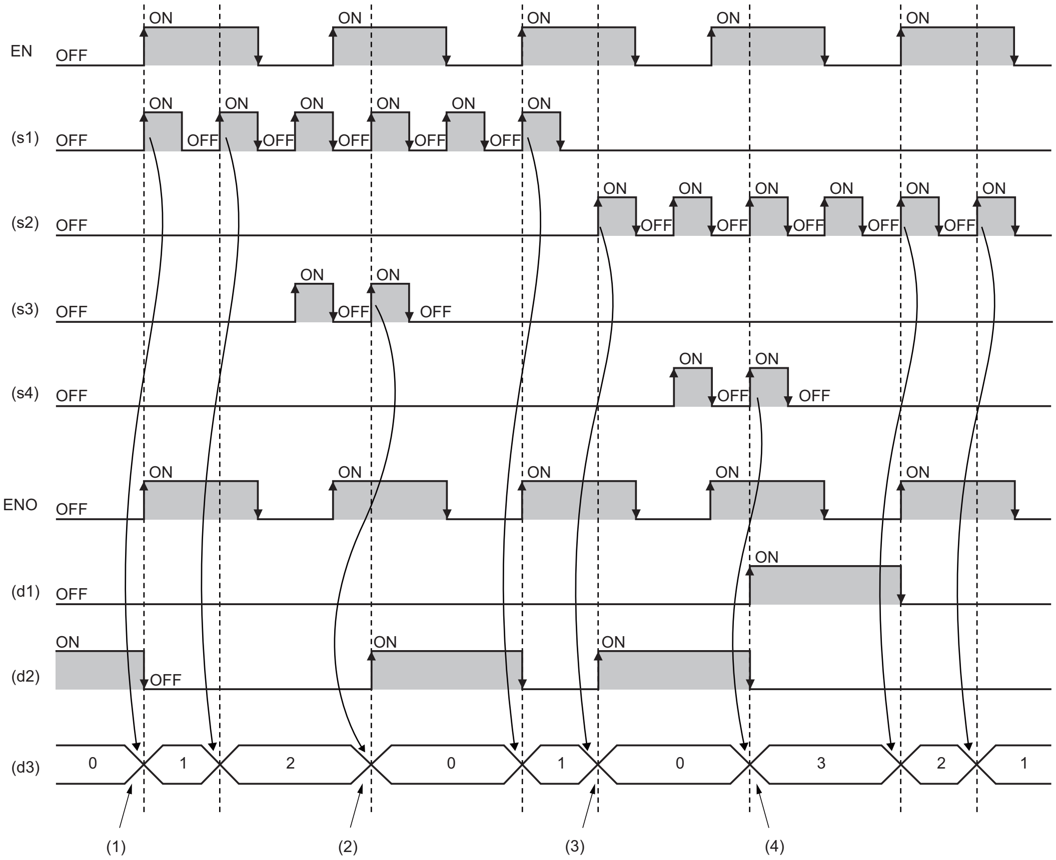
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2), (d3)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When (n)=3



(1) When (s1) turns on while EN is on, the value in (d3) is counted up.
(2) When (s3) turns on while EN is on, the value in (d3) is initialized.
(3) When (s2) turns on while EN is on, the value in (d3) is counted down.
(4) When (s4) turns on while EN is on, the value in (d3) is initialized.

Operation error

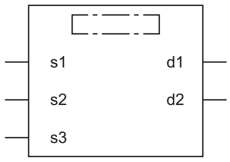
There is no operation error.

36.4 Counter Function Block

COUNTER_FB_M

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✕ RnSFCPU (Safety)

This function block starts counting up when the execution condition is satisfied.

Ladder, FBD/LD	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)	Counter setting value	Input variable	INT
s3 (ValueIn)	Initial counter value	Input variable	INT
d1 (ValueOut)	Current counter value	Output variable	ANY16
d2 (Status)	Output	Output variable	BOOL

Processing details

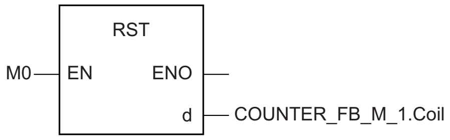
■Operation processing

- The number of rising edges (status changes (off to on)) of (s1) is counted. Counting is not performed while (s1) remains on. The counting starts from the (s3) value. When it reached the (s2) value, (d2) turns on. The current value is stored in (d1).
- 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 the current value (d1), reset (s1) of FB directly.

Ex.

Label name: COUNTER_FB_M_1

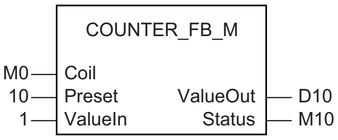
[Ladder program]



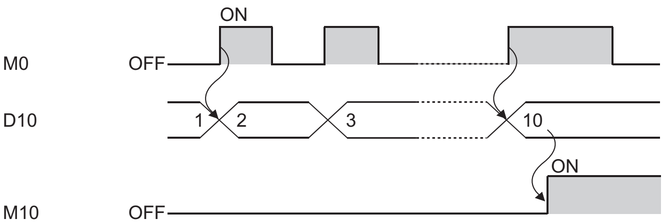
[ST program]

RST(M0, COUNTER_FB_M_1.Coil)

[Ladder example]



[Timing chart]



Operation error

There is no operation error.

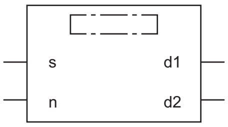
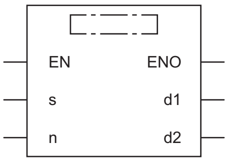
36.5 Pulse Timer

TP(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- [RnPCPU (redundant)] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks keep the signal on for the specified period of time.

Ladder, FBD/LD		Structured text
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Start of output	Input variable	BOOL
n (PT)	Output time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d1) turns on for the 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#1ms to T#2147483ms. 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 1ms, the minimum value will be 1ms.	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 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

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

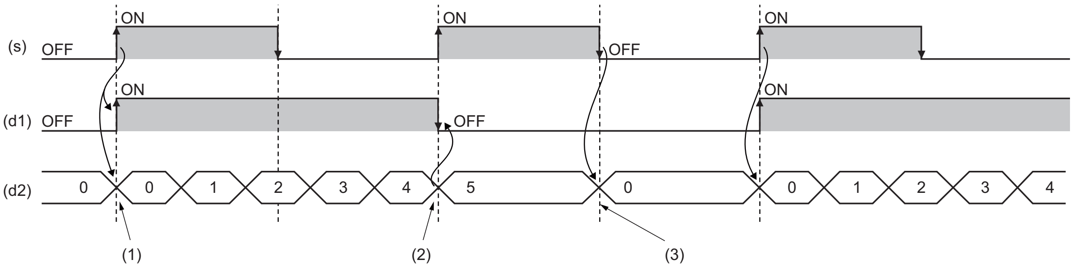
1. Function block without EN/ENO

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.

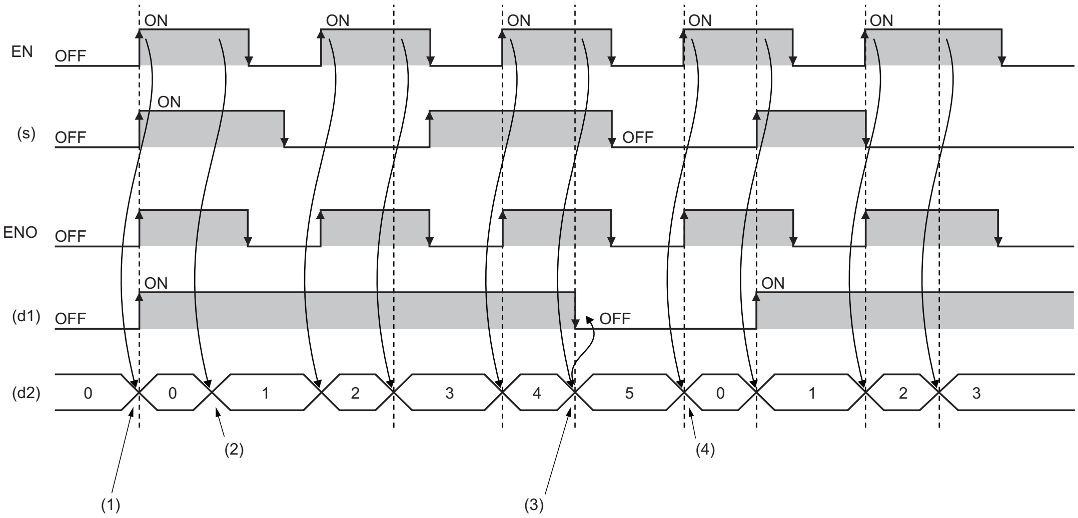
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d1), (d2)
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Undefined value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns on while EN is on, (d1) turns on. When (s) turns on while EN is on, (d2) starts measuring time.

(2) While EN is on, the time value is incremented by 1.

(3) When the time measured in (d2) reaches the time set in (n), (d1) turns off.

(4) When EN is on and both (s) and (d1) are off, the value in (d2) is initialized.

Operation error

Error code (SD0)	Description
3401H	The output time setting value exceeds the valid range.

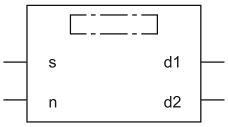
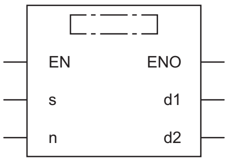
36.6 On Delay Timer

TON(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) ✗ RnSFCPU (Safety)

- [RnPCPU (redundant)] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks turn on a signal after the specified period of time.

Ladder, FBD/LD		Structured text
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d1) turns on 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#1ms to T#2147483ms. 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 1ms, the minimum value will be 1ms.	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 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

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

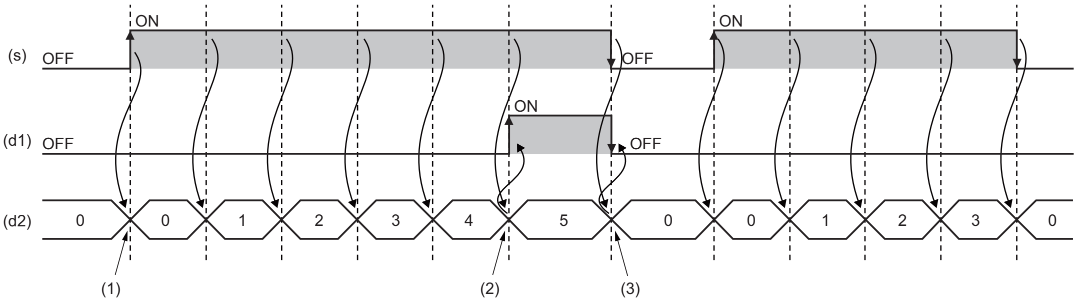
1. Function block without EN/ENO

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.

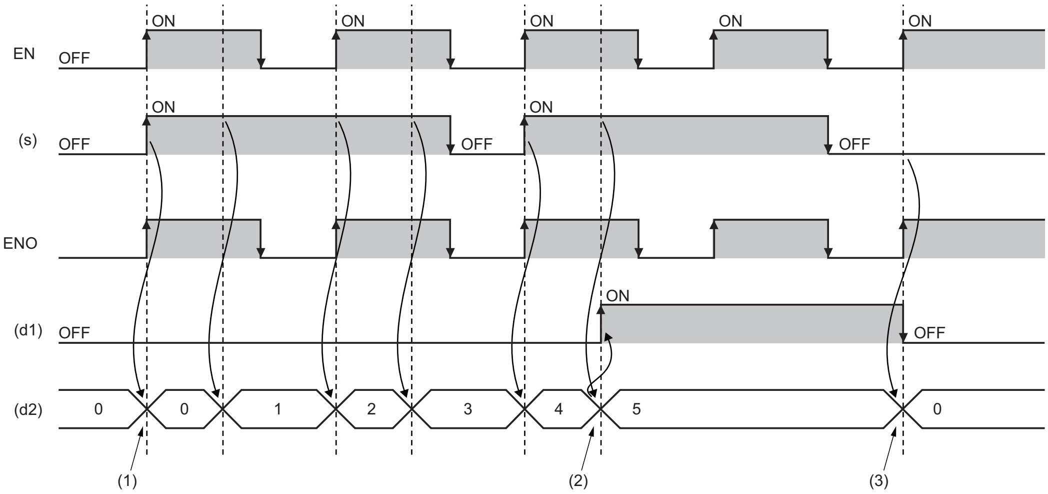
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d1), (d2)
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Previous output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns on while EN is 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 while EN is on, the value in (d2) is initialized.

Operation error

Error code (SD0)	Description
3401H	The output time setting value exceeds the valid range.

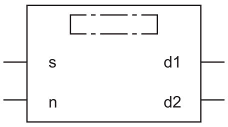
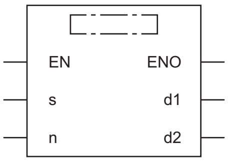
36.7 Off Delay Timer

TOF(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnSFCPU (Standard) RnSFCPU (Safety)

- [RnPCPU (redundant)] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks turn off a signal after the specified period of time.

Ladder, FBD/LD		Structured text
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d) 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#1ms to T#2147483ms. 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 1ms, the minimum value will be 1ms.	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 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

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

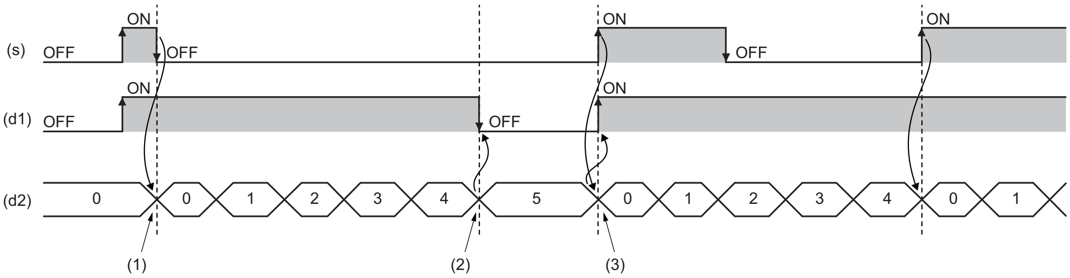
1. Function block without EN/ENO

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.

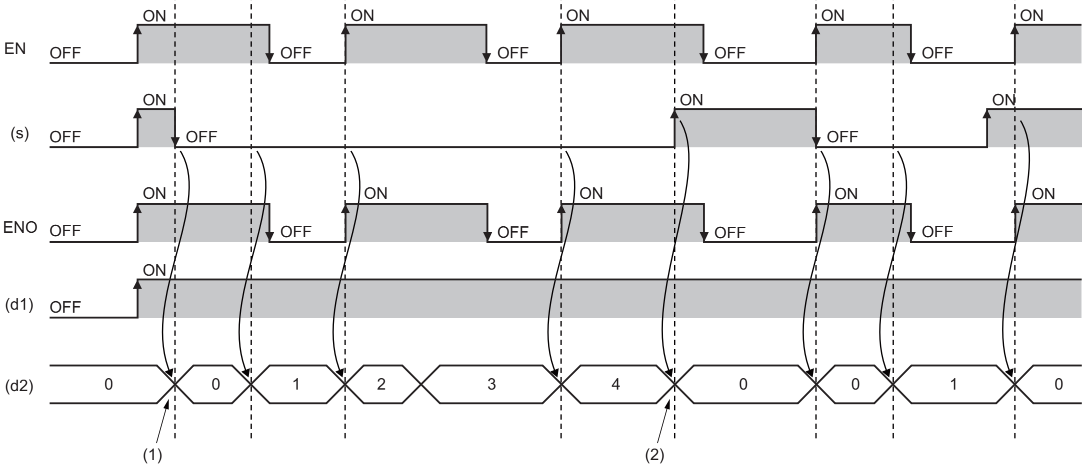
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Previous output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart

When n=T#5s (5s)



(1) When (s) turns off while EN is on, (d2) starts measuring time.

(2) When (s) turns on while EN is on, the value in (d2) is initialized.

Operation error

Error code (SD0)	Description
3401H	The output time setting value exceeds the valid range.

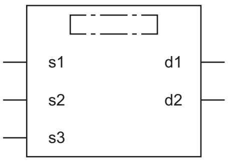
36.8 Timer Function Block

TIMER_□_M

RnCPU **RnENCPU** **RnPCPU (Process)** **RnPCPU (Redundant)** **RnSFCPU (Standard)** ~~**RnSFCPU (Safety)**~~

- [RnPCPU (redundant)] If these function blocks are used in a program executed in both systems, there are restrictions on their operation when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks start counting a timer when the execution condition is satisfied, and continue counting until the timer reaches the set value.

Ladder, FBD/LD	Structured text
 (□ is to be replaced by any of the following: TIMER_10_FB_M, TIMER_100_FB_M, TIMER_HIGH_FB_M, TIMER_LOW_FB_M, TIMER_CONT_FB_M, or TIMER_CONTHFB_M.)	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	ANY16
d2 (Status)	Output	Output variable	BOOL

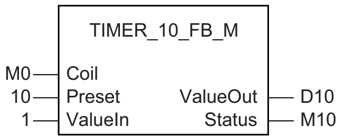
Processing details

■TIMER_10_FB_M

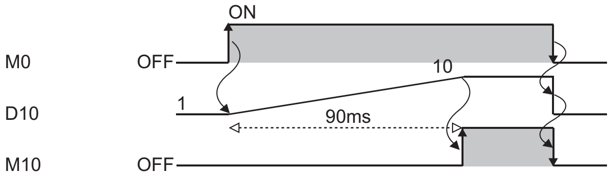
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×10ms. When the value reaches (s2)×10ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (s2) also turns off.
- If 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 compilation.
- 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.

[Ladder example]



[Timing chart]

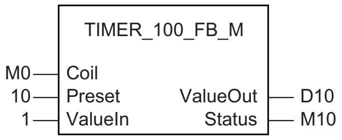


■TIMER_100_FB_M

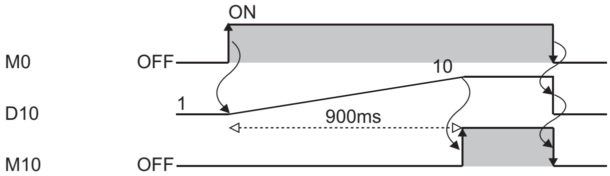
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×100ms. When the value reaches (s2)×100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (s2) also turns off.
- 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 compilation. a warning will be issued during compilation.
- 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.

[Ladder example]



[Timing chart]

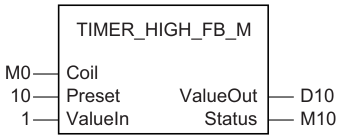


■TIMER_HIGH_FB_M

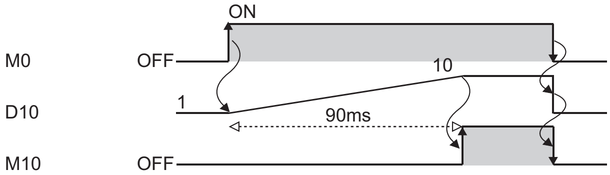
- This is a high-speed timer whose unit of measurement is 0.1 to 100ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×0.1 to 100ms (variable; set in parameter). When the value reaches (s2)×0.1 to 100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (s2) also turns off.
- The unit of measurement of the high-speed timer is 10ms by default. The unit can be changed in the range from 0.01 to 100ms.
- 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.

[Ladder example]



[Timing chart]

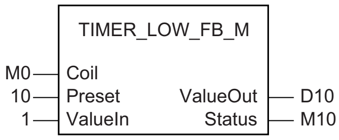


■TIMER_LOW_FB_M

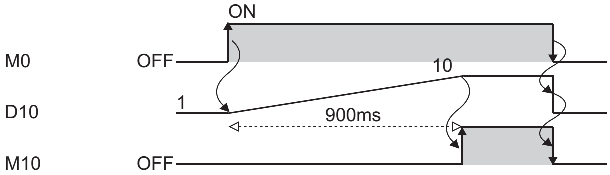
- This is a low-speed timer whose unit of measurement is 1 to 1000ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×1 to 1000ms (variable; set in parameter). When the value reaches (s2)×1 to 1000ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (s2) also turns off.
- The unit of measurement of the low-speed timer is 100ms by default. The unit can be changed in the range from 1 to 1000ms (in increments of 1ms).
- 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.

[Ladder example]



[Timing chart]



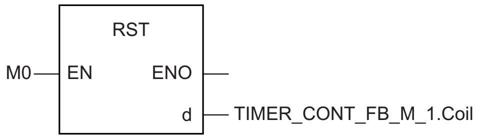
■TIMER_CONT_FB_M/TIMER_CONTHFB_M

- 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 1000ms (0.1 to 100ms for the high-speed retentive timer) (variable; set in parameter). When the value reaches (s2)×1 to 1000ms (0.1 to 100ms 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 name: TIMER_CONT_FB_M_1

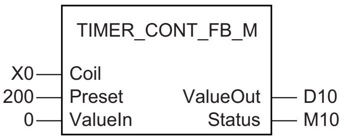
[Ladder program]



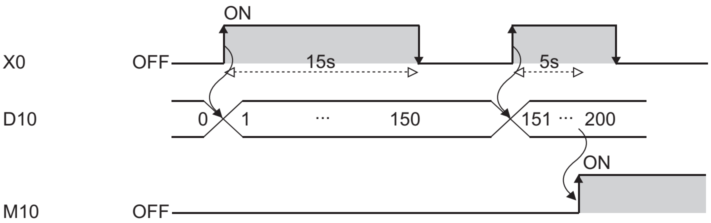
[ST program]

RST(M0,TIMER_CONT_FB_M_1.Coil)

[Ladder example]



[Timing chart]



Operation error

There is no operation error.

APPENDICES

Appendix 1 Instruction Processing Time

The following table lists the processing time of each instruction.

The processing time varies slightly depending on the contents of the source and destination. Assume that the values in the table are reference processing time.

Point

When using the file register (R/ZR), module access device (U□\G□), or link direct device (J□\□), add extra time described in the section below to each instruction processing time.

☞ Page 2030 Time added to instruction processing time

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
LD	When executed	0.00098	
LDI	When executed	0.00098	
AND	When executed	0.00098	
ANI	When executed	0.00098	
OR	When executed	0.00098	
ORI	When executed	0.00098	
LDP	When executed	0.00294	
LDF	When executed	0.00294	
ANDP	When executed	0.00294	
ANDF	When executed	0.00294	
ORP	When executed	0.00294	
ORF	When executed	0.00294	
LDPI	When executed	0.00294	
LDFI	When executed	0.00294	
ANDPI	When executed	0.00294	
ANDFI	When executed	0.00294	
ORPI	When executed	0.00294	
ORFI	When executed	0.00294	
ANB	—	0.00098	
ORB	—	0.00098	
MPS	—	0.00098	
MRD	—	0.00098	
MPP	—	0.00098	
INV	When not executed/When executed	0.00098	
MEP	When not executed/When executed	0.00098	
MEF	When not executed/When executed	0.00098	
EGP	When not executed/When executed	0.00196	
EGF	When not executed/When executed	0.00196	
OUT	When not executed/When executed	0.00196	
OUT (F)	When not executed	0.00696	
	When executed	50.900	81.600
OUT (T/ST/C)	When not executed	0.01196	
	When executed: when counting/after timeout	0.01196	
OUT (LT/LST)	When not executed	0.00996	
	When executed: when counting/after timeout	0.00996	
OUT (LC)	When not executed	0.01196	
	When executed: when counting/after timeout	0.01196	

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
SET	When not executed	0.00196	
	When executed: when the value is changed/when the value is not changed	0.00196	
SET (F)	When not executed	0.00696	
	When executed	50.300	81.700
RST	When not executed	0.00196	
	When executed: when the value is changed/when the value is not changed	0.00196	
RST (F)	When not executed	0.00196	
	When executed	15.900	29.300
RST (T/ST/C)	When not executed	0.00996	
	When executed	0.00996	
RST (LT/LST)	When not executed	0.00596	
	When executed	0.00596	
RST (LC)	When not executed	0.00996	
	When executed	0.00996	
PLS	—	0.00196	
PLF	—	0.00196	
FF	When not executed/When executed	0.00196	
DELTA	When not executed	0.00392	
	When executed	1.700	6.300
SFT	When not executed	0.00392	
	When executed	0.900	3.300
MC	—	0.00196	
MCR	—	0.00196	
FEND	—	Refer to the following. 📖MELSEC iQ-R CPU Module User's Manual (Application)	
END	—		
STOP	—	—	
NOP	—	0.00098	
LD=	Continuity/Non-continuity	0.00588	
LD<>	Continuity/Non-continuity	0.00588	
LD>	Continuity/Non-continuity	0.00588	
LD<=	Continuity/Non-continuity	0.00588	
LD<	Continuity/Non-continuity	0.00588	
LD>=	Continuity/Non-continuity	0.00588	
AND=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND>=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR<>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
OR<=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR<	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR>=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
LD=_U	Continuity/Non-continuity	0.00588	
LD<>_U	Continuity/Non-continuity	0.00588	
LD>_U	Continuity/Non-continuity	0.00588	
LD<=_U	Continuity/Non-continuity	0.00588	
LD<_U	Continuity/Non-continuity	0.00588	
LD>=_U	Continuity/Non-continuity	0.00588	
AND=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND<_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
AND>=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR<>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR<=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR<_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
OR>=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
LDD=	Continuity/Non-continuity	0.00588	
LDD<>	Continuity/Non-continuity	0.00588	
LDD>	Continuity/Non-continuity	0.00588	
LDD<=	Continuity/Non-continuity	0.00588	
LDD<	Continuity/Non-continuity	0.00588	
LDD>=	Continuity/Non-continuity	0.00588	
ANDD=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
ANDD>=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD>	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD>=	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
LDD=_U	Continuity/Non-continuity	0.00588	
LDD<>_U	Continuity/Non-continuity	0.00588	
LDD>_U	Continuity/Non-continuity	0.00588	
LDD<=_U	Continuity/Non-continuity	0.00588	
LDD<_U	Continuity/Non-continuity	0.00588	
LDD>=_U	Continuity/Non-continuity	0.00588	
ANDD=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD<_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ANDD>=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD>_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD<_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
ORD>=_U	When not executed	0.00392	
	When executed: continuity/non-continuity	0.00588	
CMP	—	2.900	5.400
CMP_U	—	2.900	5.400
DCMP	—	2.900	5.400
DCMP_U	—	2.900	5.400
ZCP	—	3.300	6.200
ZCP_U	—	3.300	6.200
DZCP	—	3.300	6.200
DZCP_U	—	3.300	6.200

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
BKCMP=	(n)=1	3.600	10.500
	(n)=96	17.000	23.400
BKCMP<>	(n)=1	3.700	9.900
	(n)=96	17.400	23.600
BKCMP>	(n)=1	3.600	10.500
	(n)=96	16.800	24.600
BKCMP<=	(n)=1	3.500	10.300
	(n)=96	17.500	23.300
BKCMP<	(n)=1	3.600	10.300
	(n)=96	17.500	24.500
BKCMP>=	(n)=1	3.600	10.000
	(n)=96	17.700	24.100
BKCMP=_U	(n)=1	3.600	10.300
	(n)=96	16.900	23.900
BKCMP<>_U	(n)=1	3.700	9.800
	(n)=96	17.300	23.600
BKCMP>_U	(n)=1	3.500	10.200
	(n)=96	16.900	23.600
BKCMP<=_U	(n)=1	3.600	10.100
	(n)=96	16.700	23.300
BKCMP<_U	(n)=1	3.700	10.100
	(n)=96	16.800	24.100
BKCMP>=_U	(n)=1	3.600	9.900
	(n)=96	17.000	25.400
DBKCMP=	(n)=1	3.700	10.400
	(n)=96	17.000	23.500
DBKCMP<>	(n)=1	3.700	10.000
	(n)=96	17.400	23.700
DBKCMP>	(n)=1	3.600	10.100
	(n)=96	17.400	24.100
DBKCMP<=	(n)=1	3.800	10.100
	(n)=96	17.100	23.600
DBKCMP<	(n)=1	3.700	9.900
	(n)=96	17.300	23.800
DBKCMP>=	(n)=1	3.600	10.200
	(n)=96	17.100	22.900
DBKCMP=_U	(n)=1	3.700	10.200
	(n)=96	17.000	23.600
DBKCMP<>_U	(n)=1	3.700	10.000
	(n)=96	17.400	23.600
DBKCMP>_U	(n)=1	3.700	10.200
	(n)=96	24.300	30.800
DBKCMP<=_U	(n)=1	3.700	10.200
	(n)=96	23.900	30.300
DBKCMP<_U	(n)=1	3.800	10.000
	(n)=96	24.300	31.000
DBKCMP>=_U	(n)=1	3.800	10.500
	(n)=96	24.400	31.000
+ (s) (d)	When executed	0.00588	
+ (s1) (s2) (d)	When executed	0.00588	
+_U (s) (d)	When executed	0.00588	
+_U (s1) (s2) (d)	When executed	0.00588	

Instruction name	Condition	Processing time (μs)	
		Minimum	**Maximum**
- (s) (d)	When executed	0.00588	
- (s1) (s2) (d)	When executed	0.00588	
-_U (s) (d)	When executed	0.00588	
-_U (s1) (s2) (d)	When executed	0.00588	
D+ (s) (d)	When executed	0.00588	
D+ (s1) (s2) (d)	When executed	0.00588	
D+_U (s) (d)	When executed	0.00588	
D+_U (s1) (s2) (d)	When executed	0.00588	
D- (s) (d)	When executed	0.00588	
D- (s1) (s2) (d)	When executed	0.00588	
D-_U (s) (d)	When executed	0.00588	
D-_U (s1) (s2) (d)	When executed	0.00588	
*	When executed	0.01176	
*_U	When executed	0.01176	
/	When executed	0.01960	
/_U	When executed	0.01960	
D*	When executed	0.01960	
D*_U	When executed	0.01960	
D/	When executed	0.02940	
D/_U	When executed	0.02940	
B+ (s) (d)	When executed	1.300	3.700
B+ (s1) (s2) (d)	When executed	1.800	5.100
B- (s) (d)	When executed	1.300	3.800
B- (s1) (s2) (d)	When executed	1.900	5.000
DB+ (s) (d)	When executed	2.000	6.200
DB+ (s1) (s2) (d)	When executed	2.100	6.100
DB- (s) (d)	When executed	2.000	6.100
DB- (s1) (s2) (d)	When executed	2.100	5.900
B*	When executed	1.400	4.900
B/	When executed	1.500	4.800
DB*	When executed	2.400	8.000
DB/	When executed	2.200	7.900
BK+	(n)=1	3.500	8.100
	(n)=96	17.200	21.500
BK+_U	(n)=1	3.200	8.300
	(n)=96	17.500	21.600
BK-	(n)=1	3.100	8.200
	(n)=96	17.400	21.400
BK-_U	(n)=1	3.200	8.300
	(n)=96	17.500	21.500
DBK+	(n)=1	3.300	6.900
	(n)=96	17.400	21.900
DBK+_U	(n)=1	3.000	7.800
	(n)=96	17.400	21.900
DBK-	(n)=1	3.000	7.900
	(n)=96	17.400	22.000
DBK-_U	(n)=1	3.000	7.900
	(n)=96	17.400	22.000
INC	When executed	0.00392	
INC_U	When executed	0.00392	
DEC	When executed	0.00392	
DEC_U	When executed	0.00392	

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
DINC	When executed	0.00392	
DINC_U	When executed	0.00392	
DDEC	When executed	0.00392	
DDEC_U	When executed	0.00392	
WAND (s) (d)	When executed	0.00392	
WAND (s1) (s2) (d)	When executed	0.00392	
DAND (s) (d)	When executed	0.00392	
DAND (s1) (s2) (d)	When executed	0.00392	
BKAND	(n)=1	3.300	8.400
	(n)=96	17.500	22.400
WOR (s) (d)	When executed	0.00392	
WOR (s1) (s2) (d)	When executed	0.00392	
DOR (s) (d)	When executed	0.00392	
DOR (s1) (s2) (d)	When executed	0.00392	
BKOR	(n)=1	3.300	8.300
	(n)=96	17.500	22.800
WXOR (s) (d)	When executed	0.00392	
WXOR (s1) (s2) (d)	When executed	0.00392	
DXOR (s) (d)	When executed	0.00392	
DXOR (s1) (s2) (d)	When executed	0.00392	
BKXOR	(n)=1	3.300	8.300
	(n)=96	17.500	22.400
WXNR (s) (d)	When executed	0.00392	
WXNR (s1) (s2) (d)	When executed	0.00392	
DXNR (s) (d)	When executed	0.00392	
DXNR (s1) (s2) (d)	When executed	0.00392	
BKXNR	(n)=1	3.300	8.200
	(n)=96	17.900	22.800
BSET	(n)=1	0.00294	
	(n)=15	0.00294	
BRST	(n)=1	0.00294	
	(n)=15	0.00294	
TEST	When executed	0.0120	
DTEST	When executed	0.0190	
BKRST	(n)=1	0.900	2.000
	(n)=96	1.200	2.200
SFR	(n)=1	0.00882	
	(n)=15	0.00882	
SFL	(n)=1	0.00882	
	(n)=15	0.00882	
BSFR	(n)=1	0.800	1.800
	(n)=96	1.500	2.600
BSFL	(n)=1	0.800	1.900
	(n)=96	1.500	2.600
DSFR	(n)=1	1.100	3.300
	(n)=96	8.200	10.800
DSFL	(n)=1	1.100	3.400
	(n)=96	8.200	10.700
SFTBR	(n1)=16, (n2)=1	0.900	2.100
	(n1)=16, (n2)=15	0.900	2.100
SFTR	(n1)=16, (n2)=1	4.400	11.300
	(n1)=16, (n2)=15	4.400	11.300

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
SFTBL	(n1)=16, (n2)=1	0.900	2.100
	(n1)=16, (n2)=15	0.900	2.100
SFTL	(n1)=16, (n2)=1	4.400	11.300
	(n1)=16, (n2)=15	4.400	11.300
SFTWR	(n1)=16, (n2)=1	2.400	5.400
	(n1)=16, (n2)=15	2.400	5.700
WSFR	(n1)=16, (n2)=1	5.100	9.600
	(n1)=16, (n2)=15	5.100	9.600
SFTWL	(n1)=16, (n2)=1	2.400	5.800
	(n1)=16, (n2)=15	2.400	5.400
WSFL	(n1)=16, (n2)=1	5.100	9.600
	(n1)=16, (n2)=15	5.100	9.600
BCD	When executed	0.01078	
DBCD	When executed	0.01862	
BIN	When executed	0.00686	
DBIN	When executed	0.00686	
FLT2INT	(s)=0	0.00686	
	(s)=32766.5	0.00686	
FLT2UINT	(s)=0	0.00686	
	(s)=65534.5	0.00686	
FLT2DINT	(s)=0	0.00686	
	(s)=1234567890.3	0.00686	
FLT2UDINT	(s)=0	0.00686	
	(s)=1234567890.3	0.00686	
DBL2INT	(s)=0	1.400	3.400
	(s)=32766.5	1.500	4.100
DBL2UINT	(s)=0	1.300	3.600
	(s)=65534.5	1.500	4.000
DBL2DINT	(s)=0	1.400	3.400
	(s)=1234567890.3	1.400	4.200
DBL2UDINT	(s)=0	1.500	3.400
	(s)=1234567890.3	1.500	4.100
INT2UINT	When executed	0.00294	
INT2DINT	When executed	0.00294	
INT2UDINT	When executed	0.00294	
UINT2INT	When executed	0.00294	
UINT2DINT	When executed	0.00294	
UINT2UDINT	When executed	0.00294	
DINT2INT	When executed	0.00294	
DINT2UINT	When executed	0.00294	
DINT2UDINT	When executed	0.00294	
UDINT2INT	When executed	0.00294	
UDINT2UINT	When executed	0.00294	
UDINT2DINT	When executed	0.00294	
GRY	When executed	0.00490	
GRY_U	When executed	0.00490	
DGRY	When executed	0.00490	
DGRY_U	When executed	0.00490	
GBIN	When executed	0.00490	
GBIN_U	When executed	0.00490	
DGBIN	When executed	0.00490	
DGBIN_U	When executed	0.00490	

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
BKBCD	(n)=1	2.400	7.800
	(n)=96	20.700	25.700
BKBIN	(n)=1	2.300	7.400
	(n)=96	17.100	21.900
DABIN	(s)=1	3.000	10.800
	(s)=-32768	3.000	10.800
DABIN_U	(s)=1	3.000	10.800
	(s)=65535	3.000	10.800
DDABIN	(s)=1	3.400	11.100
	(s)=-2147483648	3.400	11.100
DDABIN_U	(s)=1	3.400	11.100
	(s)=4294967295	3.400	11.100
HABIN	(s)=1	2.900	8.700
	(s)=FFFFH	2.900	8.700
DHABIN	(s)=1	3.100	9.000
	(s)=FFFFFFFFH	3.100	9.000
DABCD	(s)=1	2.800	8.900
	(s)=9999	2.800	8.900
DDABCD	(s)=1	3.000	9.100
	(s)=99999999	3.000	9.100
VAL	—	3.800	11.900
VAL_U	—	4.100	11.900
DVAL	—	4.900	13.100
DVAL_U	—	5.200	12.600
ASC2INT	(n)=1	2.600	7.400
	(n)=96	8.300	13.700
EMOD	—	3.000	8.200
NEG	When executed	0.00490	
DNEG	When executed	0.00490	
DECO	(n)=2	2.400	5.900
	(n)=8	2.400	6.300
ENCO	(n)=2, M1=ON	2.500	6.400
	(n)=2, M4=ON	2.500	6.400
	(n)=8, M1=ON	3.500	7.800
	(n)=8, M256=ON	2.400	6.200
SEG	When executed	0.400	1.600
DIS	(n)=1	2.200	4.200
	(n)=4	2.200	4.300
UNI	(n)=1	2.400	5.300
	(n)=4	2.500	5.200
NDIS	When executed	2.200	5.000
NUNI	When executed	2.200	5.000
WTOB	(n)=1	2.500	5.500
	(n)=96	13.000	16.000
BTOW	(n)=1	2.600	5.500
	(n)=96	9.800	12.600
MOV	—	0.00196	
DMOV	—	0.00196	
CML	—	0.00196	
DCML	—	0.00196	
SMOV	SM773=OFF	3.500	8.500
	SM773=ON	3.200	6.100

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
CMLB	—	0.00196	
BMOV	(n)=1	1.100	1.300
	(n)=96	2.000	2.200
BMOVL	(n)=1	1.300	2.500
	(n)=96	2.200	3.600
FMOV	(n)=1	0.700	1.000
	(n)=96	1.800	2.000
FMOVL	(n)=1	1.000	2.000
	(n)=96	2.000	3.300
DFMOV	(n)=1	1.000	2.000
	(n)=96	2.200	3.400
DFMOVL	(n)=1	1.000	2.000
	(n)=96	2.200	3.500
XCH	—	0.900	1.500
DXCH	—	0.900	1.500
BXCH	(n)=1	2.700	6.800
	(n)=96	16.900	20.700
SWAP	—	1.200	2.200
DSWAP	—	1.200	2.200
MOVB	—	0.00196	
BLKMOVB	(n)=1	3.100	7.600
	(n)=96	3.600	8.600
ROR	(n)=1	0.00882	
	(n)=15	0.00882	
RCR	(n)=1	0.00882	
	(n)=15	0.00882	
DROR	(n)=1	0.00882	
	(n)=31	0.00882	
DRCR	(n)=1	0.00882	
	(n)=31	0.00882	
ROL	(n)=1	0.00882	
	(n)=15	0.00882	
RCL	(n)=1	0.00882	
	(n)=15	0.00882	
DROL	(n)=1	0.00882	
	(n)=31	0.00882	
DRCL	(n)=1	0.00882	
	(n)=31	0.00882	
CJ	—	1.000	2.000
SCJ	—	1.000	2.000
JMP	—	1.000	2.100
GOEND	—	0.9000	
DI	—	2.800	4.200
DI (s)	—	3.000	4.200
EI	—	4.100	9.200
IMASK	—	1.000	1.800
SIMASK	—	0.800	1.700
IRET	—	1.700	2.200
WDT	—	4.900	16.400
FOR	—	0.00196	
NEXT	—	0.03920	
BREAK	—	4.400	10.000

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
CALL Pn	Local pointer	0.800	1.500
	Global pointer	3.900	13.500
CALL Pn (s1) to (s5)	Local pointer	17.400	40.600
RET	Return to the own program	0.2000	
	Return to the another program	2.0000	
FCALL Pn	Local pointer	1.000	2.000
	Global pointer	4.300	22.900
FCALL Pn (s1) to (s5)	Local pointer	16.400	37.800
ECALL Pn	Local pointer, file name="P1"	74.700	118.400
ECALL Pn (s1) to (s5)	Local pointer, file name="P1"	94.500	148.400
EFCALL Pn	Local pointer, file name="P1"	72.400	114.300
EFCALL Pn (s1) to (s5)	Local pointer, file name="P1"	91.100	132.200
XCALL Pn	—	3.800	16.700
FIFR	Number of data stored = 1	2.100	4.800
	Number of data stored = 96	7.100	10.300
FPOP	Number of data stored = 1	2.000	4.600
	Number of data stored = 96	2.100	4.600
FIFW	Number of data stored = 0	2.100	4.800
	Number of data stored = 96	2.100	4.700
FINS	Number of data stored = 0	2.500	5.800
	Number of data stored = 96	9.200	13.700
FDEL	Number of data stored = 1	2.300	6.000
	Number of data stored = 96	7.400	11.500
S.DEVLD	—	4.300	7.300
SP.DEVST	—	20.200	24.800
SP.FREAD	—	36.800	42.800
SP.FWRITE	—	41.500	47.400
LEDR	No self-diagnostic error	2.100	5.200
	Self-diagnostic error (continuation error, annunciator ON)	15.000	25.900
PALERT(P)	—	77.100	146.000
PABORT	—	—	—
LD$=	Continuity/Non-continuity	1.600	4.500
LD$<>	Continuity/Non-continuity	1.600	4.500
LD$>	Continuity/Non-continuity	1.600	4.500
LD$<=	Continuity/Non-continuity	1.600	4.500
LD$<	Continuity/Non-continuity	1.600	4.500
LD$>=	Continuity/Non-continuity	1.600	4.500
AND$=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
AND$<>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
AND$>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
AND$<=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
AND$<	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
AND$>=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
OR$=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
OR$<>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
OR$>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
OR$<=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
OR$<	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
OR$>=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.600	4.500
$+ (s) (d)	When executed	2.400	6.700
$+ (s1) (s2) (d)	When executed	2.900	8.200
$MOV	0 characters	2.400	9.200
	32 characters	4.400	12.100
$MOV_WS	0 characters	2.400	10.100
	32 characters	6.100	14.200
BINDA	(s)=1	2.600	5.900
	(s)=-32768	3.000	6.800
BINDA_U	(s)=1	2.600	5.900
	(s)=65535	3.000	6.800
DBINDA	(s)=1	2.700	6.100
	(s)=-2147483648	3.100	7.400
DBINDA_U	(s)=1	2.700	6.100
	(s)=4294967295	3.100	7.400
BINHA	(s)=1	2.600	5.500
	(s)=FFFFH	2.600	5.600
DBINHA	(s)=1	2.600	5.300
	(s)=FFFFFFFFH	2.600	5.700
STR	—	2.800	7.300
STR_U	—	2.900	7.300
DSTR	—	3.200	7.400
DSTR_U	—	3.300	7.900
BCDDA	(s)=1	2.600	6.200
	(s)=9999	2.700	6.300
DBCDDA	(s)=1	2.600	6.400
	(s)=99999999	2.700	6.700
ESTR	—	4.7000	17.1000
INT2ASC	(n)=1	2.600	6.900
	(n)=96	6.100	11.000
WS2SJIS	Number of characters = 1	3.300	10.300
	Number of characters = 96	52.500	59.200
SJIS2WS	Number of characters = 1	3.100	9.900
	Number of characters = 96	48.300	55.200
SJIS2WSB	Number of characters = 1	3.200	9.800
	Number of characters = 96	48.300	55.100
LEN	1 characters	1.400	3.800
	96 characters	9.600	11.900
RIGHT	Number of characters to be extracted = 1	3.200	11.200
	Number of characters to be extracted = 96	15.800	23.400
LEFT	Number of characters to be extracted = 1	3.200	11.100
	Number of characters to be extracted = 96	15.700	23.200
MIDR	—	3.600	12.200

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
MIDW	—	4.200	12.300
INSTR	No match	7.500	15.000
	Match: Start	5.300	13.000
	Match: Last	7.500	15.300
STRINS	(s)=128, (d)=40, (n)=1	17.900	27.200
	(s)=128, (d)=40, (n)=48	20.400	30.100
STRDEL	(s)=128, (d)=40, (n)=1	15.800	23.000
	(s)=128, (d)=40, (n)=48	13.600	20.900
LDE=	Continuity/Non-continuity	0.00588	
LDE<>	Continuity/Non-continuity	0.00588	
LDE>	Continuity/Non-continuity	0.00588	
LDE<=	Continuity/Non-continuity	0.00588	
LDE<	Continuity/Non-continuity	0.00588	
LDE>=	Continuity/Non-continuity	0.00588	
ANDE=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ANDE<>	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ANDE>	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ANDE<=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ANDE<	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ANDE>=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE<>	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE>	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE<=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE<	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
ORE>=	When not executed	0.00588	
	When executed: continuity/non-continuity	0.00588	
LDED=	Continuity/Non-continuity	1.900	4.500
LDED<>	Continuity/Non-continuity	1.900	4.500
LDED>	Continuity/Non-continuity	1.800	4.400
LDED<=	Continuity/Non-continuity	1.900	4.500
LDED<	Continuity/Non-continuity	1.900	4.500
LDED>=	Continuity/Non-continuity	1.900	4.500
ANDED=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ANDED<>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ANDED>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ANDED<=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
ANDED<	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ANDED>=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED<>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED>	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED<=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED<	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ORED>=	When not executed	0.00588	
	When executed: continuity/non-continuity	1.900	4.500
ECMP	—	2.900	5.900
EDCMP	—	3.100	7.400
EZCP	—	3.300	6.600
EDZCP	—	3.700	8.600
E+ (s) (d)	(s)=0, (d)=0	0.0098	
	(s)=2^{127}, (d)=2^{127}	0.0098	
E+ (s1) (s2) (d)	(s1)=0, (s2)=0	0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.0098	
E- (s) (d)	(s)=0, (d)=0	0.0098	
	(s)=2^{127}, (d)=2^{127}	0.0098	
E- (s1) (s2) (d)	(s1)=0, (s2)=0	0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.0098	
ED+ (s) (d)	(s)=0, (d)=0	1.600	5.800
	(s)=2^{1023}, (d)=2^{1023}	2.000	7.000
ED+ (s1) (s2) (d)	(s1)=0, (s2)=0	2.000	5.700
	(s1)=2^{1023}, (s2)=2^{1023}	2.200	7.000
ED- (s) (d)	(s)=0, (d)=0	1.800	5.500
	(s)=2^{1023}, (d)=2^{1023}	1.900	5.700
ED- (s1) (s2) (d)	(s1)=0, (s2)=0	2.000	5.500
	(s1)=2^{1023}, (s2)=2^{1023}	2.100	5.900
E*	(s1)=0, (s2)=0	0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.0098	
E/	(s1)=2^{127}, (s2)=2^{127}	0.5684	
ED*	(s1)=0, (s2)=0	2.000	5.900
	(s1)=2^{1023}, (s2)=2^{1023}	2.300	7.300
ED/	(s1)=2^{1023}, (s2)=2^{1023}	2.300	7.300
INT2FLT	(s)=0	0.00686	
	(s)=7FFFH	0.00686	
UINT2FLT	(s)=0	0.00686	
	(s)=FFFFH	0.00686	
DINT2FLT	(s)=0	0.00686	
	(s)=7FFFFFFFH	0.00686	
UDINT2FLT	(s)=0	0.00686	
	(s)=FFFFFFFFH	0.00686	
DBL2FLT	—	1.600	4.000

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
INT2DBL	(s)=0	1.400	3.100
	(s)=7FFFH	1.400	3.100
UINT2DBL	(s)=0	1.400	3.300
	(s)=FFFFH	1.400	3.200
DINT2DBL	(s)=0	1.400	3.100
	(s)=7FFFFFFFH	1.400	3.100
UDINT2DBL	(s)=0	1.400	3.200
	(s)=FFFFFFFFH	1.400	3.200
FLT2DBL	—	1.400	4.600
EVAL	Decimal point format 2-digit full specification	3.700	12.300
	Number of digits format 6-digit full specification	4.100	12.000
EREXP	—	3.100	9.400
ENEG	(d)=0	1.200	2.200
	(d)=-1.0	1.300	2.900
EDNEG	(d)=0	1.300	4.300
	(d)=-1.0	1.100	4.400
EMOV	—	0.00196	
EDMOV	—	0.00196	
SIN	—	1.400	4.300
COS	—	1.400	4.200
TAN	—	1.400	4.200
ASIN	—	1.400	4.500
ACOS	—	1.400	4.500
ATAN	—	1.400	3.400
SIND	—	2.400	12.700
COSD	—	2.300	12.700
TAND	—	2.800	13.900
ASIND	—	2.400	10.500
ACOSD	—	2.200	9.700
ATAND	—	2.000	9.200
BSIN	—	2.600	9.300
BCOS	—	2.600	8.900
BTAN	—	2.700	9.600
BASIN	—	2.400	7.300
BACOS	—	2.500	7.400
BATAN	—	2.400	7.400
RAD	—	1.400	2.700
DEG	—	1.400	2.900
RADD	—	1.600	7.300
DEGD	—	1.600	7.100
ESQRT	—	1.100	2.900
EDSQRT	—	1.700	7.200
EXP	(s)=-10	1.400	4.500
	(s)=1	1.500	4.600
EXPD	(s)=-10	2.200	10.500
	(s)=1	2.100	10.300
LOG	(s)=1	1.300	4.100
	(s)=10	1.400	4.700
LOGD	(s)=1	1.800	8.600
	(s)=10	2.300	10.900
BSQRT	(s)=0	1.500	3.100
	(s)=9999	2.200	6.900

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
BDSQRT	(s)=0	1.400	2.600
	(s)=99999999	2.100	6.000
POW	(s1)=1.23E+5, (s2)=3.45E+0	3.000	8.900
POWD	(s1)=1.23E+5, (s2)=3.45E+0	4.400	19.300
LOG10	(s)=1.23E+20	1.500	4.700
LOG10D	(s)=1.23E+20	2.400	12.900
EMAX	(n)=1	2.500	5.100
	(n)=96	14.700	17.900
EDMAX	(n)=1	2.700	6.900
	(n)=96	27.700	32.500
EMIN	(n)=1	2.500	5.100
	(n)=96	14.700	17.900
EDMIN	(n)=1	2.700	6.800
	(n)=96	27.200	32.400
RND	—	0.800	1.700
SRND	—	1.100	1.800
ZPUSH (d)	—	0.900	1.700
ZPUSH (s) (d)	All areas of Z and LZ	1.600	3.200
	All area of Z (default: 20 points)	1.600	3.100
	All area of LZ (default: 2 points)	1.300	3.100
ZPOP (d)	—	0.900	1.800
ZPOP (s) (d)	All areas of Z and LZ	1.500	3.200
	All area of Z (default: 20 points)	1.500	3.100
	All area of LZ (default: 2 points)	1.300	3.100
LIMIT	—	1.000	1.800
LIMIT_U	—	1.000	1.800
DLIMIT	—	1.000	1.800
DLIMIT_U	—	1.000	1.800
BAND	—	1.800	3.100
BAND_U	—	1.800	3.100
DBAND	—	1.900	3.000
DBAND_U	—	1.900	2.900
ZONE	—	1.800	3.000
ZONE_U	—	1.800	3.000
DZONE	—	1.900	2.900
DZONE_U	—	1.900	3.000
SCL	SM755=ON, point No.1<(s1)<point No.2	3.300	8.900
	SM755=ON, point No.9<(s1)<point No.10	3.300	8.900
	SM755=OFF, point No.1<(s1)<point No.2	3.100	8.700
	SM755=OFF, point No.9<(s1)<point No.10	3.700	8.900
SCL_U	SM755=ON, point No.1<(s1)<point No.2	3.200	8.700
	SM755=ON, point No.9<(s1)<point No.10	3.200	8.800
	SM755=OFF, point No.1<(s1)<point No.2	3.000	8.500
	SM755=OFF, point No.9<(s1)<point No.10	3.400	8.900
DSCL	SM755=ON, point No.1<(s1)<point No.2	3.300	9.900
	SM755=ON, point No.9<(s1)<point No.10	3.300	9.800
	SM755=OFF, point No.1<(s1)<point No.2	3.000	9.300
	SM755=OFF, point No.9<(s1)<point No.10	3.600	9.700
DSCL_U	SM755=ON, point No.1<(s1)<point No.2	3.200	10.200
	SM755=ON, point No.9<(s1)<point No.10	3.200	10.200
	SM755=OFF, point No.1<(s1)<point No.2	2.900	9.500
	SM755=OFF, point No.9<(s1)<point No.10	3.500	9.900

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
SCL2	SM755=ON, point No.1<(s1)<point No.2	3.300	9.200
	SM755=ON, point No.9<(s1)<point No.10	3.300	9.100
	SM755=OFF, point No.1<(s1)<point No.2	3.300	8.700
	SM755=OFF, point No.9<(s1)<point No.10	3.500	8.800
SCL2_U	SM755=ON, point No.1<(s1)<point No.2	3.200	8.800
	SM755=ON, point No.9<(s1)<point No.10	3.300	8.900
	SM755=OFF, point No.1<(s1)<point No.2	2.900	8.500
	SM755=OFF, point No.9<(s1)<point No.10	3.500	9.000
DSCL2	SM755=ON, point No.1<(s1)<point No.2	3.300	10.000
	SM755=ON, point No.9<(s1)<point No.10	3.300	10.000
	SM755=OFF, point No.1<(s1)<point No.2	3.000	9.500
	SM755=OFF, point No.9<(s1)<point No.10	3.300	10.100
DSCL2_U	SM755=ON, point No.1<(s1)<point No.2	3.200	10.100
	SM755=ON, point No.9<(s1)<point No.10	3.300	10.100
	SM755=OFF, point No.1<(s1)<point No.2	2.900	9.600
	SM755=OFF, point No.9<(s1)<point No.10	3.500	10.000
UDCNT1	—	0.800	1.600
UDCNT2	—	0.800	1.600
TTMR	—	2.000	5.900
STMR	—	2.700	8.100
ROTC	—	4.800	8.600
RAMPQ	—	3.400	8.200
SPD	—	0.700	1.600
PLSY	—	0.800	1.400
PWM	—	0.800	1.400
MTR	—	3.400	11.600
CCD	SM772=OFF, (n)=1	3.300	7.100
	SM772=OFF, (n)=96	11.700	15.500
	SM772=ON, (n)=1	3.300	7.100
	SM772=ON, (n)=96	11.700	15.500
SERDATA	(n)=1: All match	2.700	6.700
	(n)=1: All mismatch	2.700	6.600
	(n)=96: All match	9.700	15.000
	(n)=96: All mismatch	9.700	15.100
DSERDATA	(n)=1: All match	3.000	7.300
	(n)=1: All mismatch	3.000	7.300
	(n)=96: All match	14.500	18.900
	(n)=96: All mismatch	14.500	18.900
SERMM	(n)=1, no matched data	4.100	9.300
	(n)=1, number of the matched data = 1	4.100	9.300
	(n)=96, no matched data	13.100	18.200
	(n)=96, number of the matched data = 96	13.100	18.200
DSERMM	(n)=1, no matched data	4.100	9.300
	(n)=1, number of the matched data = 1	4.100	9.300
	(n)=96, no matched data	13.100	18.200
	(n)=96, number of the matched data = 96	13.100	18.200
SUM	(s)=0	1.300	1.900
	(s)=FFFFH	1.300	1.900
DSUM	(s)=0	1.700	2.400
	(s)=FFFFFFFFH	1.700	2.400
BON	(s)=0	2.400	4.900
	(s)=FFFFH	2.400	4.900

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
DBON	(s)=0	2.400	4.900
	(s)=FFFFFFFFH	2.400	4.900
MAX	(n)=1	2.400	5.000
	(n)=96	9.500	12.300
MAX_U	(n)=1	2.500	6.100
	(n)=96	10.700	14.700
DMAX	(n)=1	2.500	5.300
	(n)=96	17.400	20.400
DMAX_U	(n)=1	2.500	5.800
	(n)=96	10.300	14.100
MIN	(n)=1	2.400	5.000
	(n)=96	9.500	12.400
MIN_U	(n)=1	2.500	5.900
	(n)=96	10.700	14.400
DMIN	(n)=1	2.500	5.400
	(n)=96	17.500	20.500
DMIN_U	(n)=1	2.400	5.900
	(n)=96	10.300	14.000
SORTD	(n)=1, (s1)=1	3.100	5.600
	(n)=96, (s1)=16	8.400	11.900
SORTD_U	(n)=1, (s1)=1	3.100	5.600
	(n)=96, (s1)=16	8.800	12.900
DSORTD	(n)=1, (s1)=1	3.000	5.500
	(n)=96, (s1)=16	9.900	13.500
DSORTD_U	(n)=1, (s1)=1	3.100	5.800
	(n)=96, (s1)=16	11.100	15.800
WSUM	(n)=1	0.900	1.800
	(n)=96	4.700	5.600
WSUM_U	(n)=1	0.900	1.800
	(n)=96	4.700	5.600
DWSUM	(n)=1	2.300	7.000
	(n)=96	9.400	14.100
DWSUM_U	(n)=1	2.400	7.800
	(n)=96	9.500	14.900
MEAN	(n)=1	1.900	4.900
	(n)=96	5.100	8.800
MEAN_U	(n)=1	1.900	4.800
	(n)=96	4.700	8.700
DMEAN	(n)=1	2.600	7.800
	(n)=96	8.900	14.200
DMEAN_U	(n)=1	2.200	7.500
	(n)=96	8.300	13.400
SQRT	—	1.800	3.300
DSQRT	—	1.800	3.300
CRC	SM772=OFF, (n)=1	3.200	7.900
	SM772=OFF, (n)=96	12.200	18.800
	SM772=ON, (n)=1	3.200	7.900
	SM772=ON, (n)=96	12.200	18.800
DBOPEN	When executed (for the programmable controller CPU with firmware version earlier than "28")	18.300	20.900
	When executed (for the programmable controller CPU with firmware version "28" or later)	19.300	24.500

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
DBCLOSE	When executed (for the programmable controller CPU with firmware version earlier than "28")	11.500	13.100
	When executed (for the programmable controller CPU with firmware version "28" or later)	13.300	18.300
DBINSERT	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	25.400	28.900
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	46.400	50.100
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	23.300	29.100
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	40.200	45.800
DBUPDATE	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	31.700	34.700
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	49.300	52.900
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	29.900	35.900
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	46.900	53.200
DBSELECT	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	32.100	35.300
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	51.600	55.000
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	32.800	38.300
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	48.900	55.200
DBDELETE	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	26.000	29.900
	(s3)=2 (for the programmable controller CPU with firmware version earlier than "28")	27.400	31.200
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	24.800	30.900
	(s3)=2 (for the programmable controller CPU with firmware version "28" or later)	26.600	32.500
DBIMPORT	When executed (for the programmable controller CPU with firmware version earlier than "28")	16.200	19.200
	When executed (for the programmable controller CPU with firmware version "28" or later)	17.800	22.400
DBEXPORT	When executed (for the programmable controller CPU with firmware version earlier than "28")	16.000	19.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	17.300	22.500
DBTRANS	When executed (for the programmable controller CPU with firmware version earlier than "28")	11.400	13.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	12.800	18.500
DBCOMMIT	When executed (for the programmable controller CPU with firmware version earlier than "28")	11.600	13.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	12.800	18.500
DBROLBAK	When executed (for the programmable controller CPU with firmware version earlier than "28")	11.500	12.900
	When executed (for the programmable controller CPU with firmware version "28" or later)	12.800	18.400
RSET	—	1.800	6.300
QDRSET	—	65.100	92.100
ZRRDB	—	1.600	2.900
ZRWRB	—	1.700	3.200
ADRSET	—	1.100	2.000
DATERD	—	3.400	10.700
DATEWR	—	12.200	37.400
DATE+	No carry	3.300	6.800
	Carry	3.400	6.500

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
DATE-	No carry	3.400	6.500
	Carry	3.400	6.600
TIME2SEC	—	1.900	3.700
SEC2TIME	—	1.800	3.900
LDDT=	When compared with the specified date: Continuity	2.900	9.500
	When compared with the specified date: Non-continuity	2.900	9.400
	When compared with the current date: Continuity	4.700	15.000
	When compared with the current date: Non-continuity	4.800	15.100
LDDT<>	When compared with the specified date: Continuity	2.600	9.300
	When compared with the specified date: Non-continuity	2.600	9.400
	When compared with the current date: Continuity	4.600	15.600
	When compared with the current date: Non-continuity	4.800	15.500
LDDT>	When compared with the specified date: Continuity	2.700	9.500
	When compared with the specified date: Non-continuity	2.700	9.700
	When compared with the current date: Continuity	4.600	15.500
	When compared with the current date: Non-continuity	4.600	15.300
LDDT<=	When compared with the specified date: Continuity	2.700	9.900
	When compared with the specified date: Non-continuity	2.800	9.800
	When compared with the current date: Continuity	4.800	14.600
	When compared with the current date: Non-continuity	4.900	14.400
LDDT<	When compared with the specified date: Continuity	2.800	9.900
	When compared with the specified date: Non-continuity	2.700	9.900
	When compared with the current date: Continuity	4.800	15.000
	When compared with the current date: Non-continuity	4.800	14.900
LDDT>=	When compared with the specified date: Continuity	2.700	9.800
	When compared with the specified date: Non-continuity	2.800	9.800
	When compared with the current date: Continuity	5.000	14.500
	When compared with the current date: Non-continuity	4.900	14.700
ANDDT=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.400	10.000
	When compared with the specified date: Non-continuity	2.500	10.100
	When compared with the current date: Continuity	4.600	15.900
	When compared with the current date: Non-continuity	4.500	15.900
ANDDT<>	When not executed	0.00588	
	When compared with the specified date: Continuity	2.900	9.400
	When compared with the specified date: Non-continuity	2.900	9.400
	When compared with the current date: Continuity	4.800	15.800
	When compared with the current date: Non-continuity	4.900	15.800
ANDDT>	When not executed	0.00588	
	When compared with the specified date: Continuity	2.700	9.700
	When compared with the specified date: Non-continuity	2.700	10.000
	When compared with the current date: Continuity	4.700	15.600
	When compared with the current date: Non-continuity	4.700	15.600
ANDDT<=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.700	9.800
	When compared with the specified date: Non-continuity	2.800	9.900
	When compared with the current date: Continuity	4.900	14.600
	When compared with the current date: Non-continuity	4.900	14.500

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
ANDDT<	When not executed	0.00588	
	When compared with the specified date: Continuity	2.600	9.400
	When compared with the specified date: Non-continuity	2.600	9.600
	When compared with the current date: Continuity	4.800	16.200
	When compared with the current date: Non-continuity	4.900	15.200
ANDDT>=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.700	9.900
	When compared with the specified date: Non-continuity	2.700	9.700
	When compared with the current date: Continuity	4.900	14.600
	When compared with the current date: Non-continuity	4.900	14.900
ORDT=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.700	9.800
	When compared with the specified date: Non-continuity	2.600	9.800
	When compared with the current date: Continuity	4.800	15.600
	When compared with the current date: Non-continuity	4.800	15.400
ORDT<>	When not executed	0.00588	
	When compared with the specified date: Continuity	2.800	10.000
	When compared with the specified date: Non-continuity	2.700	10.100
	When compared with the current date: Continuity	4.700	15.900
	When compared with the current date: Non-continuity	4.800	15.600
ORDT>	When not executed	0.00588	
	When compared with the specified date: Continuity	2.800	10.000
	When compared with the specified date: Non-continuity	2.700	10.000
	When compared with the current date: Continuity	4.800	15.600
	When compared with the current date: Non-continuity	4.800	15.700
ORDT<=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.700	9.800
	When compared with the specified date: Non-continuity	2.600	10.000
	When compared with the current date: Continuity	4.900	15.000
	When compared with the current date: Non-continuity	4.900	15.000
ORDT<	When not executed	0.00588	
	When compared with the specified date: Continuity	2.800	10.200
	When compared with the specified date: Non-continuity	2.700	10.100
	When compared with the current date: Continuity	4.800	15.500
	When compared with the current date: Non-continuity	4.900	15.400
ORDT>=	When not executed	0.00588	
	When compared with the specified date: Continuity	2.800	9.500
	When compared with the specified date: Non-continuity	2.800	9.600
	When compared with the current date: Continuity	4.700	14.500
	When compared with the current date: Non-continuity	4.700	14.900
LDTM=	When compared with the specified time: Continuity	2.900	9.500
	When compared with the specified time: Non-continuity	2.900	9.500
	When compared with the current time: Continuity	4.700	15.600
	When compared with the current time: Non-continuity	4.700	14.900
LDTM<>	When compared with the specified time: Continuity	2.600	9.200
	When compared with the specified time: Non-continuity	2.600	9.300
	When compared with the current time: Continuity	4.900	14.500
	When compared with the current time: Non-continuity	4.800	14.500
LDTM>	When compared with the specified time: Continuity	2.600	9.200
	When compared with the specified time: Non-continuity	2.600	9.300
	When compared with the current time: Continuity	4.800	14.600
	When compared with the current time: Non-continuity	4.900	14.700

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
LDTM<=	When compared with the specified time: Continuity	2.600	9.200
	When compared with the specified time: Non-continuity	2.700	9.200
	When compared with the current time: Continuity	4.900	14.500
	When compared with the current time: Non-continuity	4.900	14.600
LDTM<	When compared with the specified time: Continuity	2.600	9.300
	When compared with the specified time: Non-continuity	2.600	9.300
	When compared with the current time: Continuity	4.700	14.800
	When compared with the current time: Non-continuity	4.900	14.700
LDTM>=	When compared with the specified time: Continuity	2.500	9.100
	When compared with the specified time: Non-continuity	2.600	9.200
	When compared with the current time: Continuity	4.900	14.500
	When compared with the current time: Non-continuity	4.800	14.600
ANDTM=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.400	9.100
	When compared with the specified time: Non-continuity	2.500	9.200
	When compared with the current time: Continuity	4.900	14.600
	When compared with the current time: Non-continuity	4.900	14.700
ANDTM<>	When not executed	0.00588	
	When compared with the specified time: Continuity	2.400	9.100
	When compared with the specified time: Non-continuity	2.600	9.400
	When compared with the current time: Continuity	4.900	14.800
	When compared with the current time: Non-continuity	4.900	14.800
ANDTM>	When not executed	0.00588	
	When compared with the specified time: Continuity	2.600	9.400
	When compared with the specified time: Non-continuity	2.600	9.500
	When compared with the current time: Continuity	4.800	14.700
	When compared with the current time: Non-continuity	4.900	14.800
ANDTM<=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.600	9.400
	When compared with the specified time: Non-continuity	2.600	9.400
	When compared with the current time: Continuity	4.900	15.000
	When compared with the current time: Non-continuity	4.900	14.900
ANDTM<	When not executed	0.00588	
	When compared with the specified time: Continuity	2.600	9.500
	When compared with the specified time: Non-continuity	2.600	9.600
	When compared with the current time: Continuity	4.900	14.700
	When compared with the current time: Non-continuity	4.900	14.800
ANDTM>=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.600	9.400
	When compared with the specified time: Non-continuity	2.600	9.400
	When compared with the current time: Continuity	4.900	14.900
	When compared with the current time: Non-continuity	4.800	14.900
ORTM=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.200
	When compared with the specified time: Non-continuity	2.700	9.100
	When compared with the current time: Continuity	5.000	14.900
	When compared with the current time: Non-continuity	5.000	14.800
ORTM<>	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.200
	When compared with the specified time: Non-continuity	2.700	9.200
	When compared with the current time: Continuity	5.000	15.000
	When compared with the current time: Non-continuity	5.000	15.000

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
ORTM>	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.300
	When compared with the specified time: Non-continuity	2.800	9.300
	When compared with the current time: Continuity	5.100	15.100
	When compared with the current time: Non-continuity	5.100	15.100
ORTM<=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.200
	When compared with the specified time: Non-continuity	2.600	9.300
	When compared with the current time: Continuity	5.000	14.900
	When compared with the current time: Non-continuity	4.900	15.000
ORTM<	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.400
	When compared with the specified time: Non-continuity	2.700	9.400
	When compared with the current time: Continuity	5.000	14.800
	When compared with the current time: Non-continuity	4.900	14.900
ORTM>=	When not executed	0.00588	
	When compared with the specified time: Continuity	2.700	9.200
	When compared with the specified time: Non-continuity	2.500	9.200
	When compared with the current time: Continuity	4.900	14.900
	When compared with the current time: Non-continuity	4.900	14.900
TCMP	—	4.200	7.800
TZCP	—	4.700	9.000
S.DATERD	—	3.500	11.600
S.DATE+	No carry	3.500	7.600
	Carry	3.600	7.600
S.DATE-	No borrow	3.600	8.000
	Borrow	3.500	7.800
DUTY	—	2.400	6.600
TIMCHK	—	2.200	4.800
HOURM	—	2.900	6.200
DHOURM	—	2.900	6.200
RFS (X)	(n)=1	5.500	15.500
	(n)=64	13.000	33.800
RFS (Y)	(n)=1	4.700	14.900
	(n)=64	9.000	29.300
COM	When only I/O refresh is selected	6.500	19.700
	When only CC-Link IE Controller Network refresh is selected (control station side)	17.500	34.700
	When only CC-Link IE Field Network refresh is selected (master station side)	17.700	37.400
	When only CC-Link IE Controller Network refresh is selected (normal station side)	18.800	31.800
	When only CC-Link IE Field Network refresh is selected (local station side)	18.100	36.000
	When only CC-Link IE Field Network Basic refresh is selected	7.7	20.4
	When only MELSECNET/H network refresh is selected (control station side)	19.000	40.000
	When only MELSECNET/H network refresh is selected (normal station side)	19.400	38.600
	When only an intelligent function module is selected	6.800	11.500
	Refresh using CPU buffer memory in the multiple CPU system (during END processing)	3.800	12.100
	Import of input/output outside the group of multiple CPU system Input: 64 points + output: 64 points	3.800	12.700
	Device/label access service processing (Communication with the engineering tool, GOT, or other external devices)	7.100	23.600

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
S.ZCOM	When only CC-Link IE Controller Network refresh is selected (control station side)	18.700	51.600
	When only CC-Link IE Field Network refresh is selected (master station side)	22.300	51.500
	When only CC-Link IE Controller Network refresh is selected (normal station side)	17.000	38.000
	When only CC-Link IE Field Network refresh is selected (local station side)	21.000	41.100
	When only MELSECNET/H network refresh is selected (control station side)	20.200	56.900
	When only MELSECNET/H network refresh is selected (normal station side)	20.600	55.500
FROM	Reading buffer memory, (n)=1	3.700	4.300
	Reading buffer memory, (n)=1000	38.000	55.600
	Reading host CPU buffer memory, (n)=1	1.100	1.500
	Reading host CPU buffer memory, (n)=320	14.100	27.000
	Reading another CPU buffer memory, (n)=1	3.700	4.300
	Reading another CPU buffer memory, (n)=320	21.900	39.500
DFROM	Reading buffer memory, (n)=1	4.000	5.500
	Reading buffer memory, (n)=500	39.400	69.400
	Reading host CPU buffer memory, (n)=1	1.200	2.800
	Reading host CPU buffer memory, (n)=320	21.900	42.900
	Reading another CPU buffer memory, (n)=1	3.800	5.500
	Reading another CPU buffer memory, (n)=320	32.300	66.300
TO	Writing to buffer memory, (n)=1	2.100	2.700
	Writing to buffer memory, (n)=1000	45.000	65.500
	Writing to host CPU buffer memory, (n)=1	0.900	1.200
	Writing to host CPU buffer memory, (n)=320	11.200	26.600
DTO	Writing to buffer memory, (n)=1	2.400	4.000
	Writing to buffer memory, (n)=500	44.400	71.000
	Writing to host CPU buffer memory, (n)=1	1.000	2.600
	Writing to host CPU buffer memory, (n)=320	15.500	37.900
FROMD	Reading buffer memory, (n)=1	4.000	5.700
	Reading buffer memory, (n)=1000	41.000	63.600
	Reading host CPU buffer memory, (n)=1	1.400	2.900
	Reading host CPU buffer memory, (n)=320	15.500	34.700
	Reading another CPU buffer memory, (n)=1	4.000	5.700
	Reading another CPU buffer memory, (n)=320	24.500	47.600
DFROMD	Reading buffer memory, (n)=1	4.000	5.700
	Reading buffer memory, (n)=500	41.400	70.400
	Reading host CPU buffer memory, (n)=1	1.200	2.800
	Reading host CPU buffer memory, (n)=320	22.400	45.300
	Reading another CPU buffer memory, (n)=1	3.900	5.600
	Reading another CPU buffer memory, (n)=320	34.000	62.800
TOD	Writing to buffer memory, (n)=1	2.300	4.100
	Writing to buffer memory, (n)=1000	47.100	76.000
	Writing to host CPU buffer memory, (n)=1	1.200	2.700
	Writing to host CPU buffer memory, (n)=320	13.700	37.600
DTOD	Writing to buffer memory, (n)=1	2.300	4.100
	Writing to buffer memory, (n)=500	46.300	81.400
	Writing to host CPU buffer memory, (n)=1	1.000	2.800
	Writing to host CPU buffer memory, (n)=320	17.200	46.500
TYPERD	—	10.500	25.400

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
UNIINFRD	(n)=1	13.400	20.000 (When the Process CPU is used: 22.900)
	(n)=16	26.100	33.100 (When the Process CPU is used: 36.800)
S.RTREAD	—	2.400	5.900
S.RTWRITE	—	13.500	43.800
LOGTRG	—	55.700	74.500
LOGTRGR	—	11.000	18.600
SP.SOCOPEN	TCP: Active	15.600	33.800
	TCP: Unpassive	15.600	33.800
	TCP: Fullpassive	15.600	33.800
	UDP	15.600	33.800
SP.SOCCLOSE	TCP: Executed from own device	15.300	34.000
	TCP: Executed from external device	15.300	34.000
	UDP	15.300	34.000
SP.SOCRCV	TCP: Minimum amount of data (1 byte)	15.000	36.700
	TCP: Maximum amount of data (10238 bytes)	15.000	36.700
	UDP: Minimum amount of data (1 byte)	15.200	36.800
	UDP: Maximum amount of data (10238 bytes)	15.200	36.800
S.SOCRCVS	TCP: Minimum amount of data (1 byte)	15.800	32.000
	TCP: Maximum amount of data (10238 bytes)	141.000	180.000
	UDP: Minimum amount of data (1 byte)	15.800	32.000
	UDP: Maximum amount of data (10238 bytes)	145.000	182.000
SP.SOCSND	TCP: Minimum amount of data (1 byte)	18.000	35.400
	TCP: Maximum amount of data (10238 bytes)	155.000	180.000
	UDP: Minimum amount of data (1 byte)	18.000	35.400
	UDP: Maximum amount of data (10238 bytes)	155.000	180.000
SP.SOCCINF	—	3.100	9.200
SP.SOCCSET	—	3.300	14.500
SP.SOCRMODE	Switching from normal mode to fixed-length mode	4.800	14.200
	Switching from fixed-length mode to normal mode	4.800	13.900
S.SOCRDATA	Minimum amount of data (1 word)	4.200	10.100
	Maximum amount of data (5120 words)	114.400	162.800
SP.ECPRTCL	—	33.200	39.800
SP.SLMPSND	"Read (command: 0401H)" (reading in units of words): Number of read points = 1 point	16.4	41.4
SP.FTPPUT	File name + server path string = 32 characters	76.000	80.000
	File name + server path string = 64 characters	93.000	97.000
SP.FTPGET	File name + server path string = 32 characters	76.000	80.000
	File name + server path string = 64 characters	93.000	97.000
PSTOP	File name: "P1"	43.000	78.000
POFF	File name: "P1"	42.300	77.000
PSCAN	File name: "P1"	43.700	78.200
PID	—	3.900	10.200
S.PIDINIT	1 loop	3.500	7.000
	32 loop	53.300	56.900
S.PIDCONT	1 loops (first time)	16.600	18.100
	1 loops (second and later)	13.900	19.500
	32 loops (first time)	205.000	208.700
	32 loops (second and later)	193.300	214.100

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
S.PIDSTOP	1 loop	1.100	2.600
S.PIDRUN	1 loop	1.500	2.700
S.PIDPRMW	1 loop	3.700	6.500
PIDINIT	1 loop	3.000	6.700
	32 loop	37.000	40.500
PIDCONT	1 loops (first time)	16.600	17.900
	1 loops (second and later)	13.500	15.100
	32 loops (first time)	199.600	201.600
	32 loops (second and later)	190.800	201.700
PIDSTOP	1 loop	1.200	2.500
PIDRUN	1 loop	1.200	2.500
PIDPRMW	1 loop	3.000	6.500
D.DDRD	Number of read data points = 1	75.300	121.500
	Number of read data points = 16	75.500	121.500
	Number of read data points = 96	78.000	124.000
	Number of read data points = 8192	153.100	196.800
D.DDWR	Number of read data points = 1	73.900	121.100
	Number of read data points = 16	74.300	121.200
	Number of read data points = 96	74.400	121.400
	Number of read data points = 8192	74.500	121.400
M.DDRD	Number of write data points = 1	64.700	112.400
	Number of write data points = 16	65.700	112.700
	Number of write data points = 96	65.400	113.100
	Number of write data points = 8192	145.200	188.200
M.DDWR	Number of write data points = 1	63.600	111.400
	Number of write data points = 16	63.600	112.400
	Number of write data points = 96	63.900	113.500
	Number of write data points = 8192	64.700	113.600
S.IN	The loop is running and the ALM bit does not turn on.	32.400	33.800
S.OUT1	The loop is running in AUT mode and the ALM bit does not turn on.	27.500	27.900
S.OUT2	The loop is running in AUT mode and the ALM bit does not turn on.	25.700	26.600
S.MOUT	The loop is running in MAN mode.	19.900	20.400
S.DUTY	Execution cycle = 1 Control output cycle = 10 The loop is running in AUT mode and the ALM bit does not turn on.	29.200	29.900
S.BC	The loop is running in AUT mode and the ALM bit does not turn on.	21.200	21.500
S.PSUM	Integration start signal = On, Integration hold signal = Off	14.700	15.700
S.PID	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 Integral constant ≠ 0 Derivative constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	48.700	49.700
S.2PID	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 Integral constant ≠ 0 Derivative constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	57.300	59.100
S.PIDP	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 Integral constant ≠ 0 Derivative constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	53.200	54.600

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
S.SPI	Set value pattern = 3 (without a cascade) Tracking bit = 0 Operating time = Sample cycle (ST=STHT) Integral constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	35.900	36.900
S.IPD	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 Integral constant ≠ 0 Derivative constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	45.600	47.800
S.BPI	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 Integral constant ≠ 0 The loop is running in AUT mode and the ALM bit does not turn on.	34.700	35.400
S.R	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 The loop is running in AUT mode.	31.300	32.600
S.PHPL	The loop is running in AUT mode and the ALM bit does not turn on.	38.000	39.500
S.LLAG	Input data = 50 with lead-lag compensation Lead time = 1, Lag time = 1	21.100	21.400
S.I	Input data = 50, Integral time = 1 Output initial value = 0	17.100	17.700
S.D	Input data = 50, Derivative time = 1 Output initial value = 0	18.500	18.900
S.DED	Input data = 50 Operation control signal 0 → 1 Data sampling interval = 1 Sampling count = 10 Output initial value = 0 Initial output switching = 0	10.200	10.900
S.HS	Number of inputs = 5 Input data = 50, 100, 150, 200, 250	13.300	13.700
S.LS	Number of inputs = 5 Input data = 50, 100, 150, 200, 250	13.000	13.200
S.MID	Number of inputs = 5 Input data = 50, 100, 150, 200, 250	17.400	17.900
S.AVE	Number of inputs = 2, Input data = 50, 100	16.000	16.300
S.LIMT	Input data = 50 Upper limit value = 100 Lower limit value = 0 Upper limit hysteresis = 0 Lower limit hysteresis = 0	18.200	18.700
S.VLMT1	Input data = 50 Positive direction limit value = 100 Negative direction limit value = 100 Positive direction hysteresis = 0 Negative direction hysteresis = 0	17.500	17.600
S.VLMT2	Input data = 50 Positive direction limit value = 100 Negative direction limit value = 100 Positive direction hysteresis = 0 Negative direction hysteresis = 0	17.300	17.800
S.ONF2	Input data = 10 Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 The loop is running in MAN mode.	32.600	34.100

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
S.ONF3	Input data = 10 Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = Control cycle = 1 The loop is running in MAN mode.	34.500	35.800
S.DBND	Input data = 50 Dead band upper limit = 100, Dead band lower limit = 0 Input range = 1	17.000	17.600
S.PGS	Number of operation constant break points = 16 Operation type = 0 (Hold type) Execution cycle = 1 Set value = 10 The loop is running in AUT mode and the ALM bit does not turn on.	27.500	28.000
S.SEL	Set value pattern = 18H (E1 and E2 used, without a cascade) Tracking bit = 0 The loop is running in AUT mode and the ALM bit does not turn on.	31.300	32.400
S.BUMP	Output set value = 0, Output control value = 50 Mode switching signal = 1 Delay time = 1, Delay time zone = 1	12.000	12.400
S.AMR	Output addition value = 50, Output subtraction value = 50 Output set value = 0, Operation output signal = 1 Output addition signal = 1, Output subtraction signal = 0 Output upper limit value = 50, Output lower limit value = 0	15.100	15.600
S.FG	Input data = 50, Number of break points = 2 Break point coordinates (30, 40), (60, 70)	21.600	21.900
S.IFG	Input data = 50, Number of break points = 2 Break point coordinates (30, 40), (60, 70)	20.600	21.300
S.FLT	Input data = 50, Data sampling interval = 1 Sampling count = 10	20.800	21.500
S.SUM	Input data = 50 Input low-cut value = 0, Initial value = 0 Input range = 1	17.500	18.000
S.TPC	When both temperature and pressure are corrected Differential pressure = 100, Measured temperature = 300 Measured pressure = 10000, Design temperature = 0 Bias (temperature) = 273.15 Design pressure = 0 Bias (pressure) = 10332.0	19.300	19.800
S.ENG	Input data = 50, Engineering value upper limit = 100 Engineering value lower limit = 0	18.800	19.000
S.IENG	Input data = 50, Engineering value upper limit = 100 Engineering value lower limit = 0	18.600	18.900
S.ADD	Number of inputs = 2, Input data = 50, 100 Number of coefficients = 2, Coefficient = 1, 1, Bias = 0	17.400	17.800
S.SUB	Number of inputs = 2, Input data = 50, 100 Number of coefficients = 2, Coefficient = 1, 1, Bias = 0	18.800	19.200
S.MUL	Number of inputs = 2, Input data = 50, 100 Number of coefficients = 2, Coefficient = 1, 1, Bias = 0	17.500	17.900
S.DIV	Input data = 50, 100 Coefficient = 1, 1, 1, Bias = 0, 0, 0	18.600	19.000
S.SQR	Input data = 50 Output low-cut value = 0, Coefficient = 10	15.800	16.500
S.ABS	Input data = 50	11.400	11.800
S.>	Input data = 50, 100 Set value = 0, Hysteresis = 0	15.700	16.100
S.<	Input data = 50, 100 Set value = 0, Hysteresis = 0	13.900	14.200
S.=	Input data = 50, 100 Set value = 0	14.000	14.600
S.>=	Input data = 50, 100 Set value = 0, Hysteresis = 0	15.600	16.200

Instruction name	Condition	Processing time (μs)	
		Minimum	Maximum
S.<=	Input data = 50, 100 Set value = 0, Hysteresis = 0	13.800	14.100
S.AT1	Set value pattern = 3 (without a cascade) Tracking bit = 0 Execution cycle = 1 The loop is running in MAN mode.	24.600	25.700
LD [S□]	Continuity/Non-continuity	0.500	1.100
LD [BL□\S□]	Continuity/Non-continuity	2.800	5.000
LD [BL□]	Continuity/Non-continuity	1.800	3.600
AND [S□]	Continuity/Non-continuity	0.500	1.100
AND [BL□\S□]	Continuity/Non-continuity	2.800	5.000
AND [BL□]	Continuity/Non-continuity	1.800	3.600
OR [S□]	Continuity/Non-continuity	0.500	1.100
OR [BL□\S□]	Continuity/Non-continuity	2.800	5.000
OR [BL□]	Continuity/Non-continuity	1.800	3.600
LDI [S□]	Continuity/Non-continuity	0.500	1.100
LDI [BL□\S□]	Continuity/Non-continuity	2.800	5.000
LDI [BL□]	Continuity/Non-continuity	1.800	3.600
ANI [S□]	Continuity/Non-continuity	0.500	1.100
ANI [BL□\S□]	Continuity/Non-continuity	2.800	5.000
ANI [BL□]	Continuity/Non-continuity	1.800	3.600
ORI [S□]	Continuity/Non-continuity	0.500	1.100
ORI [BL□\S□]	Continuity/Non-continuity	2.800	5.000
ORI [BL□]	Continuity/Non-continuity	1.800	3.600
MOV(P) [K4S□]	—	2.600	6.200
MOV(P) [BL□\K4S□]	—	3.800	9.200
DMOV(P) [K8S□]	—	2.600	6.200
DMOV(P) [BL□\K8S□]	—	3.800	9.200
BMOV(P) [K4S□]	(n)=1	4.200	14.100
	(n)=96	4.900	15.100
BMOV(P) [BL□\K4S□]	(n)=1	5.600	17.600
	(n)=96	6.300	18.200
SET [S□]	No change time	3.400	8.400
	Change time	3.400	13.600
SET [BL□\S□]	No change time	3.400	8.400
	Change time	3.400	13.600
SET [BL□]	No change time	1.800	3.500
	Change time	1.800	3.500
RST [S□]	No change time	2.300	6.400
	Change time	3.200	7.500
RST [BL□\S□]	No change time	2.300	6.400
	Change time	3.000	7.500
RST [BL□]	No change time	2.400	4.400
	Change time	2.400	4.400
PAUSE [BL□]	—	1.800	3.100
RSTART [BL□]	—	1.800	3.100
BRSET [S□]	—	2.100	3.500
SP.CONTSW	SM1646=ON	10.600	16.000
	SM1646=OFF	137.200	169.700
DCONTSW	Enabled → Disabled	1.400	1.600
ECONTSW	Disabled → Enabled	1.500	1.600

Time added to instruction processing time

When using the file register (R/ZR), module access device (U□\G□), or link direct device (J□\□), add extra time described below to each instruction processing time.

Device name		**Data type**	**Device part specification**	**Additional time (μs)**	
				R04CPU, R04ENCPU	**R08CPU, R08ENCPU, R16CPU, R16ENCPU, R32CPU, R32ENCPU, R120CPU, R120ENCPU**
File register (R/ZR)	When the extended SRAM cassette is not inserted	Bit	Source	0.074	0.043
			Destination	0.023	0.023
		Word	Source	0.074	0.043
			Destination	0.023	0.023
		Double word	Source	0.148	0.085
			Destination	0.044	0.044
	When the extended SRAM cassette is inserted	Bit	Source	0.099	0.099
			Destination	0.028	0.028
		Word	Source	0.099	0.099
			Destination	0.028	0.028
		Double word	Source	0.198	0.198
			Destination	0.054	0.054
Module access device (U□\G□)		Bit	Source	13.000	13.000
			Destination	14.000	14.000
		Word	Source	13.000	13.000
			Destination	14.000	14.000
		Double word	Source	13.000	13.000
			Destination	14.000	14.000
Link direct device (J□\□)		Bit	Source	51.000	51.000
			Destination	52.000	52.000
		Word	Source	51.000	51.000
			Destination	52.000	52.000
		Double word	Source	51.000	51.000
			Destination	52.000	52.000

Appendix 2 Number of Basic Steps and Availability of Subset Processing

The number of basic steps and the availability of subset processing are shown below.

Instruction name	Number of basic steps	Subset availability
LD	1	○
LDI	1	○
AND	1	○
ANI	1	○
OR	1	○
ORI	1	○
LDP	2	○
LDF	2	○
ANDP	2	○
ANDF	2	○
ORP	2	○
ORF	2	○
LDPI	2	○
LDFI	2	○
ANDPI	2	○
ANDFI	2	○
ORPI	2	○
ORFI	2	○
ANB	1	—
ORB	1	—
MPS	1	—
MRD	1	—
MPP	1	—
INV	1	—
MEP	1	—
MEF	1	—
EGP	1	—
EGF	1	—
OUT	1	○
OUT T/ST	4	—
OUT LT/LST	2	—
OUT C	4	—
OUT LC	4	—
OUT F	2	—
OUTH T/ST	4	—
SET	1	○
RST	1	○
SET F	3	—
RST F	3	—
PLS	2	○
PLF	2	○
FF	2	○
DELTA	2	—
DELTAP	3	—
SFT	2	—
SFTP	3	—
MC	2	○
MCR	1	—

Instruction name	Number of basic steps	Subset availability
FEND	2	—
END	2	—
STOP	1	—
NOP	1	—
LD=	3	○
LD<>	3	○
LD>	3	○
LD<=	3	○
LD<	3	○
LD>=	3	○
AND=	3	○
AND<>	3	○
AND>	3	○
AND<=	3	○
AND<	3	○
AND>=	3	○
OR=	3	○
OR<>	3	○
OR>	3	○
OR<=	3	○
OR<	3	○
OR>=	3	○
LD=_U	3	○
LD<>_U	3	○
LD>_U	3	○
LD<=_U	3	○
LD<_U	3	○
LD>=_U	3	○
AND=_U	3	○
AND<>_U	3	○
AND>_U	3	○
AND<=_U	3	○
AND<_U	3	○
AND>=_U	3	○
OR=_U	3	○
OR<>_U	3	○
OR>_U	3	○
OR<=_U	3	○
OR<_U	3	○
OR>=_U	3	○
LDD=	3	○
LDD<>	3	○
LDD>	3	○
LDD<=	3	○
LDD<	3	○
LDD>=	3	○
ANDD=	3	○
ANDD<>	3	○
ANDD>	3	○
ANDD<=	3	○
ANDD<	3	○
ANDD>=	3	○
ORD=	3	○

Instruction name	Number of basic steps	Subset availability
ORD<>	3	○
ORD>	3	○
ORD<=	3	○
ORD<	3	○
ORD>=	3	○
LDD=_U	3	○
LDD<>_U	3	○
LDD>_U	3	○
LDD<=_U	3	○
LDD<_U	3	○
LDD>=_U	3	○
ANDD=_U	3	○
ANDD<>_U	3	○
ANDD>_U	3	○
ANDD<=_U	3	○
ANDD<_U	3	○
ANDD>=_U	3	○
ORD=_U	3	○
ORD<>_U	3	○
ORD>_U	3	○
ORD<=_U	3	○
ORD<_U	3	○
ORD>=_U	3	○
CMP	4	—
CMPP	5	—
CMP_U	4	—
CMPP_U	5	—
DCMP	4	—
DCMPP	5	—
DCMP_U	4	—
DCMPP_U	5	—
ZCP	5	—
ZCPP	6	—
ZCP_U	5	—
ZCPP_U	6	—
DZCP	5	—
DZCPP	6	—
DZCP_U	5	—
DZCPP_U	6	—
BKCMP=	5	—
BKCMP<>	5	—
BKCMP>	5	—
BKCMP<=	5	—
BKCMP<	5	—
BKCMP>=	5	—
BKCMP=P	6	—
BKCMP<>P	6	—
BKCMP>P	6	—
BKCMP<=P	6	—
BKCMP<P	6	—
BKCMP>=P	6	—
BKCMP=_U	5	—
BKCMP<>_U	5	—

Instruction name	Number of basic steps	Subset availability
BKCMP>_U	5	—
BKCMP<=_U	5	—
BKCMP<_U	5	—
BKCMP>=_U	5	—
BKCMP=P_U	6	—
BKCMP<>P_U	6	—
BKCMP>P_U	6	—
BKCMP<=P_U	6	—
BKCMP<P_U	6	—
BKCMP>=P_U	6	—
DBKCMP=	5	—
DBKCMP<>	5	—
DBKCMP>	5	—
DBKCMP<=	5	—
DBKCMP<	5	—
DBKCMP>=	5	—
DBKCMP=P	6	—
DBKCMP<>P	6	—
DBKCMP>P	6	—
DBKCMP<=P	6	—
DBKCMP<P	6	—
DBKCMP>=P	6	—
DBKCMP=_U	5	—
DBKCMP<>_U	5	—
DBKCMP>_U	5	—
DBKCMP<=_U	5	—
DBKCMP<_U	5	—
DBKCMP>=_U	5	—
DBKCMP=P_U	6	—
DBKCMP<>P_U	6	—
DBKCMP>P_U	6	—
DBKCMP<=P_U	6	—
DBKCMP<P_U	6	—
DBKCMP>=P_U	6	—
+ (s) (d)	3	○
+P (s) (d)	4	○
+ (s1) (s2) (d)	3	○
+P (s1) (s2) (d)	4	○
+_U (s) (d)	3	○
+P_U (s) (d)	4	○
+_U (s1) (s2) (d)	3	○
+P_U (s1) (s2) (d)	4	○
- (s) (d)	3	○
-P (s) (d)	4	○
- (s1) (s2) (d)	3	○
-P (s1) (s2) (d)	4	○
-_U (s) (d)	3	○
-P_U (s) (d)	4	○
-_U (s1) (s2) (d)	3	○
-P_U (s1) (s2) (d)	4	○
D+ (s) (d)	3	○
D+P (s) (d)	4	○
D+ (s1) (s2) (d)	3	○

Instruction name	Number of basic steps	Subset availability
D+P (s1) (s2) (d)	4	○
D+_U (s) (d)	3	○
D+P_U (s) (d)	4	○
D+_U (s1) (s2) (d)	3	○
D+P_U (s1) (s2) (d)	4	○
D- (s) (d)	3	○
D-P (s) (d)	4	○
D- (s1) (s2) (d)	3	○
D-P (s1) (s2) (d)	4	○
D-_U (s) (d)	3	○
D-P_U (s) (d)	4	○
D-_U (s1) (s2) (d)	3	○
D-P_U (s1) (s2) (d)	4	○
*	3	○
*P	4	○
*_U	3	○
*P_U	4	○
/	3	○
/P	4	○
/_U	3	○
/P_U	4	○
D*	3	○
D*P	4	○
D*_U	3	○
D*P_U	4	○
D/	3	○
D/P	4	○
D/_U	3	○
D/P_U	4	○
B+ (s) (d)	3	○
B+P (s) (d)	4	○
B+ (s1) (s2) (d)	4	—
B+P (s1) (s2) (d)	5	—
B- (s) (d)	3	○
B-P (s) (d)	4	○
B- (s1) (s2) (d)	4	—
B-P (s1) (s2) (d)	5	—
DB+ (s) (d)	3	—
DB+P (s) (d)	4	—
DB+ (s1) (s2) (d)	4	—
DB+P (s1) (s2) (d)	5	—
DB- (s) (d)	3	—
DB-P (s) (d)	4	—
DB- (s1) (s2) (d)	4	—
DB-P (s1) (s2) (d)	5	—
B*	4	○
B*P	5	○
B/	4	○
B/P	5	○
DB*	4	—
DB*P	5	—
DB/	4	—
DB/P	5	—

Instruction name	Number of basic steps	Subset availability
BK+	5	—
BK+P	6	—
BK+_U	5	—
BK+P_U	6	—
BK-	5	—
BK-P	6	—
BK-_U	5	—
BK-P_U	6	—
DBK+	5	—
DBK+P	6	—
DBK+_U	5	—
DBK+P_U	6	—
DBK-	5	—
DBK-P	6	—
DBK-_U	5	—
DBK-P_U	6	—
INC	2	○
INCP	3	○
INC_U	2	○
INCP_U	3	○
DEC	2	○
DECP	3	○
DEC_U	2	○
DECP_U	3	○
DINC	2	○
DINCP	3	○
DINC_U	2	○
DINCP_U	3	○
DDEC	2	○
DDECP	3	○
DDEC_U	2	○
DDECP_U	3	○
WAND (s) (d)	3	○
WANDP (s) (d)	4	○
WAND (s1) (s2) (d)	3	○
WANDP (s1) (s2) (d)	4	○
DAND (s) (d)	3	○
DANDP (s) (d)	4	○
DAND (s1) (s2) (d)	3	○
DANDP (s1) (s2) (d)	4	○
BKAND	5	—
BKANDP	6	—
WOR (s) (d)	3	○
WORP (s) (d)	4	○
WOR (s1) (s2) (d)	3	○
WORP (s1) (s2) (d)	4	○
DOR (s) (d)	3	○
DORP (s) (d)	4	○
DOR (s1) (s2) (d)	3	○
DORP (s1) (s2) (d)	4	○
BKOR	5	—
BKORP	6	—
WXOR (s) (d)	3	○

Instruction name	Number of basic steps	Subset availability
WXORP (s) (d)	4	○
WXOR (s1) (s2) (d)	3	○
WXORP (s1) (s2) (d)	4	○
DXOR (s) (d)	3	○
DXORP (s) (d)	4	○
DXOR (s1) (s2) (d)	3	○
DXORP (s1) (s2) (d)	4	○
BKXOR	5	—
BKXORP	6	—
WXNR (s) (d)	3	○
WXNRP (s) (d)	4	○
WXNR (s1) (s2) (d)	3	○
WXNRP (s1) (s2) (d)	4	○
DXNR (s) (d)	3	○
DXNRP (s) (d)	4	○
DXNR (s1) (s2) (d)	3	○
DXNRP (s1) (s2) (d)	4	○
BKXNR	5	—
BKXNRP	6	—
BSET	3	○
BSETP	4	○
BRST	3	○
BRSTP	4	○
TEST	4	○
TESTP	5	○
DTEST	5	○
DTESTP	6	○
BKRST	3	○
BKRSTP	4	○
SFR	4	○
SFRP	5	○
SFL	4	○
SFLP	5	○
BSFR	3	○
BSFRP	4	○
BSFL	3	○
BSFLP	4	○
DSFR	3	○
DSFRP	4	○
DSFL	3	○
DSFLP	4	○
SFTBR	4	○
SFTBRP	5	○
SFTBL	4	○
SFTBLP	5	○
SFTWR	4	○
SFTWRP	5	○
SFTWL	4	○
SFTWLP	5	○
BCD	2	○
BCDP	3	○
DBCD	2	○
DBCDP	3	○

Instruction name	Number of basic steps	Subset availability
BIN	2	○
BINP	3	○
DBIN	2	○
DBINP	3	○
FLT2INT	2	○
FLT2INTP	3	○
FLT2UINT	2	○
FLT2UINTP	3	○
FLT2DINT	2	○
FLT2DINTP	3	○
FLT2UDINT	2	○
FLT2UDINTP	3	○
DBL2INT	3	—
DBL2INTP	4	—
DBL2UINT	3	—
DBL2UINTP	4	—
DBL2DINT	3	—
DBL2DINTP	4	—
DBL2UDINT	3	—
DBL2UDINTP	4	—
INT2UINT	3	○
INT2UINTP	4	○
INT2DINT	3	○
INT2DINTP	4	○
INT2UDINT	3	○
INT2UDINTP	4	○
UINT2INT	3	○
UINT2INTP	4	○
UINT2DINT	3	○
UINT2DINTP	4	○
UINT2UDINT	3	○
UINT2UDINTP	4	○
DINT2INT	2	○
DINT2INTP	3	○
DINT2UINT	3	○
DINT2UINTP	4	○
DINT2UDINT	3	○
DINT2UDINTP	4	○
UDINT2INT	3	○
UDINT2INTP	4	○
UDINT2UINT	3	○
UDINT2UINTP	4	○
UDINT2DINT	3	○
UDINT2DINTP	4	○
GRY	2	○
GRYP	3	○
GRY_U	2	○
GRYP_U	3	○
DGRY	2	○
DGRYP	3	○
DGRY_U	2	○
DGRYP_U	3	○
GBIN	2	○

Instruction name	Number of basic steps	Subset availability
GBINP	3	○
GBIN_U	2	○
GBINP_U	3	○
DGBIN	2	○
DGBINP	3	○
DGBIN_U	2	○
DGBINP_U	3	○
BKBCD	4	—
BKBCDP	5	—
BKBIN	4	—
BKBINP	5	—
DABIN	3	—
DABINP	4	—
DABIN_U	3	—
DABINP_U	4	—
DDABIN	3	—
DDABINP	4	—
DDABIN_U	3	—
DDABINP_U	4	—
HABIN	3	—
HABINP	4	—
DHABIN	3	—
DHABINP	4	—
DABCD	3	—
DABCDP	4	—
DDABCD	3	—
DDABCDP	4	—
VAL	4	—
VALP	5	—
VAL_U	4	—
VALP_U	5	—
DVAL	4	—
DVALP	5	—
DVAL_U	4	—
DVALP_U	5	—
ASC2INT	4	—
ASC2INTP	5	—
EMOD	4	—
EMODP	5	—
NEG	2	○
NEGP	3	○
DNEG	2	○
DNEGP	3	○
DECO	4	—
DECOP	5	—
ENCO	4	—
ENCOP	5	—
SEG	3	○
SEGP	4	○
DIS	4	—
DISP	5	—
UNI	4	—
UNIP	5	—

Instruction name	Number of basic steps	Subset availability
NDIS	4	—
NDISP	5	—
NUNI	4	—
NUNIP	5	—
WTOB	4	—
WTOBP	5	—
BTOW	4	—
BTOWP	5	—
MOV	2	○
MOVP	3	○
DMOV	2	○
DMOVP	3	○
CML	2	○
CMLP	3	○
DCML	2	○
DCMLP	3	○
SMOV	6	—
SMOVP	7	—
CMLB	3	○
CMLBP	4	○
BMOV	4	○
BMOVP	5	○
BMOVL	4	○
BMOVLP	5	○
FMOV	4	○
FMOVP	5	○
FMOVL	4	○
FMOVLP	5	○
DFMOV	4	○
DFMOVP	5	○
DFMOVL	4	○
DFMOVLP	5	○
XCH	3	○
XCHP	4	○
DXCH	3	○
DXCHP	4	○
BXCH	4	—
BXCHP	5	—
SWAP	2	—
SWAPP	3	—
DSWAP	3	—
DSWAPP	4	—
MOVB	2	○
MOVBP	3	○
BLKMOVB	4	—
BLKMOVBP	5	—
ROR	4	○
RORP	5	○
RCR	4	○
RCRP	5	○
DROR	4	○
DRORP	5	○
DRCR	4	○

Instruction name	Number of basic steps	Subset availability
DRCRP	5	○
ROL	4	○
ROLP	5	○
RCL	4	○
RCLP	5	○
DROL	4	○
DROLP	5	○
DRCL	4	○
DRCLP	5	○
CJ	4	—
SCJ	5	—
JMP	4	—
GOEND	1	—
DI	1	—
DI (s)	2	○
EI	1	○
IMASK	2	○
SIMASK	3	○
IRET	1	—
WDT	1	—
WDTP	2	—
FOR	2	○
NEXT	1	—
BREAK	4	—
BREAKP	5	—
CALL (P)	3	—
CALL (P) (s1)	4	—
CALL (P) (s1) (s2)	5	—
CALL (P) (s1) (s2) (s3)	6	—
CALL (P) (s1) (s2) (s3) (s4)	7	—
CALL (P) (s1) (s2) (s3) (s4) (s5)	8	—
CALLP (P)	4	—
CALLP (P) (s1)	5	—
CALLP (P) (s1) (s2)	6	—
CALLP (P) (s1) (s2) (s3)	7	—
CALLP (P) (s1) (s2) (s3) (s4)	8	—
CALLP (P) (s1) (s2) (s3) (s4) (s5)	9	—
RET	1	—
FCALL (P)	3	—
FCALL (P) (s1)	4	—
FCALL (P) (s1) (s2)	5	—
FCALL (P) (s1) (s2) (s3)	6	—
FCALL (P) (s1) (s2) (s3) (s4)	7	—
FCALL (P) (s1) (s2) (s3) (s4) (s5)	8	—
FCALLP (P)	4	—
FCALLP (P) (s1)	5	—
FCALLP (P) (s1) (s2)	6	—
FCALLP (P) (s1) (s2) (s3)	7	—
FCALLP (P) (s1) (s2) (s3) (s4)	8	—
FCALLP (P) (s1) (s2) (s3) (s4) (s5)	9	—
ECALL (file name) (P)	4 + Number of characters in the file name	—

Instruction name	Number of basic steps	Subset availability
ECALL (file name) (P) (s1)	5 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2)	6 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3)	7 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3) (s4)	8 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3) (s4) (s5)	9 + Number of characters in the file name	—
ECALLP (file name) (P)	5 + Number of characters in the file name	—
ECALLP (file name) (P) (s1)	6 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2)	7 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3)	8 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3) (s4)	9 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3) (s4) (s5)	10 + Number of characters in the file name	—
EFCALL (file name) (P)	4 + Number of characters in the file name	—
EFCALL (file name) (P) (s1)	5 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2)	6 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2) (s3)	7 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2) (s3) (s4)	8 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2) (s3) (s4) (s5)	9 + Number of characters in the file name	—
EFCALLP (file name) (P)	5 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1)	6 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2)	7 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3)	8 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3) (s4)	9 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3) (s4) (s5)	10 + Number of characters in the file name	—
XCALL (P)	4	—
XCALL (P) (s1)	5	—
XCALL (P) (s1) (s2)	6	—
XCALL (P) (s1) (s2) (s3)	7	—
XCALL (P) (s1) (s2) (s3) (s4)	8	—
XCALL (P) (s1) (s2) (s3) (s4) (s5)	9	—
FIFR	3	—
FIFRP	4	—
FPOP	3	—
FPOPP	4	—
FIFW	3	—
FIFWP	4	—
FINS	4	—

Instruction name	Number of basic steps	Subset availability
FINSP	5	—
FDEL	4	—
FDELP	5	—
S.DEVLD	4	—
SP.DEVLD	5	—
SP.DEVST	6	—
SP.FREAD	8	—
SP.FWRITE	8	—
LEDR	2	—
PALERT	2	—
PALERTP	3	—
PABORT	2	—
LD$=	3	○
LD$<>	3	○
LD$>	3	○
LD$<=	3	○
LD$<	3	○
LD$>=	3	○
AND$=	3	○
AND$<>	3	○
AND$>	3	○
AND$<=	3	○
AND$<	3	○
AND$>=	3	○
OR$=	3	○
OR$<>	3	○
OR$>	3	○
OR$<=	3	○
OR$<	3	○
OR$>=	3	○
$+ (s) (d)	3	○
$+P (s) (d)	4	○
$+ (s1) (s2) (d)	4	○
$+P (s1) (s2) (d)	5	○
$MOV	3	—
$MOVP	4	—
$MOV_WS	3	—
$MOVP_WS	4	—
BINDA	3	—
BINDAP	4	—
BINDA_U	3	—
BINDAP_U	4	—
DBINDA	3	—
DBINDAP	4	—
DBINDA_U	3	—
DBINDAP_U	4	—
BINHA	3	—
BINHAP	4	—
DBINHA	3	—
DBINHAP	4	—
STR	4	—
STRP	5	—
STR_U	4	—

Instruction name	Number of basic steps	Subset availability
STRP_U	5	—
DSTR	4	—
DSTRP	5	—
DSTR_U	4	—
DSTRP_U	5	—
BCDDA	3	—
BCDDAP	4	—
DBCDDA	3	—
DBCDDAP	4	—
ESTR	4	—
ESTRP	5	—
INT2ASC	4	—
INT2ASCP	5	—
WS2SJIS	3	—
WS2SJISP	4	—
SJIS2WS	3	—
SJIS2WSP	4	—
SJIS2WSB	3	—
SJIS2WSBP	4	—
LEN	3	○
LENP	4	○
RIGHT	4	—
RIGHTP	5	—
LEFT	4	—
LEFTP	5	—
MIDR	4	—
MIDRP	5	—
MIDW	4	—
MIDWP	5	—
INSTR	5	—
INSTRP	6	—
STRINS	4	—
STRINSP	5	—
STRDEL	4	—
STRDELP	5	—
LDE=	3	○
LDE<>	3	○
LDE>	3	○
LDE<=	3	○
LDE<	3	○
LDE>=	3	○
ANDE=	3	○
ANDE<>	3	○
ANDE>	3	○
ANDE<=	3	○
ANDE<	3	○
ANDE>=	3	○
ORE=	3	○
ORE<>	3	○
ORE>	3	○
ORE<=	3	○
ORE<	3	○
ORE>=	3	○

Instruction name	Number of basic steps	Subset availability
LDED=	3	—
LDED<>	3	—
LDED>	3	—
LDED<=	3	—
LDED<	3	—
LDED>=	3	—
ANDED=	3	—
ANDED<>	3	—
ANDED>	3	—
ANDED<=	3	—
ANDED<	3	—
ANDED>=	3	—
ORED=	3	—
ORED<>	3	—
ORED>	3	—
ORED<=	3	—
ORED<	3	—
ORED>=	3	—
ECMP	4	—
ECMPP	5	—
EDCMP	4	—
EDCMPP	5	—
EZCP	5	—
EZCPP	6	—
EDZCP	5	—
EDZCPP	6	—
E+ (s) (d)	3	○
E+P (s) (d)	4	○
E+ (s1) (s2) (d)	3	○
E+P (s1) (s2) (d)	4	○
E- (s) (d)	3	○
E-P (s) (d)	4	○
E- (s1) (s2) (d)	3	○
E-P (s1) (s2) (d)	4	○
ED+ (s) (d)	3	—
ED+P (s) (d)	4	—
ED+ (s1) (s2) (d)	4	—
ED+P (s1) (s2) (d)	5	—
ED- (s) (d)	3	—
ED-P (s) (d)	4	—
ED- (s1) (s2) (d)	4	—
ED-P (s1) (s2) (d)	5	—
E*	3	○
E*P	4	○
E/	3	○
E/P	4	○
ED*	4	—
ED*P	5	—
ED/	4	—
ED/P	5	—
INT2FLT	2	○
INT2FLTP	3	○
UINT2FLT	2	○

Instruction name	Number of basic steps	Subset availability
UINT2FLTP	3	○
DINT2FLT	2	○
DINT2FLTP	3	○
UDINT2FLT	2	○
UDINT2FLTP	3	○
DBL2FLT	3	—
DBL2FLTP	4	—
INT2DBL	3	—
INT2DBLP	4	—
UINT2DBL	3	—
UINT2DBLP	4	—
DINT2DBL	3	—
DINT2DBLP	4	—
UDINT2DBL	3	—
UDINT2DBLP	4	—
FLT2DBL	3	—
FLT2DBLP	4	—
EVAL	3	—
EVALP	4	—
EREXP	4	—
EREXPP	5	—
ENEG	2	—
ENEGP	3	—
EDNEG	2	—
EDNEGP	3	—
EMOV	2	○
EMOVP	3	○
EDMOV	2	○
EDMOVP	3	○
SIN	3	—
SINP	4	—
COS	3	—
COSP	4	—
TAN	3	—
TANP	4	—
ASIN	3	—
ASINP	4	—
ACOS	3	—
ACOSP	4	—
ATAN	3	—
ATANP	4	—
SIND	3	—
SINDP	4	—
COSD	3	—
COSDP	4	—
TAND	3	—
TANDP	4	—
ASIND	3	—
ASINDP	4	—
ACOSD	3	—
ACOSDP	4	—
ATAND	3	—
ATANDP	4	—

Instruction name	Number of basic steps	Subset availability
BSIN	3	—
BSINP	4	—
BCOS	3	—
BCOSP	4	—
BTAN	3	—
BTANP	4	—
BASIN	3	—
BASINP	4	—
BACOS	3	—
BACOSP	4	—
BATAN	3	—
BATANP	4	—
RAD	3	—
RADP	4	—
DEG	3	—
DEGP	4	—
RADD	3	—
RADDP	4	—
DEGD	3	—
DEGDP	4	—
ESQRT	3	—
ESQRTP	4	—
EDSQRT	3	—
EDSQRTP	4	—
EXP	3	—
EXPP	4	—
EXPD	3	—
EXPDP	4	—
LOG	3	—
LOGP	4	—
LOGD	3	—
LOGDP	4	—
BSQRT	3	—
BSQRTP	4	—
BDSQRT	3	—
BDSQRTP	4	—
POW	4	—
POWP	5	—
POWD	4	—
POWDP	5	—
LOG10	3	—
LOG10P	4	—
LOG10D	3	—
LOG10DP	4	—
EMAX	4	—
EMAXP	5	—
EDMAX	4	—
EDMAXP	5	—
EMIN	4	—
EMINP	5	—
EDMIN	4	—
EDMINP	5	—
RND	2	—

Instruction name	Number of basic steps	Subset availability
RNDP	3	—
SRND	2	—
SRNDP	3	—
ZPUSH (d)	2	○
ZPUSHP (d)	3	○
ZPUSH (s) (d)	3	○
ZPUSHP (s) (d)	4	○
ZPOP (d)	2	○
ZPOPP (d)	3	○
ZPOP (s) (d)	3	○
ZPOPP (s) (d)	4	○
LIMIT	5	○
LIMITP	6	○
LIMIT_U	5	○
LIMITP_U	6	○
DLIMIT	5	○
DLIMITP	6	○
DLIMIT_U	5	○
DLIMITP_U	6	○
BAND	5	—
BANDP	6	—
BAND_U	5	—
BANDP_U	6	—
DBAND	5	—
DBANDP	6	—
DBAND_U	5	—
DBANDP_U	6	—
ZONE	5	—
ZONEP	6	—
ZONE_U	5	—
ZONEP_U	6	—
DZONE	5	—
DZONEP	6	—
DZONE_U	5	—
DZONEP_U	6	—
SCL	4	—
SCLP	5	—
SCL_U	4	—
SCLP_U	5	—
DSCL	4	—
DSCLP	5	—
DSCL_U	4	—
DSCLP_U	5	—
SCL2	4	—
SCL2P	5	—
SCL2_U	4	—
SCL2P_U	5	—
DSCL2	4	—
DSCL2P	5	—
DSCL2_U	4	—
DSCL2P_U	5	—
UDCNT1	5	—
UDCNT2	5	—

Instruction name	Number of basic steps	Subset availability
TTMR	4	—
STMR	4	—
ROTC	5	—
RAMPQ	7	—
SPD	5	—
PLSY	5	—
PWM	5	—
MTR	5	—
CCD	4	—
CCDP	5	—
SERDATA	5	—
SERDATAP	6	—
DSERDATA	5	—
DSERDATAP	6	—
SERMM	6	—
SERMMP	7	—
DSERMM	6	—
DSERMMP	7	—
SUM	3	○
SUMP	4	○
DSUM	3	○
DSUMP	4	○
BON	5	—
BONP	6	—
DBON	5	—
DBONP	6	—
MAX	4	—
MAXP	5	—
MAX_U	4	—
MAXP_U	5	—
DMAX	4	—
DMAXP	5	—
DMAX_U	4	—
DMAXP_U	5	—
MIN	4	—
MINP	5	—
MIN_U	4	—
MINP_U	5	—
DMIN	4	—
DMINP	5	—
DMIN_U	4	—
DMINP_U	5	—
SORTD	7	—
SORTD_U	7	—
DSORTD	7	—
DSORTD_U	7	—
WSUM	4	○
WSUMP	5	○
WSUM_U	4	○
WSUMP_U	5	○
DWSUM	4	—
DWSUMP	5	—
DWSUM_U	4	—

Instruction name	Number of basic steps	Subset availability
DWSUMP_U	5	—
MEAN	4	—
MEANP	5	—
MEAN_U	4	—
MEANP_U	5	—
DMEAN	4	—
DMEANP	5	—
DMEAN_U	4	—
DMEANP_U	5	—
SQRT	4	—
SQRTP	5	—
DSQRT	4	—
DSQRTP	5	—
CRC	4	—
CRCP	5	—
DBOPEN	5	—
DBOPENP	6	—
DBCLOSE	4	—
DBCLOSEP	5	—
DBINSERT	7	—
DBINSERTP	8	—
DBUPDATE	8	—
DBUPDATEP	9	—
DBSELECT	8	—
DBSELECTP	9	—
DBDELETE	6	—
DBDELETEP	7	—
DBIMPORT	4	—
DBIMPORTP	5	—
DBEXPORT	4	—
DBEXPORTP	5	—
DBTRANS	4	—
DBTRANSP	5	—
DBCOMMIT	4	—
DBCOMMITP	5	—
DBROLBAK	4	—
DBROLBAKP	5	—
RSET	2	—
RSETP	3	—
QDRSET	2 + Number of characters in the file name	—
QDRSETP	3 + Number of characters in the file name	—
ZRRDB	3	—
ZRRDBP	4	—
ZRWRB	3	—
ZRWRBP	4	—
ADRSET	3	—
ADRSETP	4	—
DATERD	2	—
DATERDP	3	—
DATEWR	2	—
DATEWRP	3	—

Instruction name	Number of basic steps	Subset availability
DATE+	4	—
DATE+P	5	—
DATE-	4	—
DATE-P	5	—
TIME2SEC	3	—
TIME2SECP	4	—
SEC2TIME	3	—
SEC2TIMEP	4	—
LDDT=	4	—
LDDT<>	4	—
LDDT>	4	—
LDDT<=	4	—
LDDT<	4	—
LDDT>=	4	—
ANDDT=	4	—
ANDDT<>	4	—
ANDDT>	4	—
ANDDT<=	4	—
ANDDT<	4	—
ANDDT>=	4	—
ORDT=	4	—
ORDT<>	4	—
ORDT>	4	—
ORDT<=	4	—
ORDT<	4	—
ORDT>=	4	—
LDTM=	4	—
LDTM<>	4	—
LDTM>	4	—
LDTM<=	4	—
LDTM<	4	—
LDTM>=	4	—
ANDTM=	4	—
ANDTM<>	4	—
ANDTM>	4	—
ANDTM<=	4	—
ANDTM<	4	—
ANDTM>=	4	—
ORTM=	4	—
ORTM<>	4	—
ORTM>	4	—
ORTM<=	4	—
ORTM<	4	—
ORTM>=	4	—
TCMP	6	—
TCMPP	7	—
TZCP	5	—
TZCPP	6	—
S.DATERD	2	—
SP.DATERD	3	—
S.DATE+	4	—
SP.DATE+	5	—
S.DATE-	4	—

Instruction name	Number of basic steps	Subset availability
SP.DATE-	5	—
DUTY	5	—
TIMCHK	5	—
HOURM	5	—
DHOURM	5	—
RFS	3	—
RFSP	4	—
COM	1	—
COMP	2	—
S.ZCOM	2	—
SP.ZCOM	3	—
FROM	5	○
FROMP	6	○
DFROM	5	○
DFROMP	6	○
TO	5	○
TOP	6	○
DTO	5	○
DTOP	6	○
FROMD	5	○
FROMDP	6	○
DFROMD	5	○
DFROMDP	6	○
TOD	5	○
TODP	6	○
DTOD	5	○
DTODP	6	○
TYPERD	3	—
TYPERDP	4	—
UNIINFRD	4	—
UNIINFRDP	5	—
S.RTREAD	3	—
SP.RTREAD	4	—
S.RTWRITE	3	—
SP.RTWRITE	4	—
LOGTRG	3	—
LOGTRGR	3	—
SP.SOCOPEN	5	—
SP.SOCCLOSE	6	—
SP.SOCRCV	7	—
S.SOCRCVS	4	—
SP.SOCSND	7	—
SP.SOCCINF	6	—
SP.SOCCSET	5	—
SP.SOCRMODE	5	—
S.SOCRDATA	6	—
SP.SOCRDATA	7	—
SP.ECPRTCL	7	—
SP.SLMPSND	6	—
SP.FTPPUT	7	—
SP.FTPGET	7	—
PSTOP	2 + Number of characters in the file name	—

Instruction name	Number of basic steps	Subset availability
PSTOPP	3 + Number of characters in the file name	—
POFF	2 + Number of characters in the file name	—
POFFP	3 + Number of characters in the file name	—
PSCAN	2 + Number of characters in the file name	—
PSCANP	3 + Number of characters in the file name	—
PID	5	—
S.PIDINIT	2	—
SP.PIDINIT	3	—
S.PIDCONT	2	—
SP.PIDCONT	3	—
S.PIDSTOP	2	—
SP.PIDSTOP	3	—
S.PIDRUN	2	—
SP.PIDRUN	3	—
S.PIDPRMW	3	—
SP.PIDPRMW	4	—
PIDINIT	2	—
PIDINITP	3	—
PIDCONT	2	—
PIDCONTP	3	—
PIDSTOP	2	—
PIDSTOPP	3	—
PIDRUN	2	—
PIDRUNP	3	—
PIDPRMW	3	—
PIDPRMWP	4	—
D.DDRD	9	—
DP.DDRD	10	—
D.DDWR	9	—
DP.DDWR	10	—
M.DDRD	9	—
MP.DDRD	10	—
M.DDWR	9	—
MP.DDWR	10	—
S.IN	5	—
S.OUT1	5	—
S.OUT2	5	—
S.MOUT	5	—
S.DUTY	5	—
S.BC	5	—
S.PSUM	5	—
S.PID	6	—
S.2PID	6	—
S.PIDP	6	—
S.SPI	6	—
S.IPD	6	—
S.BPI	6	—
S.R	6	—
S.PHPL	5	—

Instruction name	Number of basic steps	Subset availability
S.LLAG	5	—
S.I	5	—
S.D	5	—
S.DED	5	—
S.HS	5	—
S.LS	5	—
S.MID	5	—
S.AVE	5	—
S.LIMT	5	—
S.VLMT1	5	—
S.VLMT2	5	—
S.ONF2	6	—
S.ONF3	6	—
S.DBND	5	—
S.PGS	5	—
S.SEL	6	—
S.BUMP	5	—
S.AMR	5	—
S.FG	5	—
S.IFG	5	—
S.FLT	5	—
S.SUM	5	—
S.TPC	5	—
S.ENG	5	—
S.IENG	5	—
S.ADD	5	—
S.SUB	5	—
S.MUL	5	—
S.DIV	5	—
S.SQR	5	—
S.ABS	5	—
S.>	5	—
S.<	5	—
S.=	5	—
S.>=	5	—
S.<=	5	—
S.AT1	6	—
LD [S□]	2	—
LD [BL□\S□]	3	—
LDI [S□]	2	—
LDI [BL□\S□]	3	—
AND [S□]	2	—
AND [BL□\S□]	3	—
ANI [S□]	2	—
ANI [BL□\S□]	3	—
OR [S□]	2	—
OR [BL□\S□]	3	—
ORI [S□]	2	—
ORI [BL□\S□]	3	—
LD [BL□]	2	—
LDI [BL□]	2	—
AND [BL□]	2	—
ANI [BL□]	2	—

Instruction name	Number of basic steps	Subset availability
OR [BL□]	2	—
ORI [BL□]	2	—
MOV [K4S□]	3	—
MOV [BL□\K4S□]	4	—
MOVP [K4S□]	4	—
MOVP [BL□\K4S□]	5	—
DMOV [K8S□]	3	—
DMOV [BL□\K8S□]	4	—
DMOVP [K8S□]	4	—
DMOVP [BL□\K8S□]	5	—
BMOV [K4S□]	4	—
BMOV [BL□\K4S□]	5	—
BMOVP [K4S□]	5	—
BMOVP [BL□\K4S□]	6	—
SET [BL□]	2	—
RST [BL□]	2	—
PAUSE [BL□]	2	—
RSTART [BL□]	2	—
SET [S□]	2	—
SET [BL□\S□]	3	—
RST [S□]	2	—
RST [BL□\S□]	3	—
BRSET	2	—
SP.CONTSW	4	—
DCONTSW	1	—
ECONTSW	1	—

Appendix 3 Determining Three PID Constants

The auto tuning function of PID operation instructions is performed in two methods: limit cycle method and step response method.

Overview of limit cycle method

This section describes the limit cycle method that is a method to determine the amplitude (a) and vibration period (τ, τ_{on}) of the input values, and calculate the proportional gain (K_P), integral time (T_I), and derivative time (T_D) according to the following expression of "Operation characteristics and three constants".

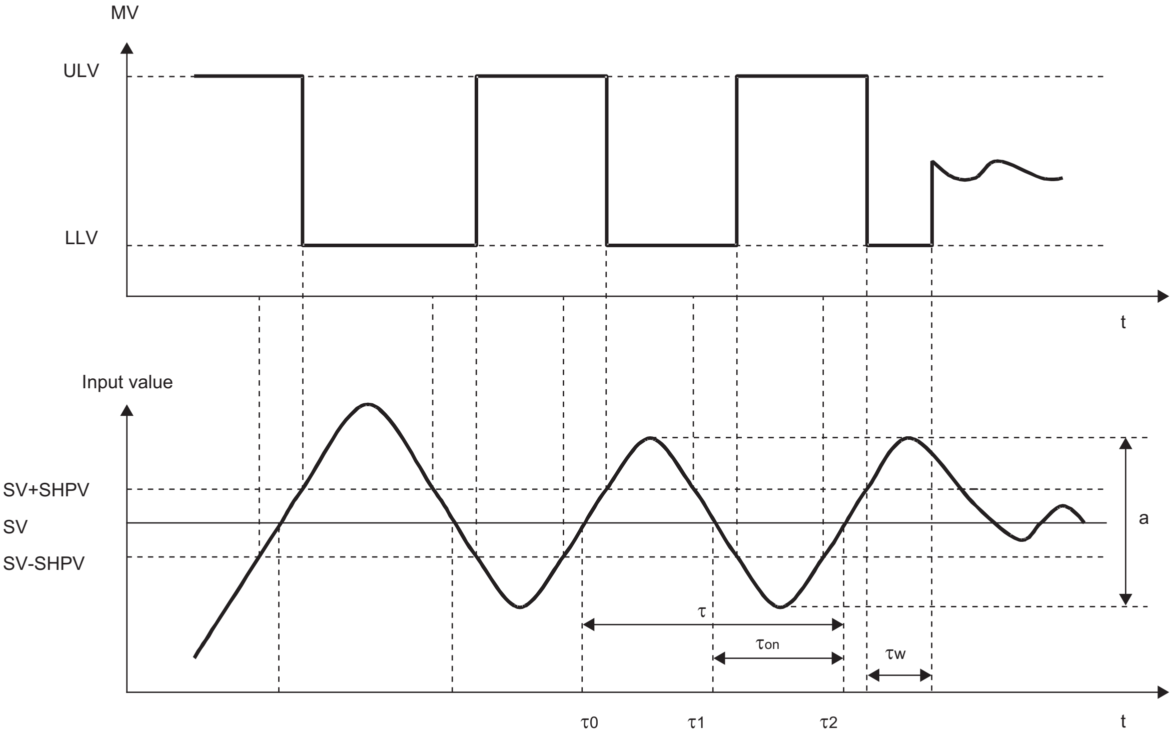
■Limit cycle method

This method determines three PID constants by measuring the variations of input values while two-position control (output by switching between the output upper limit (ULV) and output lower limit (LLV) according to the deviation) is performed.

■Operation characteristics (reverse action example)

After the end of tuning cycle, the output lower limit (LLV) is retained for the manipulated value (MV) during τ_W, and a transition to the normal PID control occurs.

τ_W can be determined by $(50+K_W)/100 \times (\tau-\tau_{on})$, and the wait setting parameter (K_W) can be set in parameter (s3)+28.
(Setting range K_W = -50 to 32717 [%]; If an abnormal range is specified, operation is performed assuming τ_W = 0.)



ULV: Output upper limit value
LLV: Output lower limit value
SV: Set value
t: Time
SHPV: PV value threshold (hysteresis) width

■Operation characteristics and three constants

Control method	Proportional gain (K_P) [%]	Integral time (T_I) [×100ms]	Derivative time (T_D) [×10ms]
Proportional control (P action) only	$\frac{1}{a}$ (ULV-LLV) × 100	—	—
PI control (PI action)	$\frac{0.9}{a}$ (ULV-LLV) × 100	$33 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$	—
PID control (PID action)	$\frac{1.2}{a}$ (ULV-LLV) × 100	$20 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$	$50 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$

Overview of step response method

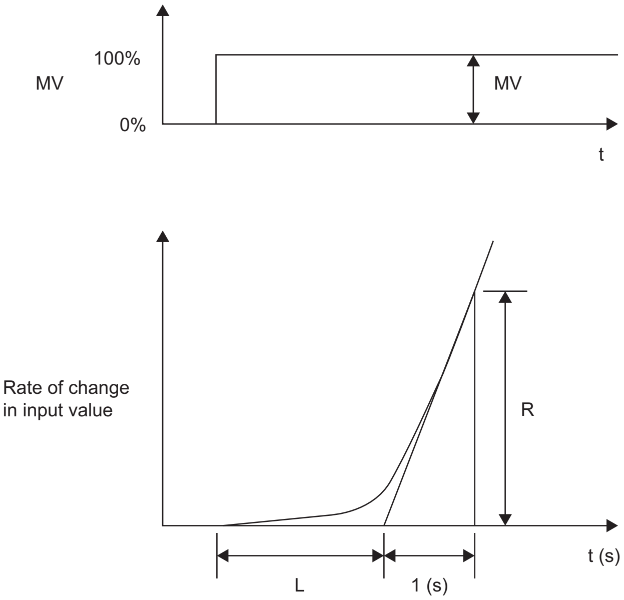
This section describes the step response method that is used to determine the optimum value of three PID constants (proportional gain (K_P), integral time (T_I), and derivative time (T_D)).

■Step response method

This method determines the three PID constants according to the operation characteristics (maximum ramp (R), dead time (L)) that can be determined from input changes by providing the control system with stepwise output of 0→100%[*1].

[*1] Stepwise output can also be determined by 0→75% or 0→50%.

■Operation characteristics



MV: Manipulated value
L: Dead time
R: Maximum ramp
t: Time

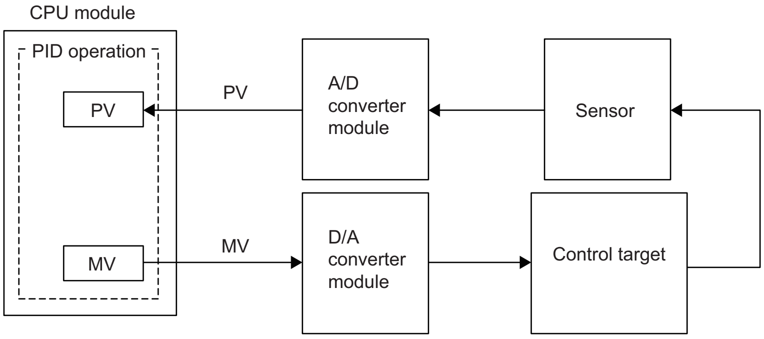
■Operation characteristics and three constants

Control method	Proportional gain (K_P) [%]	Integral time (T_I) [×100ms]	Derivative time (T_D) [×10ms]
Proportional control (P action) only	$\frac{1}{RL} \times MV \times 100$	—	—
PI control (PI action)	$\frac{0.9}{RL} \times MV \times 100$	33L	—
PID control (PID action)	$\frac{1.2}{RL} \times MV \times 100$	20L	50L

Appendix 4 PID Operation Program Examples

This section provides examples of PID operation programs where the PID operation instruction is used.

The following system configuration is used.

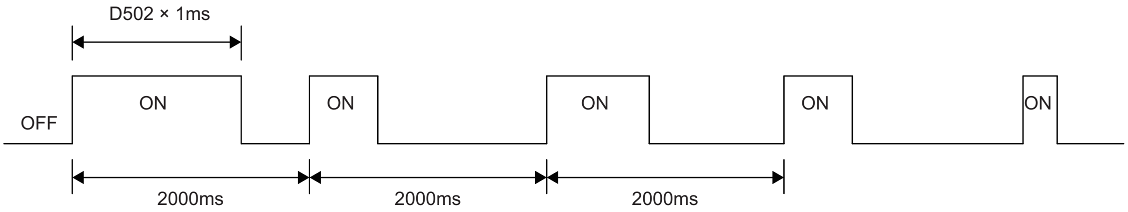


- X2: Auto tuning command
- X3: PID control command
- Y20: Error indication
- Y21: Heater output

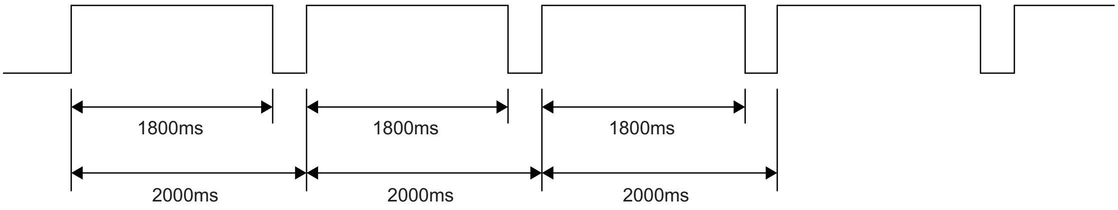
Item				**During auto tuning**	**During PID control**
Set value			(s1)	500 (+50℃)	500 (+50℃)
Control data	Sampling time (T_S)		(s3)	3000ms	500ms
	Input filter (α)		(s3)+2	70%	70%
	Derivative gain (K_D)		(s3)+5	0%	0%
	Output upper limit value		(s3)+22	2000 (2 seconds)	2000 (2 seconds)
	Output lower limit value		(s3)+23	0	0
	Action setting (ACT)	Input variation alarm	Bit 1 of (s3)+1	Disabled	Disabled
		Output variation alarm	Bit 2 of (s3)+1	Disabled	Disabled
		Output upper/lower limit value setting	Bit 5 of (s3)+1	Enabled	Enabled
Output value			(d)	1800	According to operation

Operation of output value

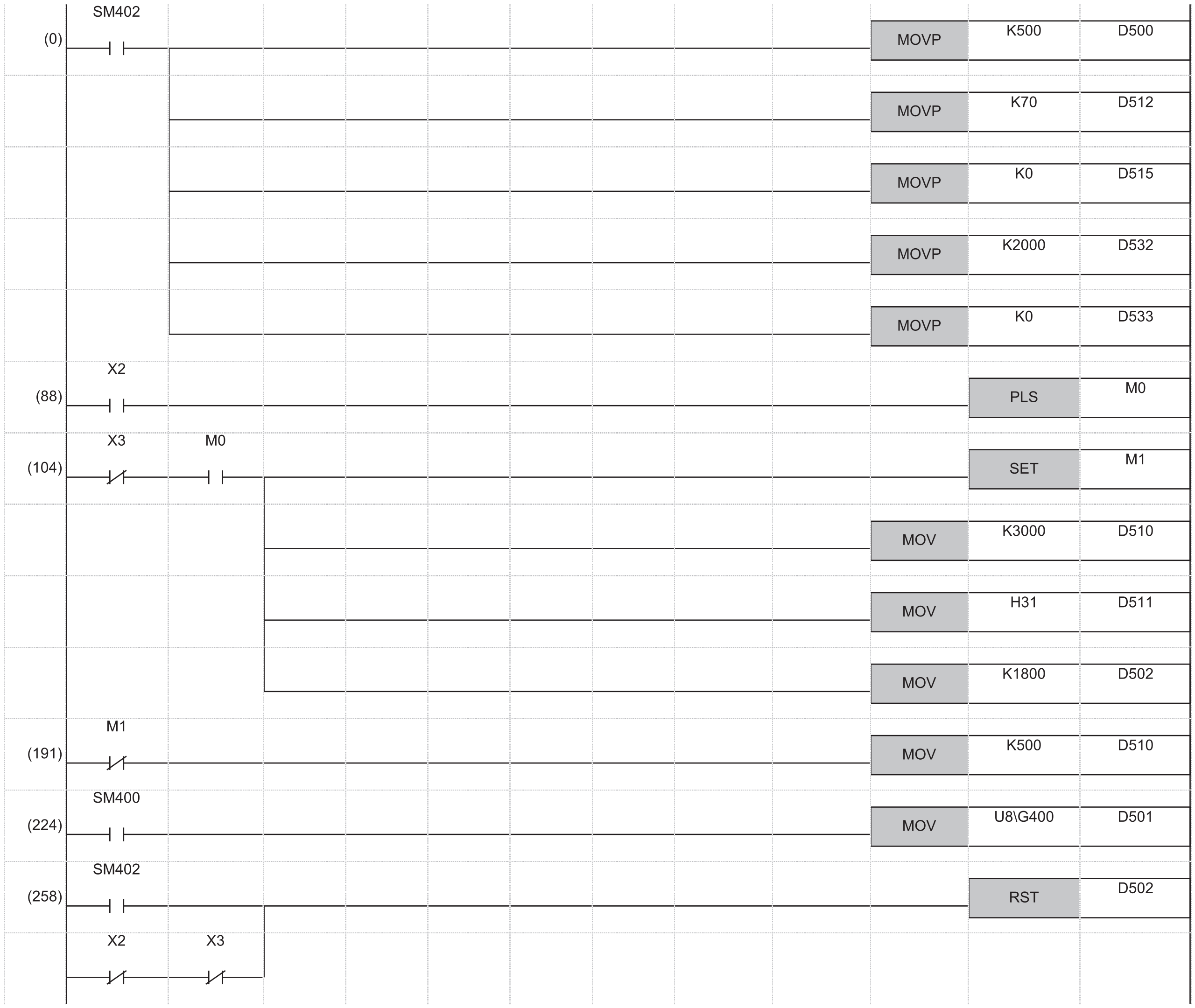
■PID control



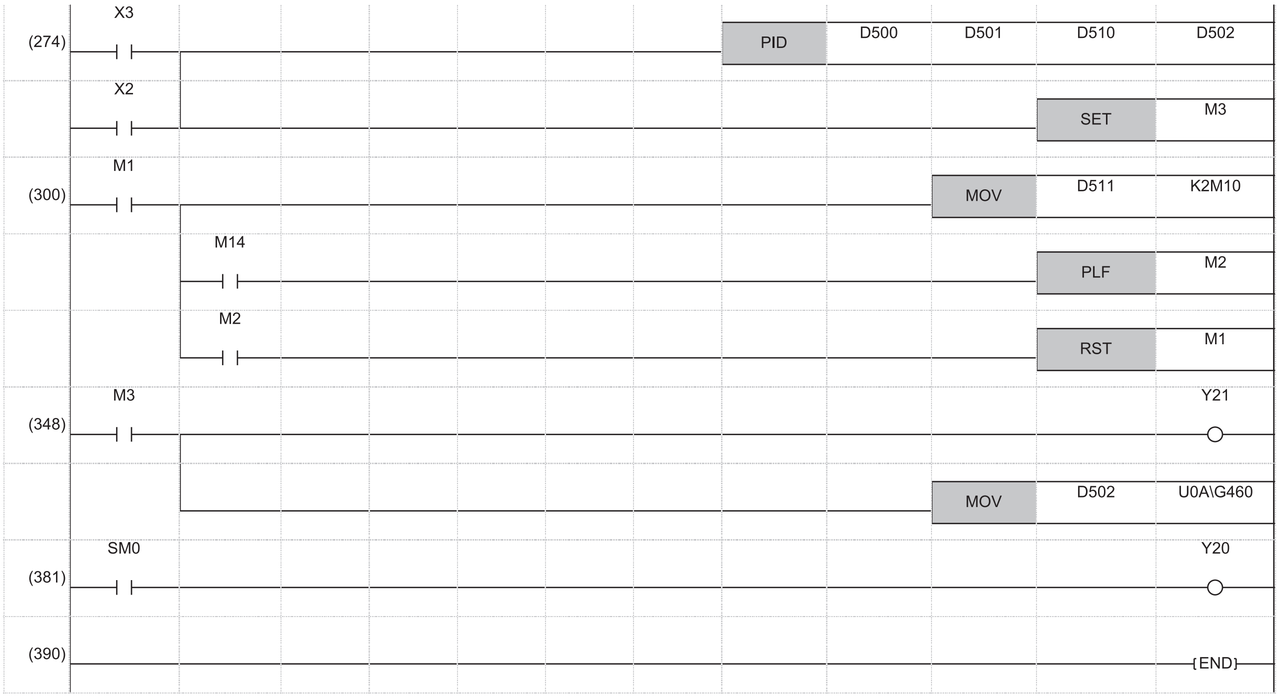
■Auto Tuning: 90% of maximum output



Auto tuning (step response method) + PID control program example



(0)	Set the set value. <50℃> Set the input filter constant (α). Set the derivative gain (K_D). <0%> Set the output upper limit value. <ON for 2 seconds> Set the output lower limit value. <ON for 0 seconds>
(88)	Start auto tuning.
(104)	Turn on the auto tuning action status flag. Set the sampling time for auto tuning (T_S). <3 seconds> Turn on the auto tuning execution bit in action setting (ACT) parameter. Set the output value for auto tuning. <ON for 1.8 seconds>
(191)	Set the sampling time (T_S) for normal PID action. <500ms>
(224)	Input the process value (PV) to the input/output data area from the A/D converter module.
(258)	Initialize the PID action.



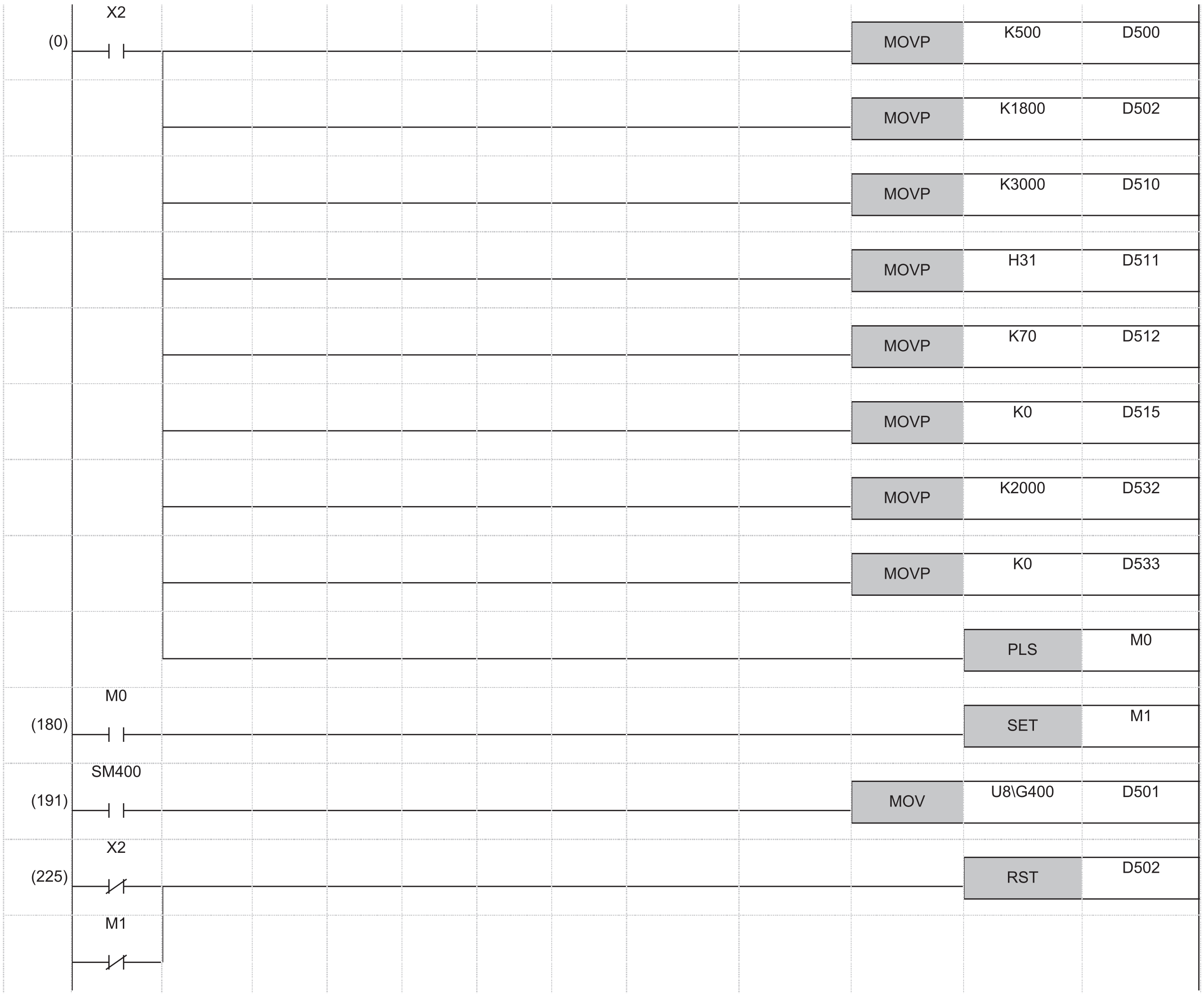
(274) Execute the PID instruction.
PID action is in progress.

(300) Check the auto tuning operation.
End the auto tuning processing.
Shift to the normal PID action.

(348) Heater output
Write the manipulated value (MV) to D/A converter module.

(381) Error

Auto tuning (step response method) program example

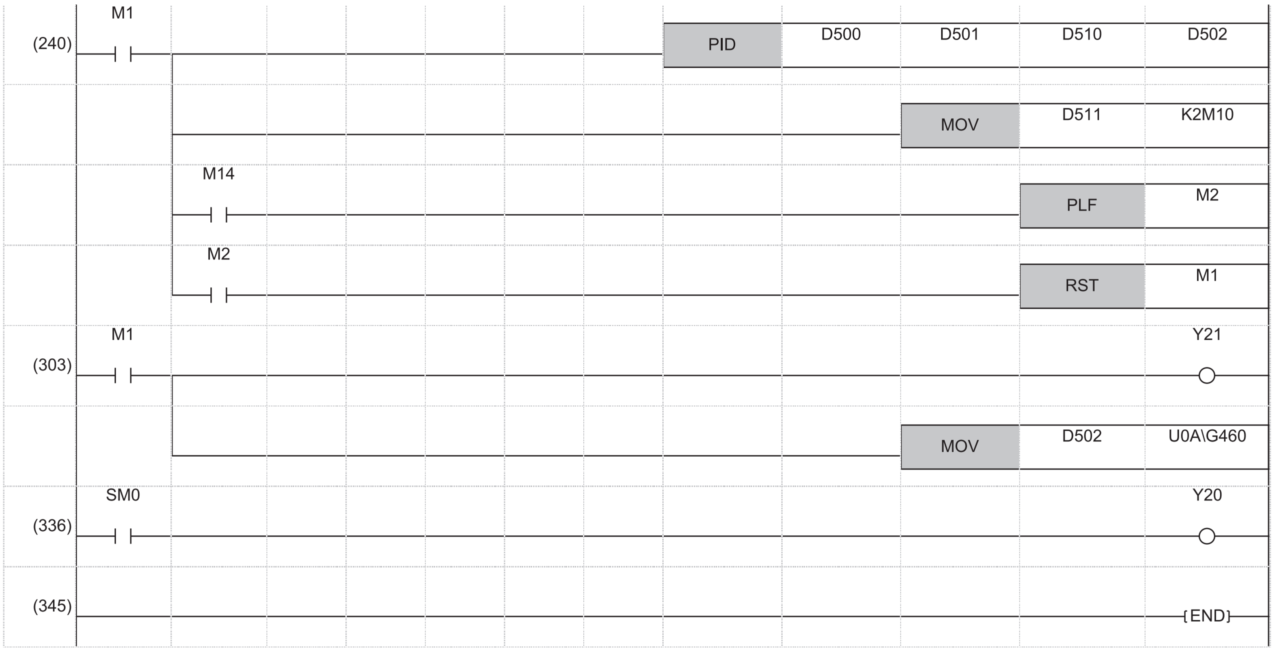


(0) Set the set value. <50℃>
Set the output value for auto tuning. <ON for 1.8 seconds>
Set the sampling time (T_S). <3 seconds>
Turn on the auto tuning execution bit in action setting (ACT) parameter.
Set the input filter constant (α).
Set the derivative gain (K_D). <0%>
Set the output upper limit value. <ON for 2 seconds>
Set the output lower limit value. <ON for 0 seconds>
Start auto tuning.

(180) Perform PID action.

(191) Input the process value (PV) to the input/output data area from the A/D converter module.

(225) Initialize the PID action.



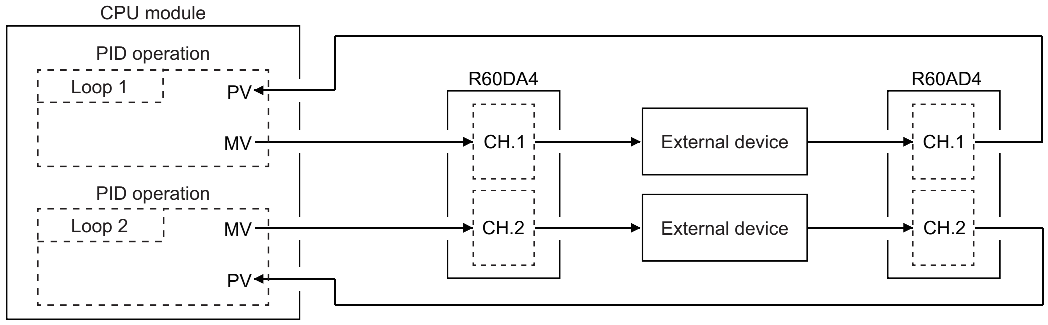
(240) Execute the PID instruction.
Check the auto tuning operation.
End the auto tuning processing.
Shift to the normal PID action.

(303) Heater output
Write the manipulated value (MV) to D/A converter module.

(336) Error

Appendix 5 PID Control Program Examples

This section provides examples of PID control programs where PID control instructions are used.
The following system configuration is used.



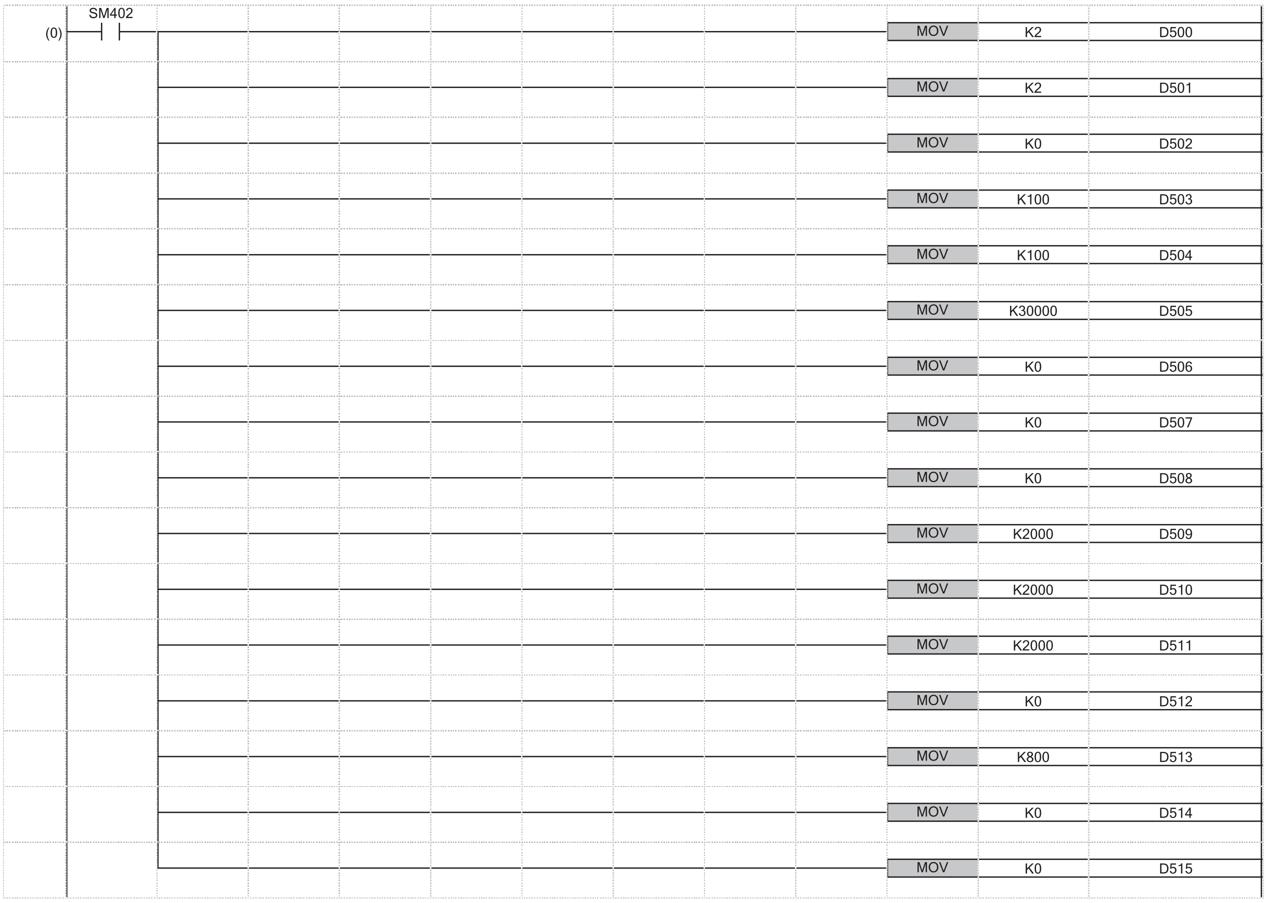
I/O number of the R60AD4: X/Y80 to X/Y8F, I/O number of the R60DA4: X/YA0 to X/YAF

Program examples for PID control in automatic mode

The program performs PID operation using the digital value input from the R60AD4 as a process value (PV), and outputs the obtained manipulated value (MV) from the R60DA4.
The program conditions are as follows:

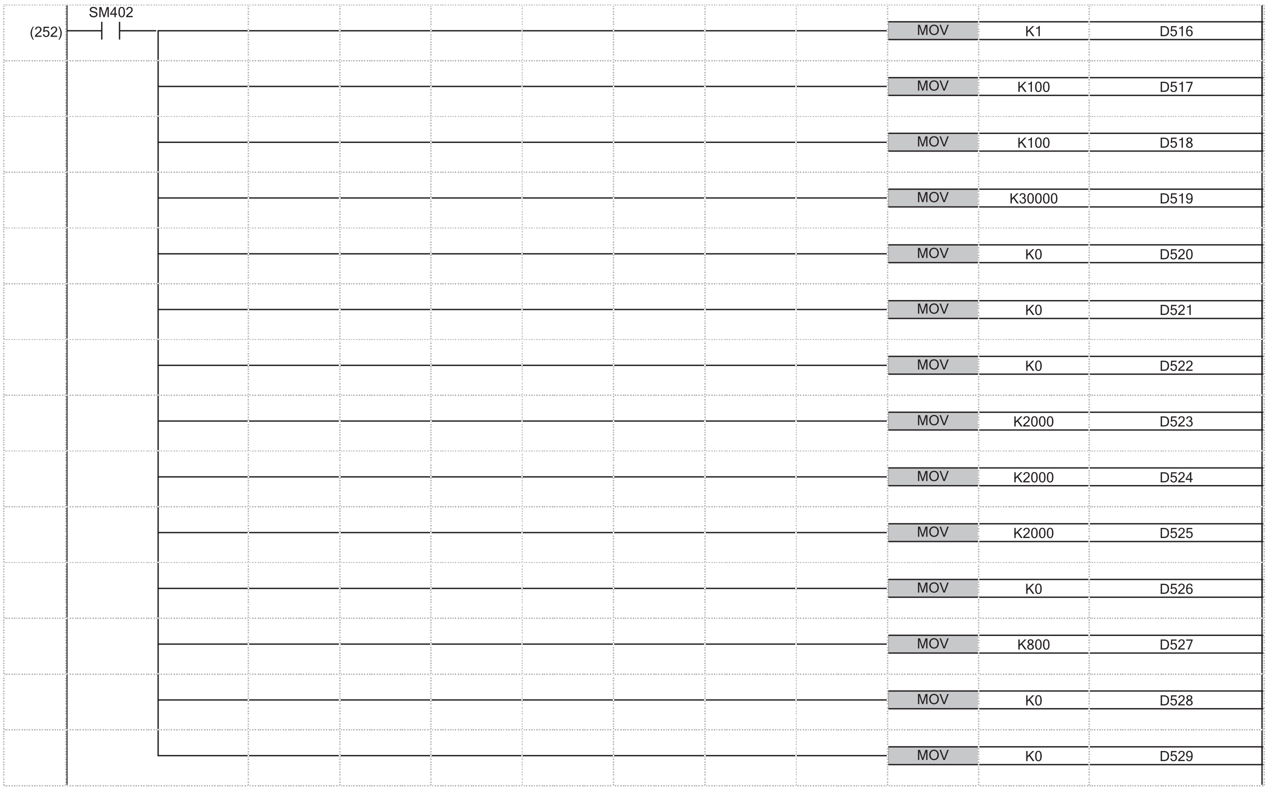
Item		Inexact differential	Exact differential
Number of loops that performs PID operation		2	
Sampling period		1 second	
Device where PID control data are set	Common data	D500, D501	
	Loop 1 data	D502 to D515	D502 to D511
	Loop 2 data	D516 to D529	D512 to D521
Device where I/O data are set	Common data	D600 to D609	
	Loop 1 data	D610 to D632	D610 to D627
	Loop 2 data	D633 to D655	D628 to D645
Set value (SV)	Loop 1	600	
	Loop 2	1000	
Device used to start/stop PID control	PID control start command	X0	
	PID control stop command	X1	
Digital value of the R60AD4 and R60DA4		0 to 2000	

Program example (inexact differential)

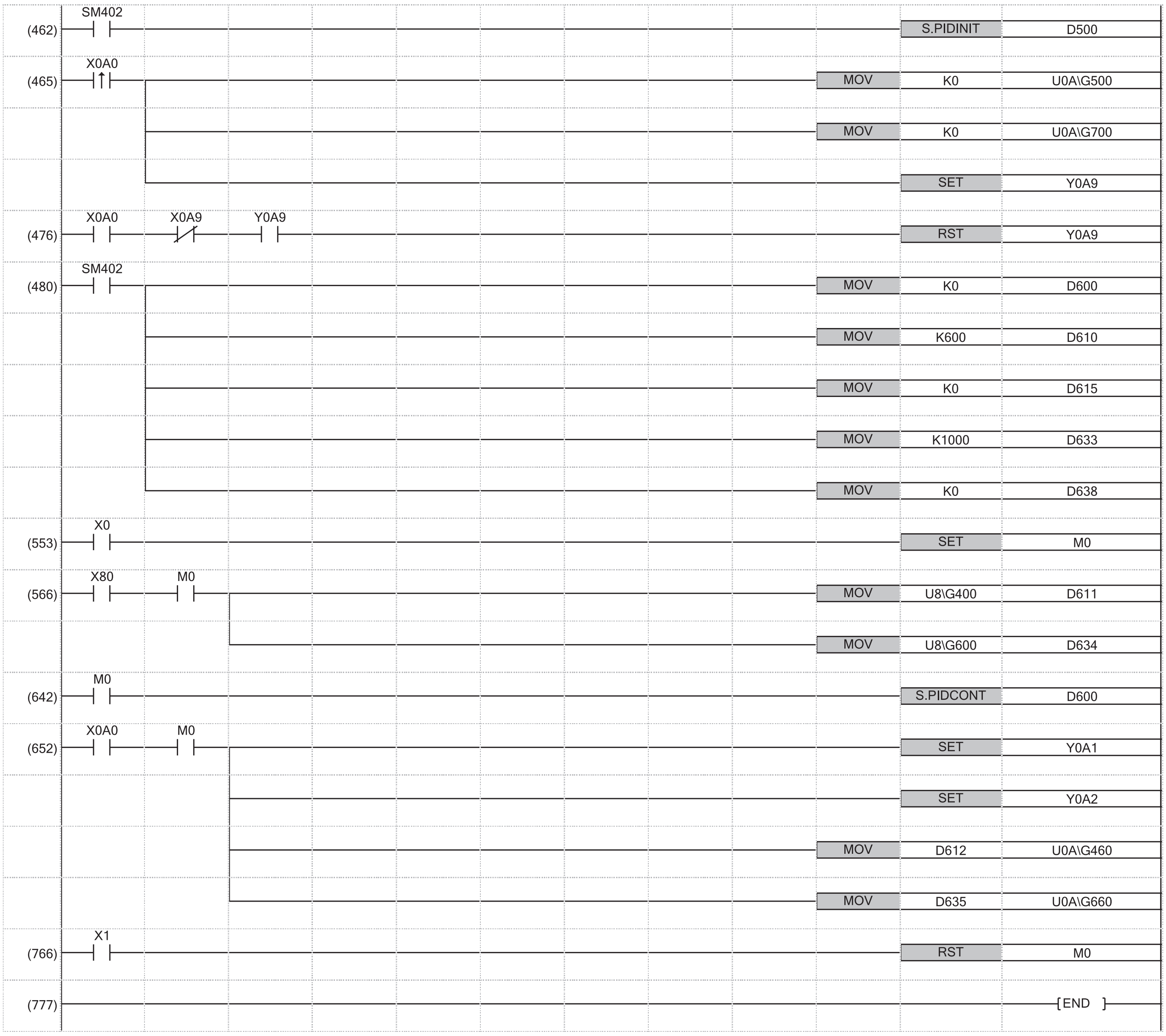


(0) Set PID control data common to all loops.

Set PID control data for loop 1.

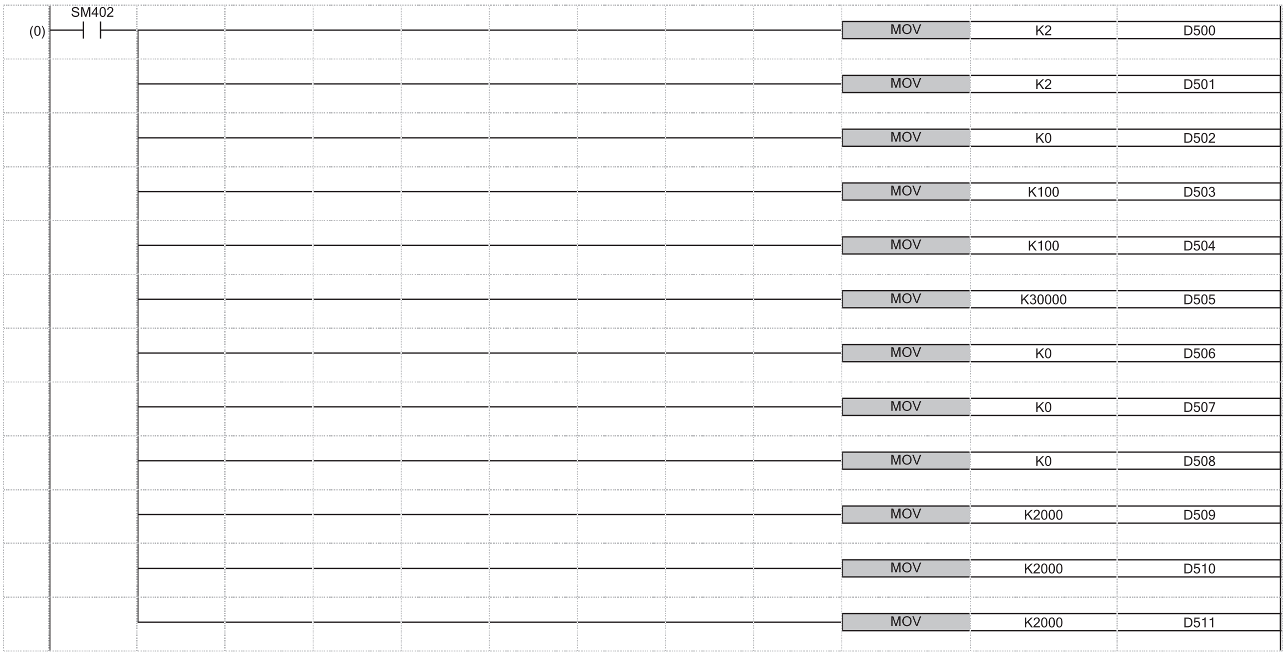


(252) Set PID control data for loop 2.



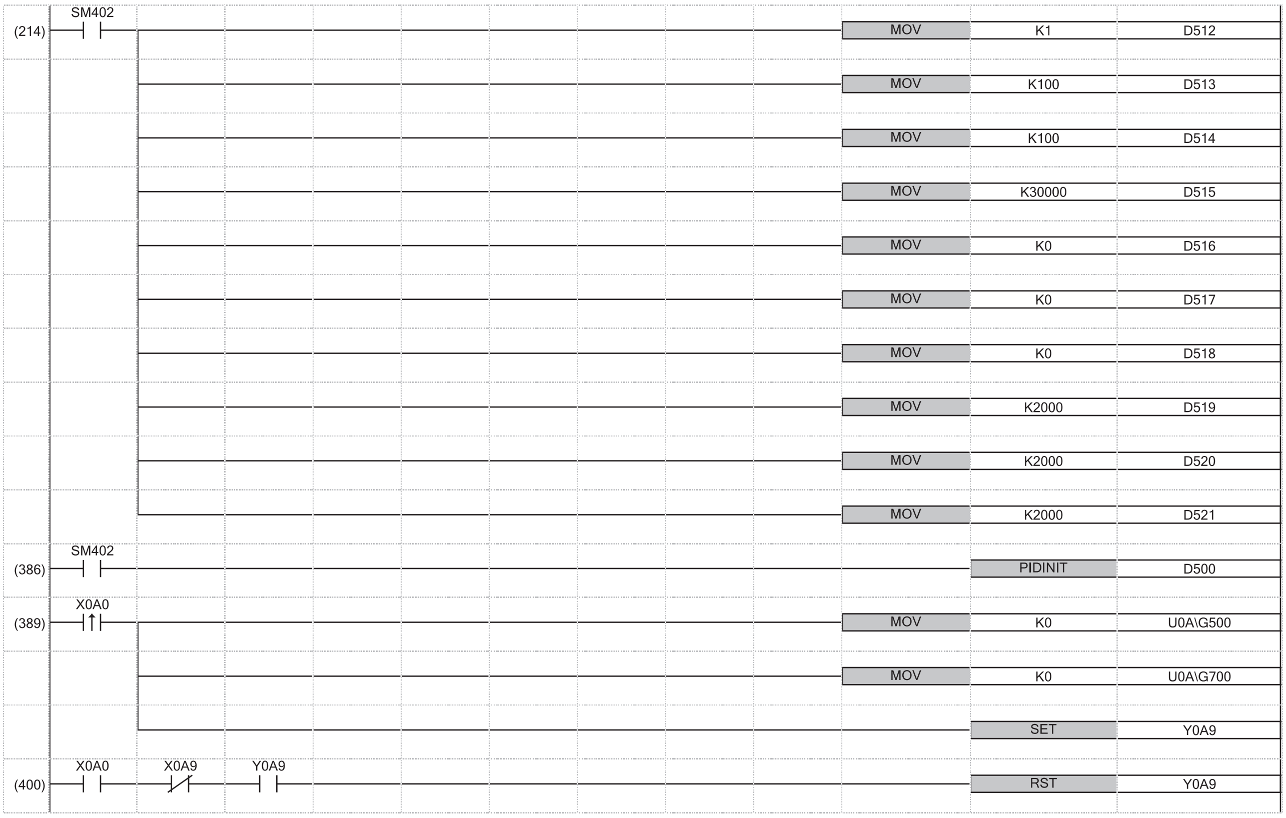
(462)	Set the PID control data that are set in D500 to D529.
(465) to (476)	Enable output of the R60DA4.
(480)	Set I/O control data common to all loops. Set I/O data for loop 1. Set I/O data for loop 2.

Program example (exact differential)



(0) Set PID control data common to all loops.

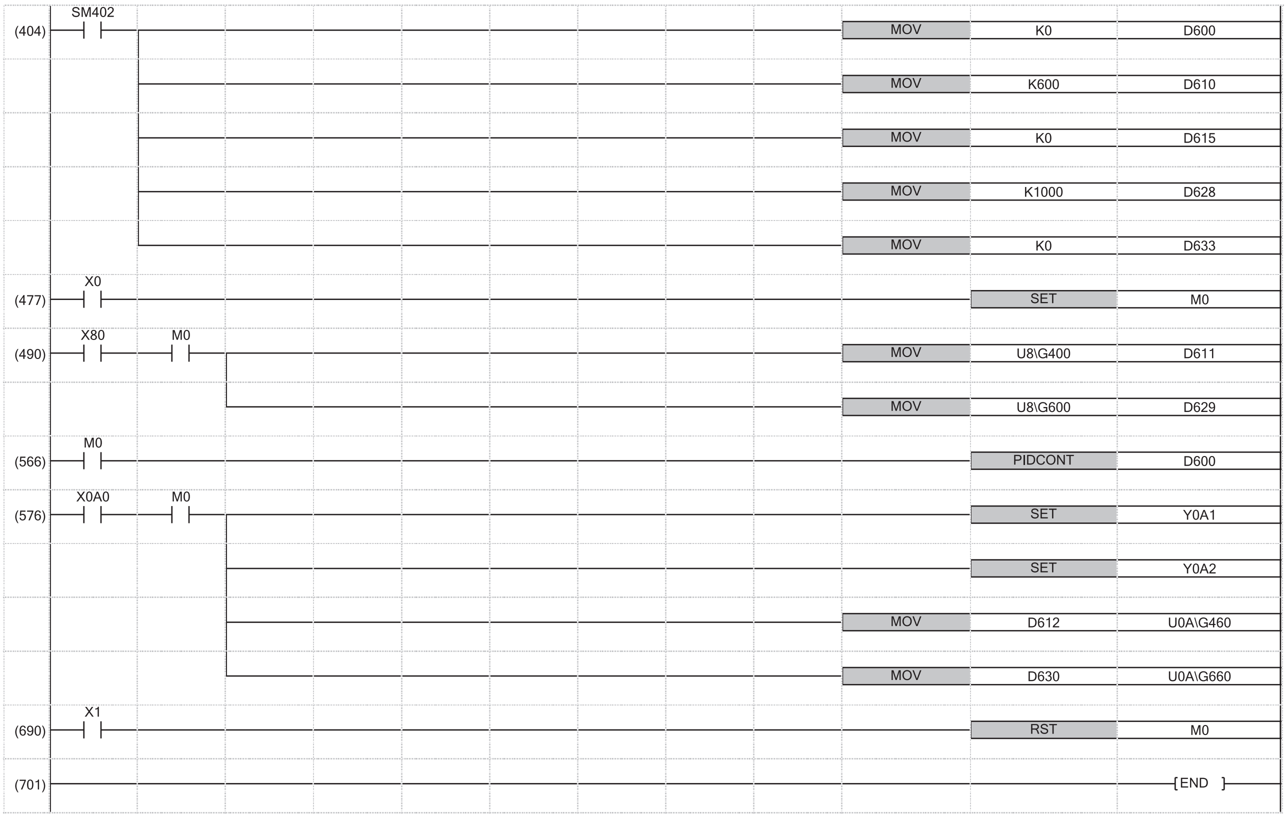
Set PID control data for loop 1.



(214) Set PID control data for loop 2.

(386) Set the PID control data that are set in D500 to D521.

(389) to (400) Enable output of the R60DA4.



(404) Set I/O control data common to all loops.

Set I/O data for loop 1.

Set I/O data for loop 2.

Program examples for PID control when switching modes

The program performs PID operation by switching an automatic mode and a manual mode.

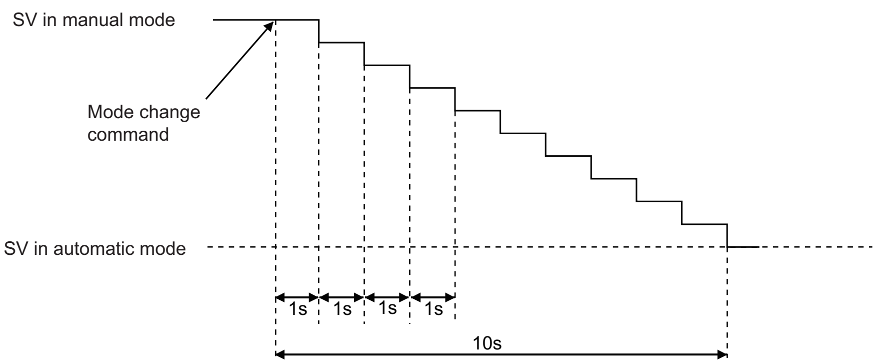
The program conditions are as follows:

Item		Inexact differential	Exact differential
Number of loops that performs PID operation		1	
Sampling period		1 second	
Device where PID control data are set	Common data	D500, D501	
	Loop 1 data	D502 to D515	D502 to D511
Device where I/O data are set	Common data	D600 to D609	
	Loop 1 data	D610 to D630	D610 to D627
Set value (SV) set by external digital switches and manipulated value (MV) in manual mode	SV	X30 to X3F	
	MV in manual mode	X20 to X2F	
Device used to start/stop PID control	PID control start command	X0	
	PID control stop command	X1	
Device used to switch the mode	SV set command	X3	
	MV (in manual mode) set command	X4	
	Automatic/manual switch command	X6 (Off: Automatic mode, On: Manual mode)	
Digital value of the R60AD4 and R60DA4		0 to 2000	
SM792, SM794		Off	

Point

In manual mode, the set value (SV) is automatically rewritten to the process value (PV) when PID operation is performed. Therefore, when the manual mode is returned to the automatic mode, the set value (SV) must be rewritten to the one used before the mode was switched.

The set value (SV) is rewritten in 10 steps as shown below:



The following formula is used to rewritten the set value (SV).

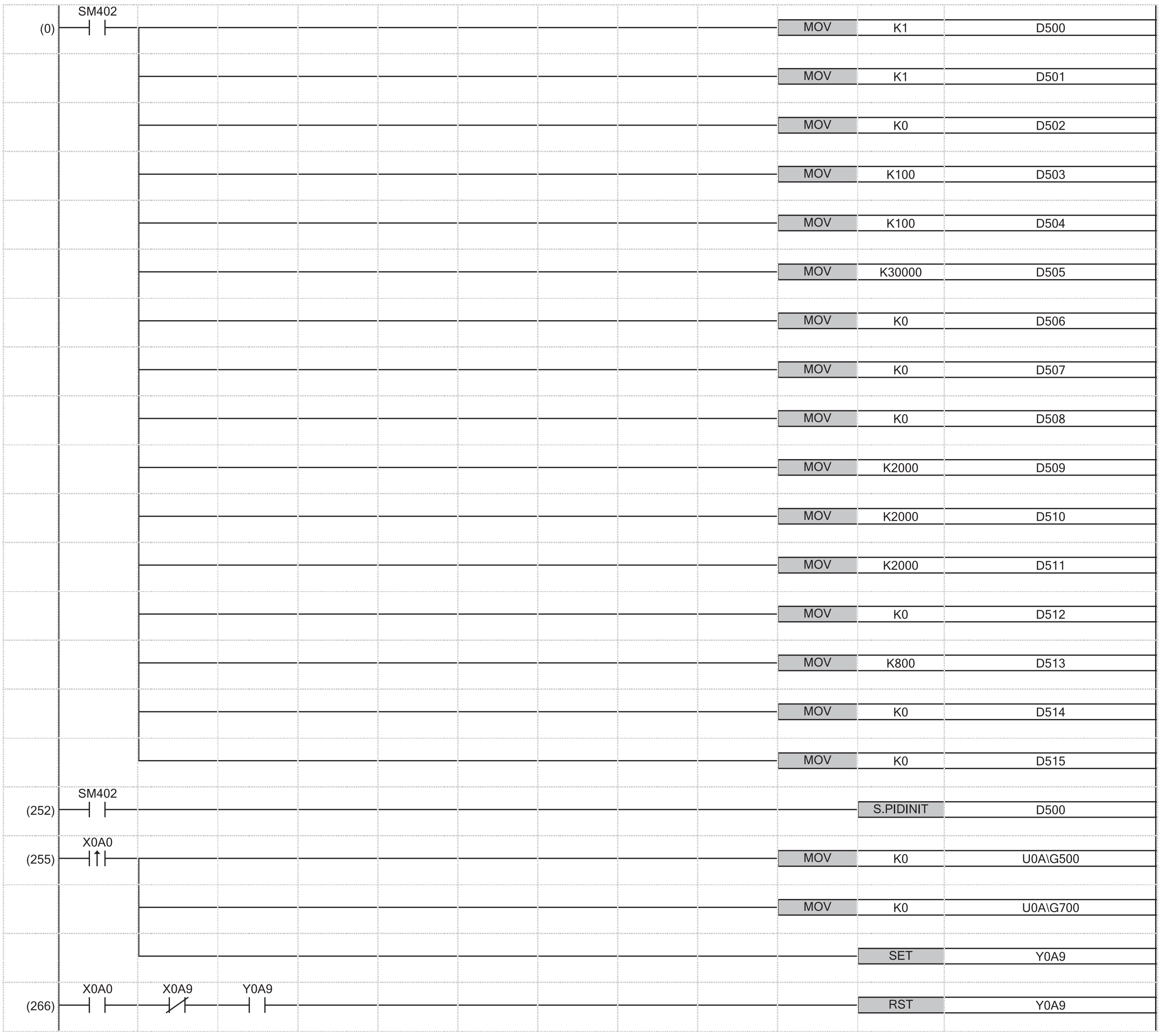
$$\frac{(SV_m)-(SV_a)}{10} = \boxed{\text{Subtraction value}} \cdots\cdots \boxed{\text{Remainder}}$$

SV_m: SV in manual mode

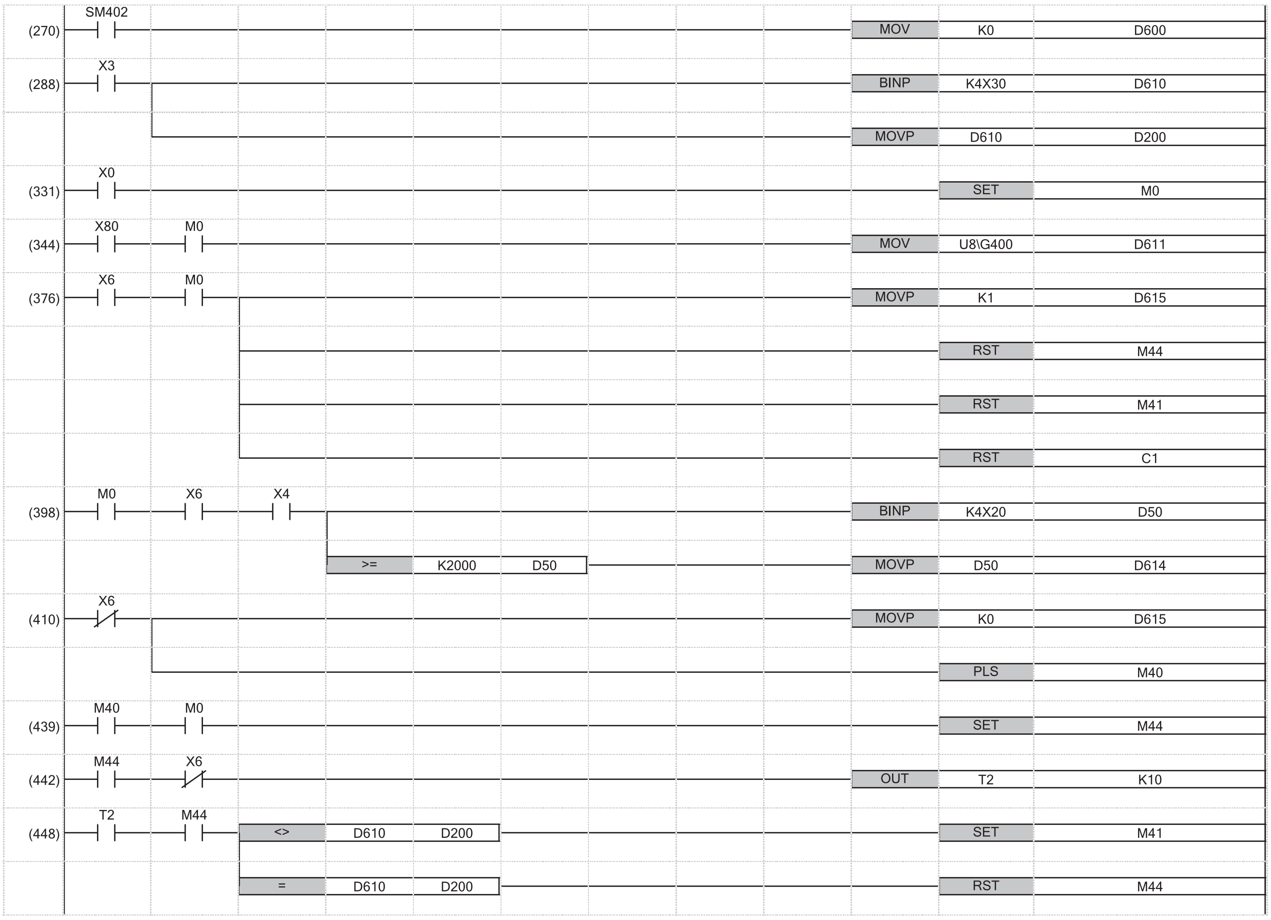
SV_a: SV in automatic mode

The value obtained with the above operational expression is subtracted from the set value (SV) every second. The remainder is subtracted in the first subtraction.

Program example (inexact differential)



(0)	Set PID control data common to all loops.
	Set PID control data for loop 1.
(252)	Set the PID control data that are set in D500 to D515.
(255)	Enable output of the R60DA4.

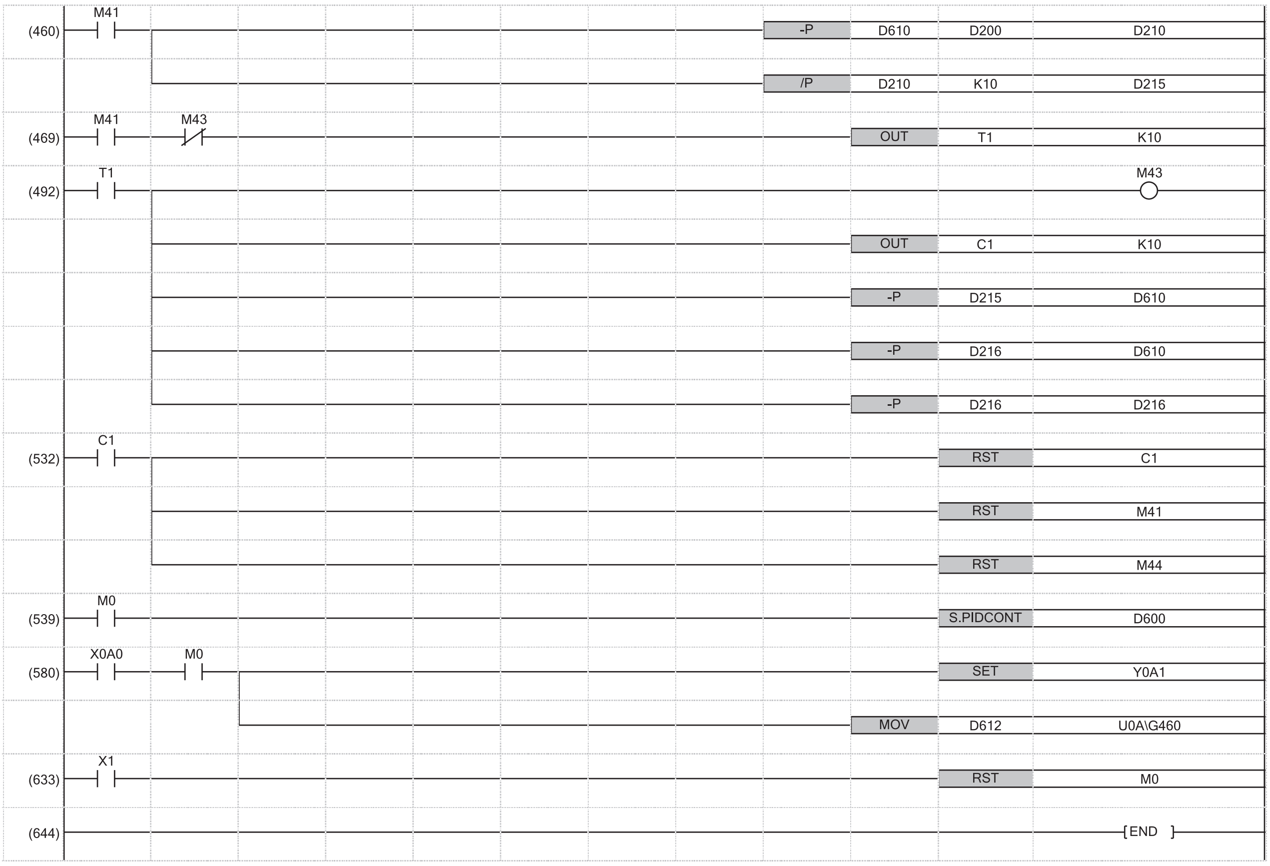


(270) Set I/O data.

(376) Reset the device used to switch manual mode to automatic mode.

(398) Externally set the manipulated value (MV).

(439) to (448) Perform the mode switch processing considering a delay in switching time. The mode switches to the automatic mode by turning on the automatic/manual switch command and performing PID operation.



(460) Store the quotient of the following operational expression in D215, and the remainder in D216.

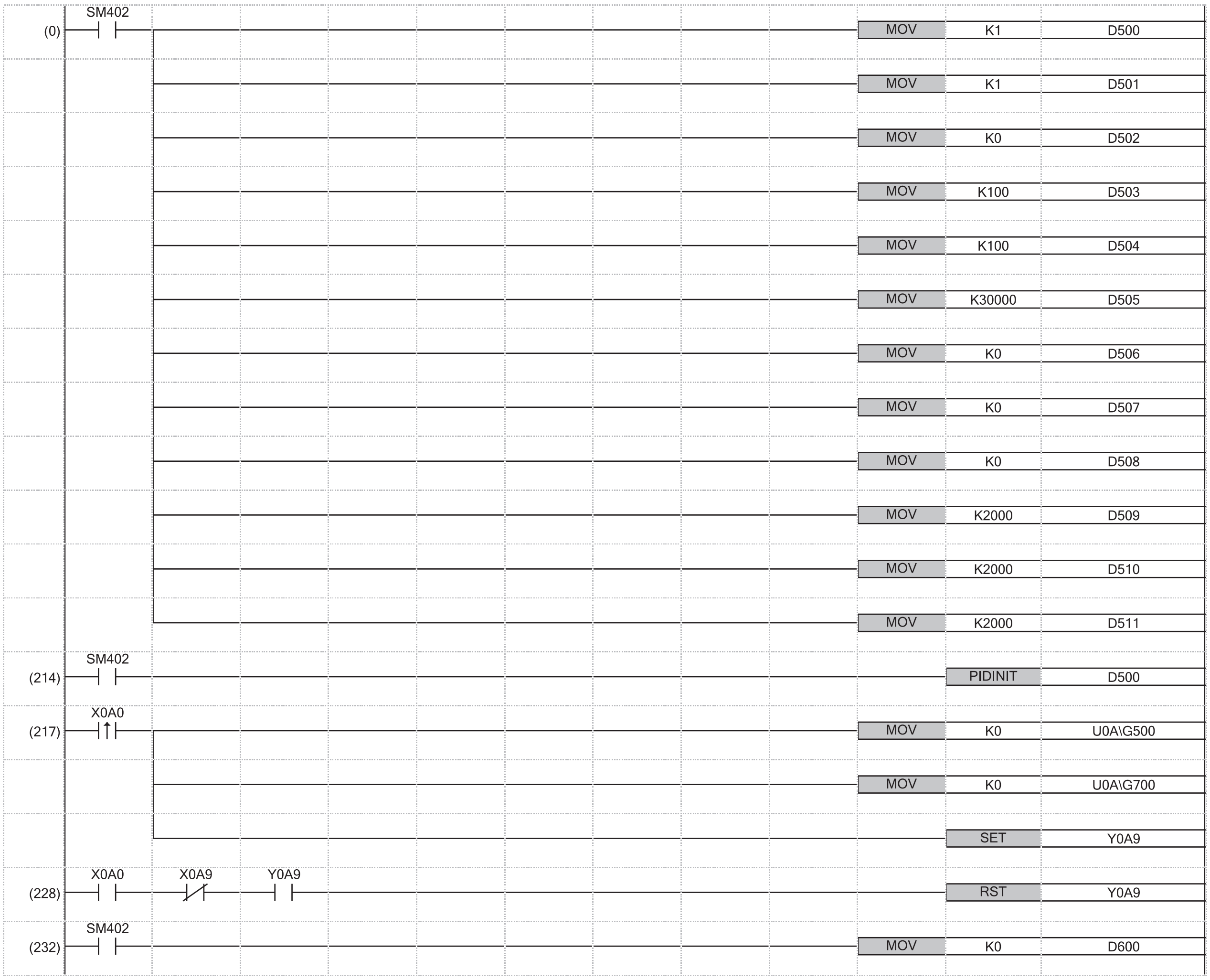
$$\frac{\left(SV_m\right) - \left(SV_a\right)}{10}$$

SV_m: SV in manual mode
SV_a: SV in automatic mode

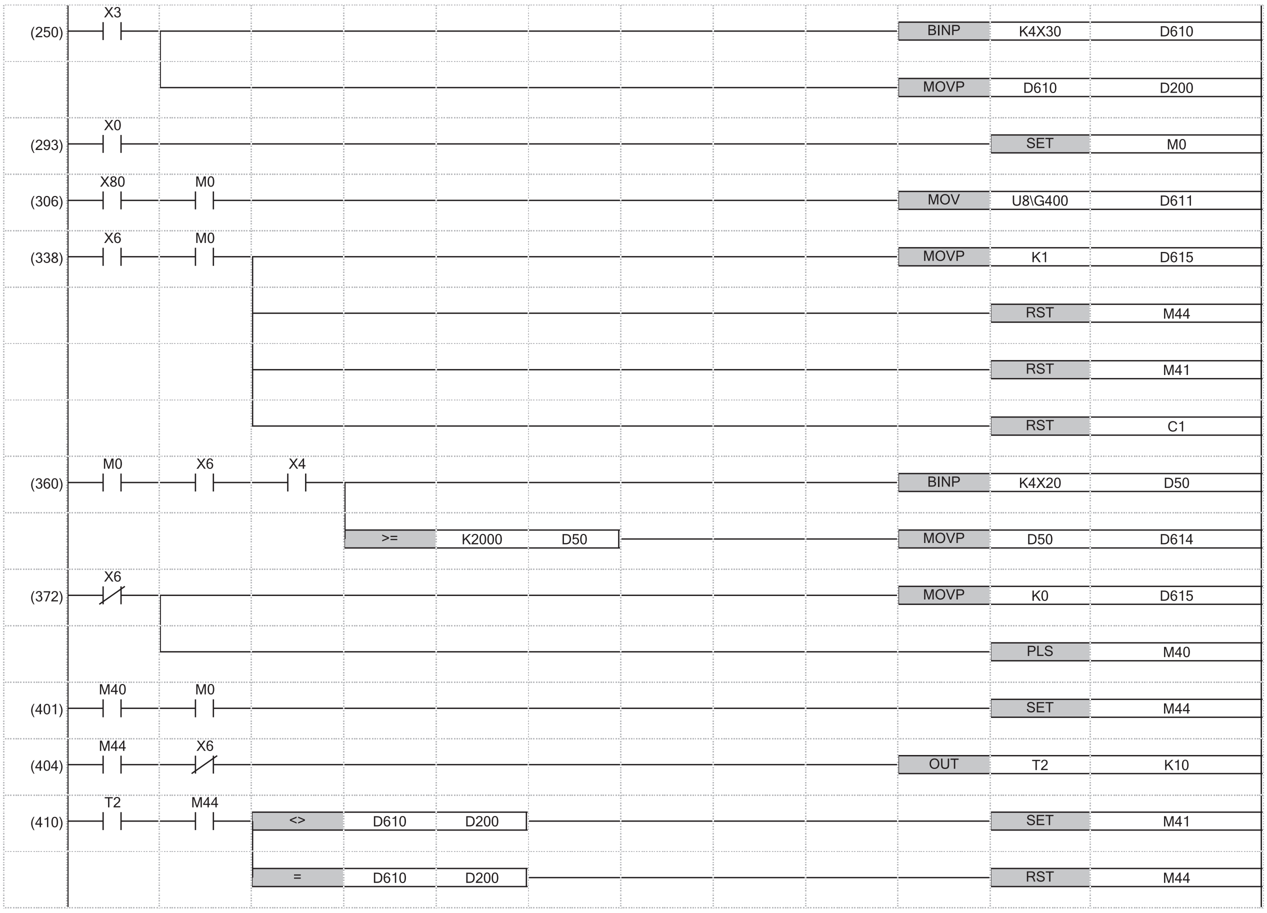
(469) to (492) Rewrite the set value (SV) to the value used in automatic mode.
The remainder is subtracted from the set value (SV) in the first subtraction.

(532) End the mode switch processing.

Program example (exact differential)



(0) to (214)	Set PID control data common to all loops. Set PID control data for loop 1.
(217) to (232)	Set the PID control data that are set in D500 to D511. Enable output of the R60DA4.

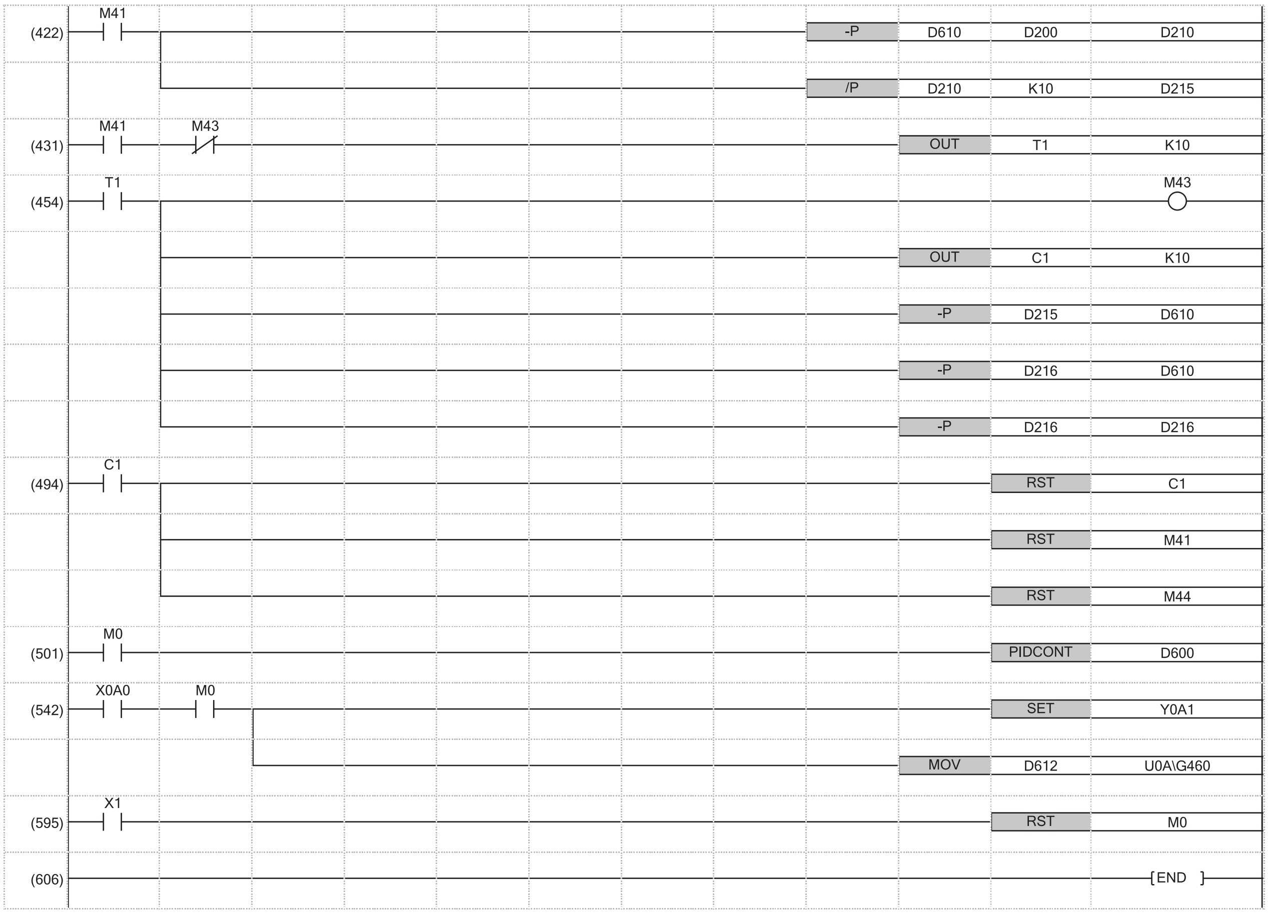


(250) Set I/O data.

(338) Reset the device used to switch manual mode to automatic mode.

(360) Externally set the manipulated value (MV).

(401) to (410) Perform the mode switch processing considering a delay in switching time. The mode switches to the automatic mode by turning on the automatic/manual switch command and performing PID operation.



(422) Store the quotient of the following operational expression in D215, and the remainder in D216.

$$\frac{\left(SV_m \right) - \left(SV_a \right)}{10}$$

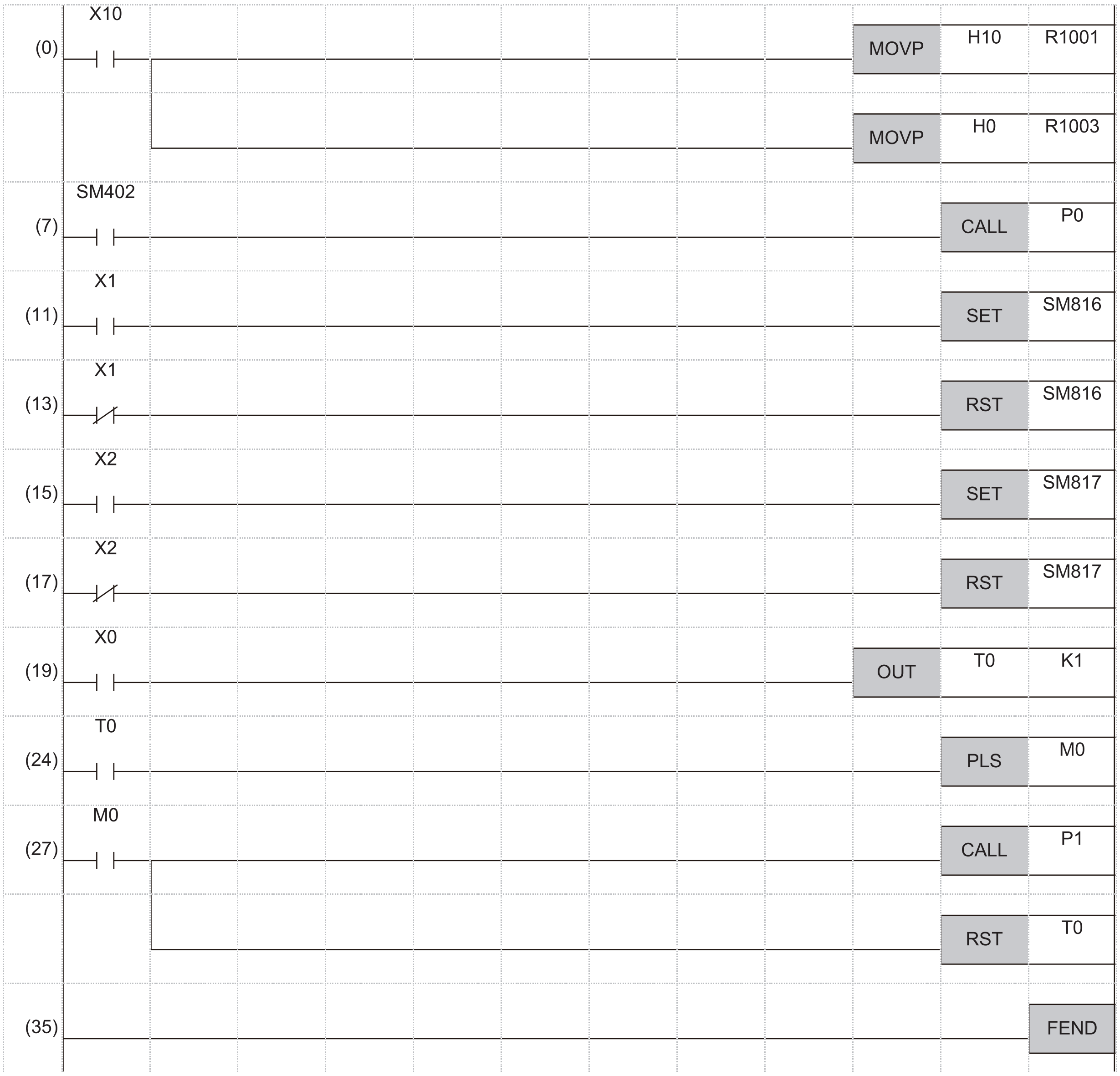
SV_m: SV in manual mode
SV_a: SV in automatic mode

(431) to (454) Rewrite the set value (SV) to the value used in automatic mode.
The remainder is subtracted from the set value (SV) in the first subtraction.

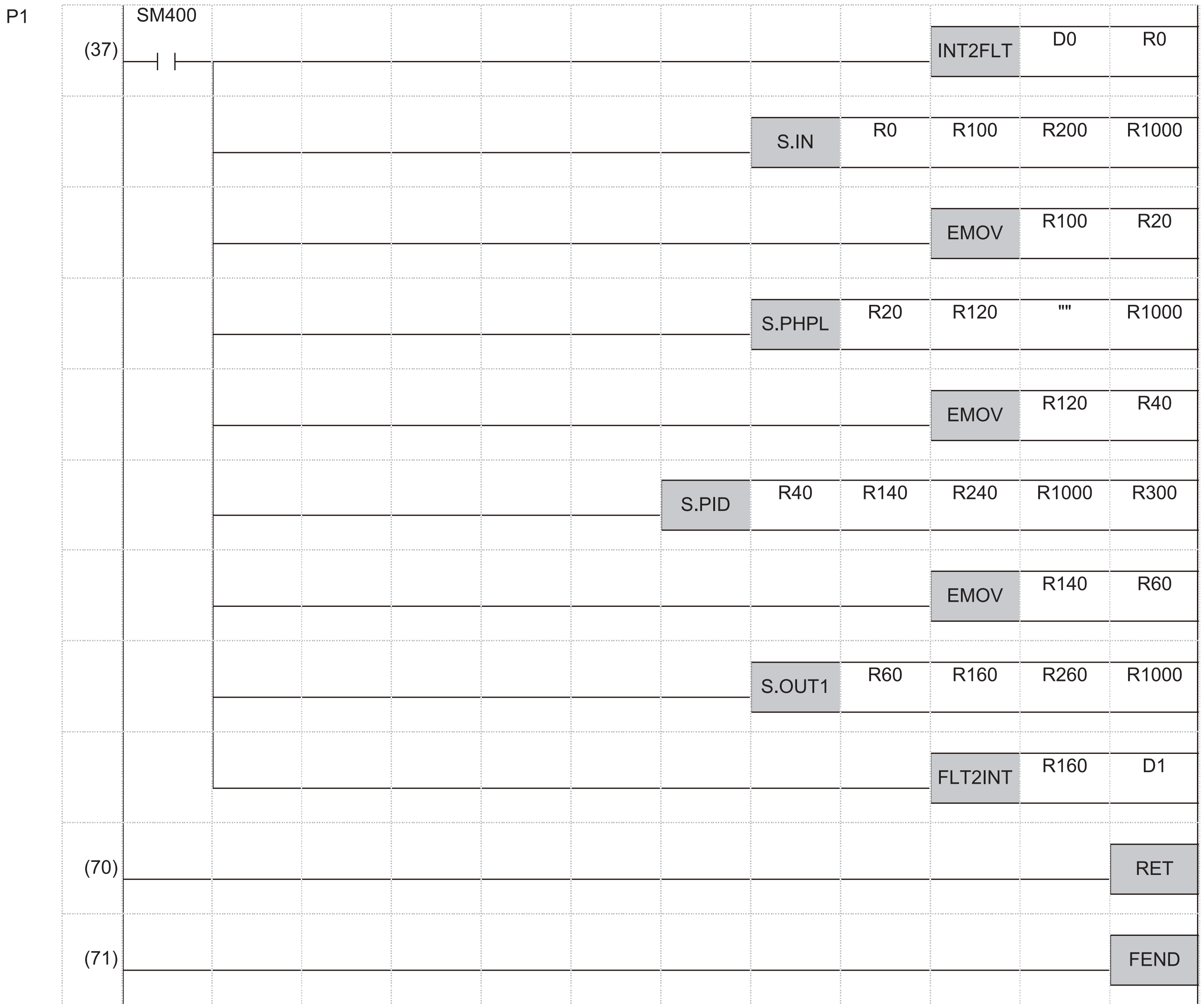
(494) End the mode switch processing.

Appendix 6 Process Control Program Examples

This section provides examples of process control programs using process control instructions.
With the following program, the control mode enters the manual mode when the power is turned on.
Turning on X10 enters the auto mode (AUTO) and performs PID control.



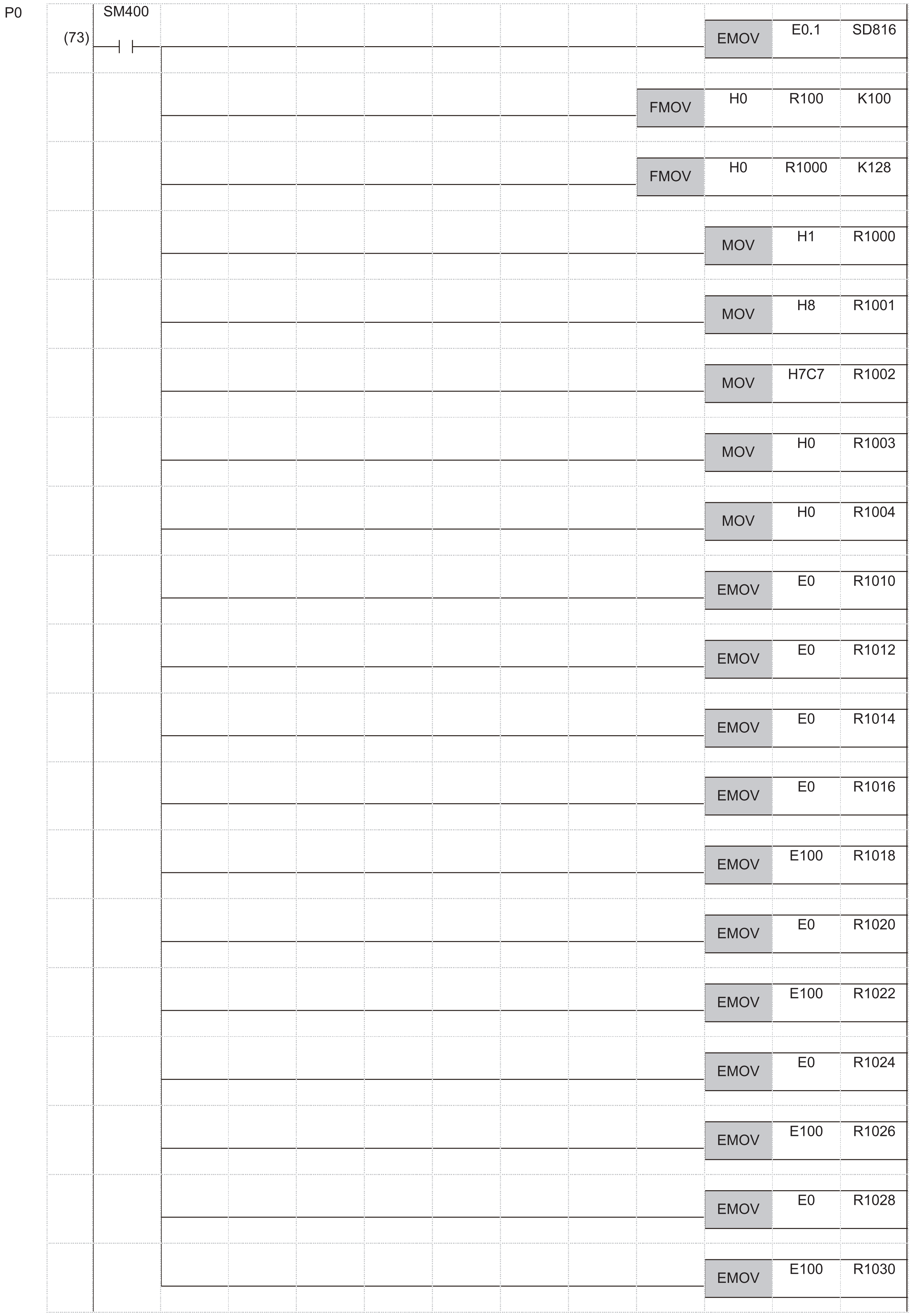
(0) Set the control mode to AUTO.
Set the alarm detection (ALM) to 0.
(7) Jump to the P0 label when SM402 is on.
(11) Set the last value hold processing.
(13) Reset the last value hold processing.
(15) Set the output value hold processing.
(17) Reset the output value hold processing.
(19) Adjust to the time set for the execution cycle.
(27) Jump to the P1 label when M0 is on.
Turning M0 on clears T0 to 0.
(35) The end of the sequence program in which SM402 is off is indicated.



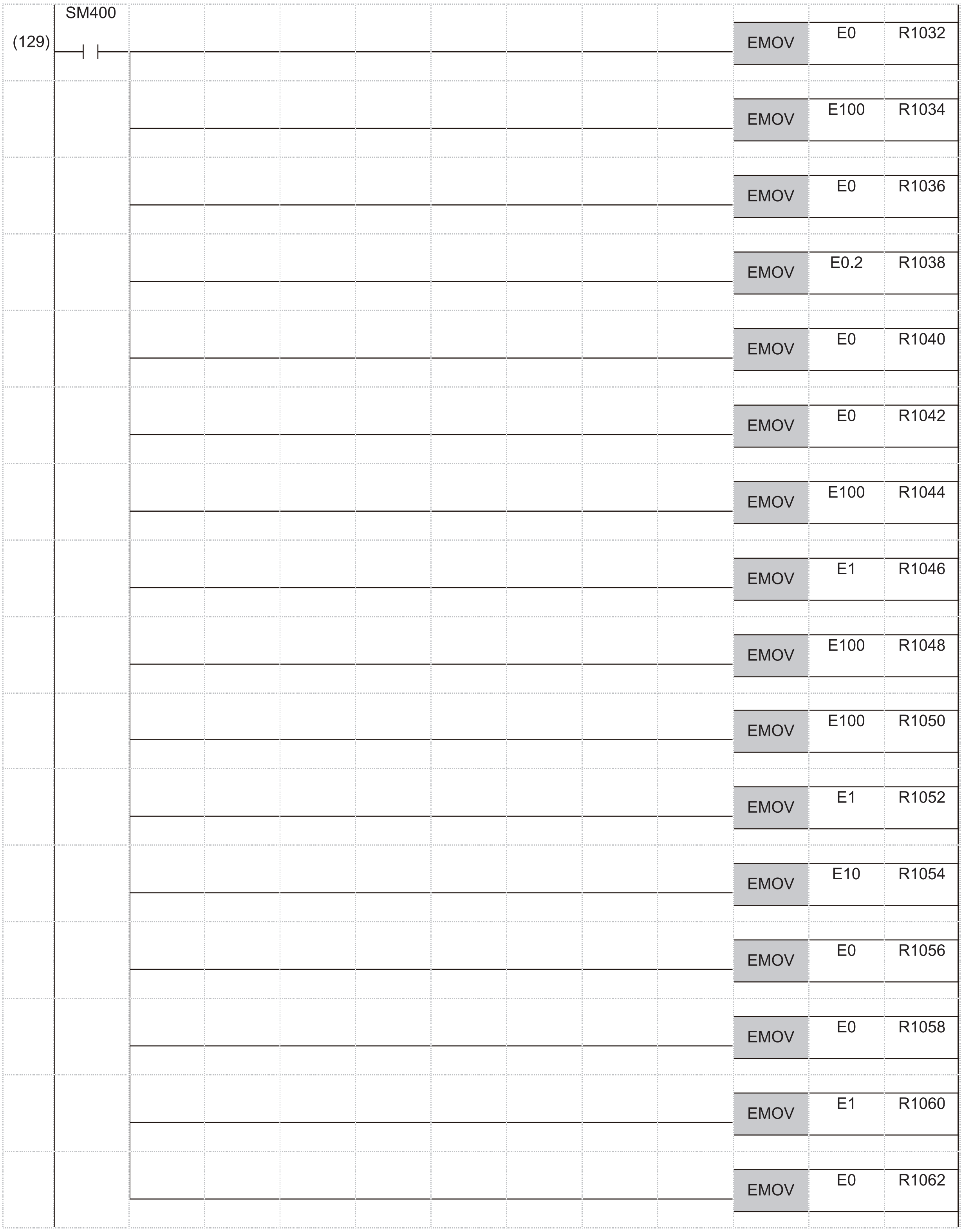
(37) Change the D0 value to a single-precision real number and stores it in R0.
Set each start device of the S.IN instruction.
Transfer the R100 value of the S.IN instruction to R20 of S.PHPL.
Set each start device of the S.PHPL instruction.
Transfer the R120 value of the S.PHPL instruction to R40 of S.PID.
Set the start device of the S.PID instruction.
Transfer the R140 value of the S.PID instruction to R60 of S.OUT1.
Set each start device of the S.OUT1 instruction.
Convert the single-precision real number in R160 and R161 to binary.

(70) The subroutine program ends.

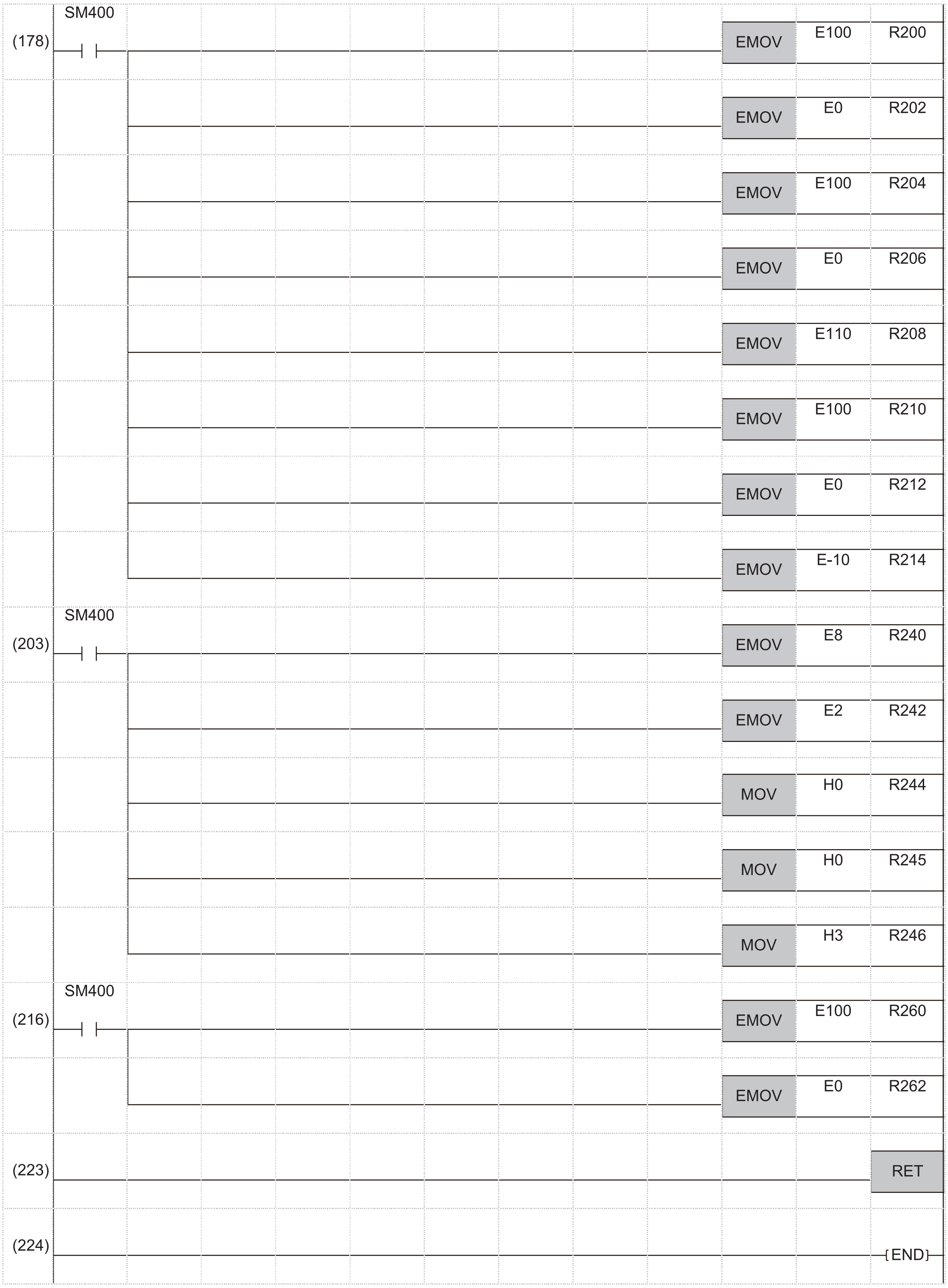
(71) The main routine program ends.



(73) Adjust the execution cycle to 0.1s. Clear the output values in S.IN, S.PHPL, S.PID, and S.OUT1 to 0.
Clear the loop tag to 0. Set the default value of the loop tag.



(129) Set the default value of the loop tag.



(178) Set the operation constant of the S.IN instruction.
(203) Set the operation constant of the S.PID instruction.
(216) Set the operation constant of the S.OUT1 instruction.
(223) The subroutine program ends.

Appendix 7 List of Loop Tag Memory Areas Used by Process Control Instructions

PID control (SPID), two-degree-of-freedom PID control (S2PID), sample PI control (SSPI)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SPID, S2PID	SSPI
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+10 +11	S.PHPL	PV	Process value	RL to RH	Single-precision real number	System	System
+12 +13	S.OUT1/ S.DUTY	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User/system	User/system
+14 +15	S.PID/ S.2PID/S.SPI	SV	Set value	RL to RH	Single-precision real number	User	User
+16 +17	S.PID/ S.2PID/S.SPI	DV	Deviation	-110 to 110 [%]	Single-precision real number	System	System
+18 +19	S.OUT1/ S.DUTY	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	User	User
+20 +21	S.OUT1/ S.DUTY	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	User	User
+22 +23	S.PHPL/ S.PID/ S.2PID/S.SPI	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	User	User
+24 +25	S.PHPL/ S.PID/ S.2PID/S.SPI	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	User	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	User	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	User	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	User	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	User	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	User	User
+46 +47	S.PID/ S.2PID/S.SPI	CT/ST	Control cycle/Operating time	0 to 999999 [s]	Single-precision real number	User (Set CT.)	User (Set ST.)
+48 +49	S.OUT1/ S.DUTY	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	User	User
+50 +51	S.PID/ S.2PID/S.SPI	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	User	User
+52 +53	S.PID/ S.2PID/S.SPI	P	Gain	0 to 999999	Single-precision real number	User	User

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SPID, S2PID	SSPI
+54 +55	S.PID/ S.2PID/ S.SPI/ S.OUT1/ S.DUTY	I[*1]	Integral constant	0 to 999999 [s]	Single-precision real number	User	User
+56 +57	S.PID/ S.2PID/S.SPI	D/STHT	Derivative constant/ Sampling cycle	0 to 999999 [s]	Single-precision real number	User (Set D.)	User (Set STHT.)
+58 +59	S.PID/ S.2PID/S.SPI	GW	Gap width	0 to 100 [%]	Single-precision real number	User	User
+60 +61	S.PID/ S.2PID/S.SPI	GG	Gap gain	0 to 999999	Single-precision real number	User	User
+62 +63	S.PID/ S.2PID/ S.SPI/ S.OUT1/ S.DUTY	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	System	System
+64 +65	S.2PID	α	Two-degree-of-freedom parameter α	0 to 1	Single-precision real number	User	—
+66 +67	S.2PID	β	Two-degree-of-freedom parameter β	0 to 1	Single-precision real number	User	—
+68 +69	S.DUTY	CTDUTY	Control output cycle	0 to 999999 [s]	Single-precision real number	User	—

[*1] The following instruction pairs use the same value in I.
- S.PID instruction and S.OUT1 instruction
- S.PID instruction and S.DUTY instruction
- S.2PID instruction and S.OUT1 instruction
- S.2PID instruction and S.DUTY instruction
- S.SPI instruction and S.OUT1 instruction

I-PD control (SIPD), blend PI control (SBPI)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SIPD	SBPI
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+10 +11	S.PHPL	PV	Process value	RH to RL	Single-precision real number	System	System
+12 +13	S.OUT1	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User/system	User/system
+14 +15	S.IPD/S.BPI	SV	Set value	RL to RH	Single-precision real number	User	User
+16 +17	S.IPD/S.BPI	DV	Deviation	-110 to 110 [%]	Single-precision real number	System	System
+18 +19	S.OUT1	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	User	User
+20 +21	S.OUT1	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	User	User
+22 +23	S.PHPL/ S.IPD/S.BPI	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	User	User
+24 +25	S.PHPL/ S.IPD/S.BPI	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	User	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	User	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	User	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	User	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	User	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	User	User
+46 +47	S.IPD/S.BPI	CT	Control cycle	0 to 999999 [s]	Single-precision real number	User	User
+48 +49	S.OUT1	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	User	User
+50 +51	S.IPD/S.BPI	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	User	User
+52 +53	S.IPD/S.BPI	P	Gain	0 to 999999	Single-precision real number	User	User
+54 +55	S.IPD/S.BPI/ S.OUT1	I[*1]	Integral constant	0 to 999999 [s]	Single-precision real number	User	User
+56 +57	S.IPD/S.BPI	D/SDV	Derivative constant/DV cumulative total	D: 0 to 999999 [s]	Single-precision real number	User	—
				SDV: -999999 to 999999 [%]	Single-precision real number	—	System
+58 +59	S.IPD/S.BPI	GW	Gap width	0 to 100 [%]	Single-precision real number	User	User
+60 +61	S.IPD/S.BPI	GG	Gap gain	0 to 999999	Single-precision real number	User	User

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SIPD	SBPI
+62 +63	S.IPD/ S.OUT1	MVP	MV internal operation value	-999999 to 999999 [%]	Single-precision real number	System	—

*1 The instruction pairs, S.IPD instruction and S.OUT1 instruction, S.BPI instruction and S.OUT1 instruction, use the same value in I.

Manual output (SMOUT), monitor (SMON)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SMOUT	SMON
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	—	User/system
+10 +11	S.PHPL	PV	Process value	RL to RH	Single-precision real number	—	System
+12 +13	S.MOUT	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User	—
+22 +23	S.PHPL	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	—	User
+24 +25	S.PHPL	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	—	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	—	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	—	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	—	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	—	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	—	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	—	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	—	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	—	User

Manual output with monitor (SMWM), PIDP control (SPIDP)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SMWM	SPIDP
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+10 +11	S.PHPL	PV	Process value	RL to RH	Single-precision real number	System	System
+12 +13	S.MOUT/ S.PIDP	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User	User/system
+14 +15	S.PIDP	SV	Set value	RL to RH	Single-precision real number	—	User
+16 +17	S.PIDP	DV	Deviation	-110 to 110 [%]	Single-precision real number	—	System
+18 +19	S.PIDP	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	—	User
+20 +21	S.PIDP	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	—	User
+22 +23	S.PHPL/ S.PIDP	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	User	User
+24 +25	S.PHPL/ S.PIDP	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	User	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	User	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	User	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	User	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	User	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	User	User
+46 +47	S.PIDP	CT	Control cycle	0 to 999999 [s]	Single-precision real number	—	User
+48 +49	S.PIDP	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	—	User
+50 +51	S.PIDP	DVL	Deviation limit value	0 to 100 [%]	Single-precision real number	—	User
+52 +53	S.PIDP	P	Gain	0 to 999999	Single-precision real number	—	User
+54 +55	S.PIDP	I	Integral constant	0 to 999999 [s]	Single-precision real number	—	User
+56 +57	S.PIDP	D	Derivative constant	0 to 999999 [s]	Single-precision real number	—	User
+58 +59	S.PIDP	GW	Gap width	0 to 100 [%]	Single-precision real number	—	User
+60 +61	S.PIDP	GG	Gap gain	0 to 999999	Single-precision real number	—	User

Two-position (on/off) control (SONF2), three-position (on/off) control (SONF3)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by	
						SONF2	SONF3
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system	User/system
+10 +11	S.PHPL	PV	Process value	RL to RH	Single-precision real number	System	System
+12 +13	S.ONF2/S.ONF3	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User/system	User/system
+14 +15	S.ONF2/S.ONF3	SV	Set value	RL to RH	Single-precision real number	User	User
+16 +17	S.ONF2/S.ONF3	DV	Deviation	-110 to 110 [%]	Single-precision real number	System	System
+18 +19	S.ONF2/S.ONF3	HSO	Hysteresis	0 to 999999	Single-precision real number	User	User
+20 +21	S.ONF3	HS1	Hysteresis	0 to 999999	Single-precision real number	—	User
+22 +23	S.PHPL	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	User	User
+24 +25	S.PHPL	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	User	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	User	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	User	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	User	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	User	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	User	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	User	User
+46 +47	S.ONF2/S.ONF3	CT	Control cycle	0 to 999999 [s]	Single-precision real number	User	User

Batch counter (SBC)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by
						SBC
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system
+10 +11	S.PSUM	SUM1	Integrated value (integral part)	0 to 2147483647	32-bit unsigned binary	System
+12 +13	S.PSUM	SUM2	Integrated value (decimal part)	0 to 2147483647	32-bit unsigned binary	System
+14 +15	S.BC	SV1	Set value 1	0 to 2147483647	32-bit unsigned binary	User
+16 +17	S.BC	SV2	Set value 2	0 to 2147483647	32-bit unsigned binary	User
+26 +27	S.BC	PH	Upper limit alarm value	0 to 2147483647	32-bit unsigned binary	User
+42 +43	S.BC	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User
+44 +45	S.BC	DPL	Variation rate alarm value	0 to 2147483647	32-bit unsigned binary	User

Ratio control (SR)

Offset	Instruction used	Item	Name	Recommended range	Data type	Set by
						SR
+1	Shared among the instructions	MODE	Control mode	0 to FFFFH	16-bit unsigned binary	User/system
+3		ALM	Alarm detection	0 to FFFFH	16-bit unsigned binary	User/system
+4		INH	Disable alarm detection	0 to FFFFH	16-bit unsigned binary	User/system
+10 +11	S.PHPL	PV	Process value	RL to RH	Single-precision real number	System
+12 +13	S.OUT2	MV	Manipulated value	-10 to 110 [%]	Single-precision real number	User/system
+14 +15	S.R	SPR	Set value	-999999 to 999999	Single-precision real number	User
+16 +17	S.R	BIAS	Bias	-999999 to 999999 [%]	Single-precision real number	User
+18 +19	S.OUT2	MH	Output upper limit value	-10 to 110 [%]	Single-precision real number	User
+20 +21	S.OUT2	ML	Output lower limit value	-10 to 110 [%]	Single-precision real number	User
+22 +23	S.PHPL	RH	Engineering value upper limit	-999999 to 999999	Single-precision real number	User
+24 +25	S.PHPL	RL	Engineering value lower limit	-999999 to 999999	Single-precision real number	User
+26 +27	S.PHPL	PH	Upper limit alarm value	RL to RH PL < PH	Single-precision real number	User
+28 +29	S.PHPL	PL	Lower limit alarm value	RL to RH PL < PH	Single-precision real number	User
+30 +31	S.PHPL	HH	Upper upper limit alarm value	RL to RH PH ≤ HH	Single-precision real number	User
+32 +33	S.PHPL	LL	Lower lower limit alarm value	RL to RH LL ≤ PL	Single-precision real number	User
+38 +39	S.IN	α	Filter coefficient	0 to 1	Single-precision real number	User
+40 +41	S.PHPL	HS	Upper/lower limit alarm hysteresis	0 to 999999 [%]	Single-precision real number	User
+42 +43	S.PHPL	CTIM	Variation rate alarm check time	0 to 999999 [s]	Single-precision real number	User
+44 +45	S.PHPL	DPL	Variation rate alarm value	0 to 100 [%]	Single-precision real number	User
+46 +47	S.R	CT	Control cycle	0 to 999999 [s]	Single-precision real number	User
+48 +49	S.OUT2	DML	Output variation rate limit value	0 to 100 [%]	Single-precision real number	User
+50 +51	S.R	DR	Variation rate limit value	0 to 999999	Single-precision real number	User
+52 +53	S.R	RMAX	Ratio upper limit value	-999999 to 999999	Single-precision real number	User
+54 +55	S.R	RMIN	Ratio lower limit value	-999999 to 999999	Single-precision real number	User
+56 +57	S.R	Rn	Ratio current value	-999999 to 999999	Single-precision real number	System

MEMO

INDEX

INSTRUCTION INDEX

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
June 2014	SH(NA)-081266ENG-A	First edition
July 2014	SH(NA)-081266ENG-B	Error correction
October 2014	SH(NA)-081266ENG-C	■Added or modified parts Section 1.3, Chapter 11, 13, "Execution condition" (for each instruction)
January 2015	SH(NA)-081266ENG-D	■Added or modified parts Chapter 10, "FBD/LD" (for each instruction)
April 2015	SH(NA)-081266ENG-E	■Added or modified parts Section 7.26, 7.27, 13.5, 13.6, 16.8, 17.2
August 2015	SH(NA)-081266ENG-F	■Added or modified parts CONDITIONS OF USE FOR THE PRODUCT, TERMS, Section 1.2, 1.5, Chapter 2, 3, 4, Section 5.3, 5.5, 7.4, 7.26, 7.29, Chapter 12, WARRANTY, "SFC devices and safety devices" (for each instruction)
August 2015	SH(NA)-081266ENG-G	■Added or modified part Section 12.1
January 2016	SH(NA)-081266ENG-H	■Added or modified parts TERMS, Section 2.2, 2.3, 2.4, 2.5, 3.9, 3.10, 6.1, 6.5, 6.7, 7.9, 7.19, 7.20, 7.25, 7.26, 8.4, Chapter 9, 13, 20, Section 21.4, 21.5, Chapter 22, 23, Appendix 1, 2, 3, 4
May 2016	SH(NA)-081266ENG-I	■Added or modified parts TERMS, Section 2.3, 2.9, 3.4, 7.7, 8.5, Chapter 13, 14, 15, Section 15.2, 17.1, 17.2, 17.3, 17.4, 18.1, 18.2, 18.3, 18.4, 18.5, 18.6
June 2016	SH(NA)-081266ENG-J	■Added or modified part Section 8.5
October 2016	SH(NA)-081266ENG-K	■Added or modified parts Section 6.7, 7.22, 7.27
January 2017	SH(NA)-081266ENG-L	■Added or modified parts TERMS, Chapter 12, 21
May 2017	SH(NA)-081266ENG-M	■Added or modified parts Section 3.4, 6.1, 6.2, 6.6, 7.6, 7.8, 7.9, 7.21, 7.25, 18.9, 31.1, 31.2, Appendix 1

Japanese manual number: SH-081226-M

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.

However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place. Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.

(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.

1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.

(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation 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. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

TRADEMARKS

SH(NA)-081266ENG-M(1705)

MODEL: R-P-MF-E

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN